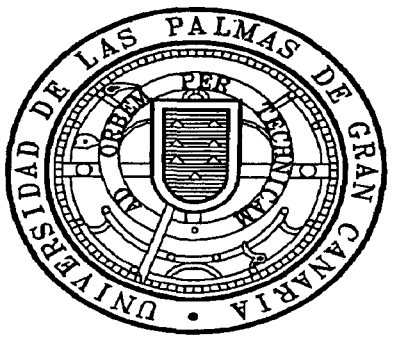
UNIVERSIDAD DE LAS PALMAS

ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA DE TELECOMUNICACIÓN



TRABAJO FIN DE CARRERA

TÍTULO: Software didáctico de planificación de un sistema de telefonía móvil automática.

ESPECIALIDAD: Equipos electrónicos.

AUTOR: Julio César Caballero Ortíz.

TUTORA: Itziar Goretti Alonso González.

AÑO: 1995

PROYECTO FIN DE CARRERA

Tutora

Presidente **Secretario** **Vocal**

Calificación: 7

Introducción

El objetivo fundamental de este Proyecto de Fin de Carrera es el de ofrecer un software en el entorno de Microsoft Windows 3.1, que permita conocer y planificar los aspectos principales de un Sistema de Telefonía Móvil Automática (TMA). Siendo TMA el nombre que se le ha dado en España a la Telefonía Móvil Celular.

El software va dirigido a los alumnos matriculados en la asignatura Sistemas de Telecomunicación de la especialidad de Telefonía y Transmisión de Datos de la EUITT de las Palmas de Gran Canaria. La aplicación se utilizará como un módulo de prácticas en el que los alumnos podrán realizar la planificación de un sistema de TMA con sólo definir ciertas magnitudes básicas que permitirán a la aplicación presentar los cálculos obtenidos. El usuario trabajará de una forma interactiva con la aplicación. Cuando se introduzcan datos contradictorios al hacer la planificación del sistema, la aplicación se lo hará saber al usuario, indicándole los pasos a seguir para corregirlos.

El área de trabajo de este software consta de tres recuadros a modo de ventanas. En un recuadro encontramos la presentación en pantalla de un posible sistema de TMA en el que se simulará el desplazamiento de cinco automóviles con teléfono portátil, que serán los abonados del sistema. Estos abonados móviles se desplazarán dentro y fuera del sistema de TMA. Un abonado móvil podrá realizar y recibir llamadas telefónicas. El estado de la comunicación, así como su localización podrán ser observadas en otro de los recuadros de la aplicación. Por último en el tercer recuadro podremos observar los cálculos realizados sobre las magnitudes seleccionadas (geométricas, radioeléctricas, de tráfico y sobre los móviles). Además la aplicación posee su propio sistema de ayuda Windows del que se podrá obtener información sobre el manejo de la aplicación y sobre los conceptos generales de la Telefonía Móvil Celular.

El proyecto se ha realizado bajo el entorno de Microsoft Windows debido a la ventaja que tiene dicho entorno sobre el MS-DOS. Permite la multitarea, lo que significa que podemos tener varias aplicaciones en ejecución al mismo tiempo. Además la programación gráfica de Windows es más potente y se está convirtiendo en el entorno de programación por excelencia de los grandes diseñadores de aplicaciones para ordenadores IBM PC y compatibles.

El lenguaje de programación utilizado ha sido el C++. Este lenguaje permite realizar una programación orientada a objetos (POO). Con la POO podemos crear aplicaciones más complejas que la programación procedimental convencional, a parte de hacer el código fuente de los programas más legibles al tener las funciones relacionadas entre sí en estructuras que forman parte de la aplicación, consiguiendo mejorar la depuración de los programas. Como Windows es un entorno orientado a objetos, la utilización de este lenguaje es la más adecuada.

El compilador utilizado ha sido el Borland C/C++ 3.1 para Windows. Este compilador es uno de los más utilizados por los programadores de Windows, debido a que ha sabido relacionar la forma en que trabaja Windows con la programación orientada a objetos que permite el C++ , creando ObjectWindows. Se trata de un modo opcional y alternativo para escribir aplicaciones Windows. ObjectWindows proporciona una potente biblioteca de funciones dirigida a objetos para el entorno estándar de programación Windows. Utilizando las propiedades de la herencia, utilizamos el código ya escrito que realiza la tarea repetitiva necesaria para escribir aplicaciones Windows.

Estructura de la memoria:

La memoria consta de cuatro capítulos y dos anexos, además al final del libro se lista toda la bibliografía que se ha utilizado en el desarrollo de este proyecto.

En el capítulo 1 se hace una introducción sobre el entorno de programación Microsoft Windows, la programación orientada a objetos (POO), el lenguaje de programación C++ y el compilador de Borland C++. Todos ellos utilizados en la realización de este proyecto.

El capítulo 2 describe todos los aspectos a tener en cuenta al hacer la planificación de un sistema de telefonía móvil celular, así como la evolución histórica de la telefonía móvil.

En el capítulo 3 se hace un análisis completo de todos los menús y ventanas pertenecientes al software de la aplicación TMA.

El capítulo 4 explica todas las clases (objetos) y funciones que se han desarrollado para la creación de este proyecto.

En el anexo 1 encontraremos todos los ficheros fuente utilizados para crear el ejecutable (TMA.EXE).

Por último, en el anexo 2 se explica cómo poder crear un sistema de ayuda de Windows y se incluyen todos los archivos utilizados para crear TMA.HLP, nuestro sistema de ayuda.

1

ENTORNO Y LENGUAJE DE PROGRAMACIÓN

En este capítulo encontraremos una descripción de los siguientes temas:

- El entorno de programación Windows 3.1.
- La programación orientada a objetos (POO).
- El lenguaje de programación C++.
- El compilador Borland C++.

En estos temas se pretende explicar por qué se ha utilizado el entorno Windows 3.1, el lenguaje C++ como lenguaje orientado a objetos y el compilador Borland C++, para desarrollar este proyecto. Además se pretende introducir al lector en el mundo de la programación en Windows con un enfoque orientado a objetos, comentando las características principales de cada uno de ellos.

1.1 Microsoft Windows 3.1

En este tema encontraremos una descripción de Windows así como su relación con el sistema operativo MS-DOS y con el BIOS. También veremos los módulos ejecutables que lo componen, la forma en que trabaja Windows mediante mensajes y la ventaja de utilizar los manejadores para referenciar a las ventanas, los menús y todos los demás objetos de Windows.

1.1.1 ¿ Por qué Windows ?

El sistema básico de entrada/salida del ordenador (BIOS), proporciona una capa aislante entre una aplicación y el hardware del ordenador. El BIOS está siempre presente, además el sistema operativo para discos MS-DOS, proporciona una capa inicial sobre el BIOS para aislar más aún a las aplicaciones del hardware subyacente y simplificar las tareas de los programas. Hasta ahora, el resto era responsabilidad del código ejecutable de la aplicación almacenado en un archivo con extensión .EXE.

Microsoft Windows está disponible en una manera similar al BIOS o al DOS. Antes de ejecutar una aplicación para Windows, el usuario debe instalar el propio Windows al igual que debe arrancar el ordenador con un sistema DOS. Una vez instalado, Windows está continuamente disponible para todos los programadores de aplicaciones, tal y como lo están el BIOS y el DOS, proporciona servicios a todos los programadores de aplicaciones.[8]

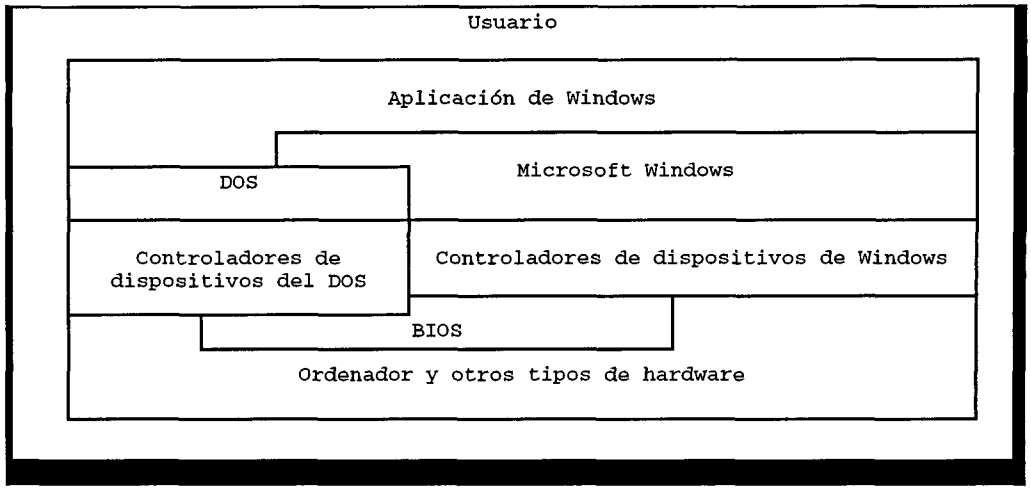


Figura 1-1

Windows es una extensión gráfica del MS-DOS, capaz de ejecutar múltiples programas de manera concurrente. Proporciona un conjunto normalizado de objetos de la interfaz de usuario, como ventanas, menús e iconos. Esto ayuda a hacer que los programas Windows sean de fácil aprendizaje y empleo. Tradicionalmente, los programas de aplicación se han controlado secuencialmente y no por sucesos, es decir, que el programa dicta la secuencia que el usuario tiene que seguir para conseguir el objetivo del programa.

Windows proporciona una interfaz de usuario gráfica multitarea, o GUI (Graphic User Interface), que favorece la creación de programas interactivos. Los programas controlados por sucesos permiten que el usuario dicte los pasos requeridos para completar una tarea dada. El trabajo gráfico en Windows tiene además otra ventaja, se basa en una metodología GDI (Graphics Device Interface), que proporciona gráficos independientes del dispositivo. Esto significa que podemos utilizar las mismas funciones para dibujar en pantalla, en una impresora, en un plotter o en un fichero en disco. Aún más, podemos diseñar nuestras propias funciones independientes del dispositivo.

El conjunto de funciones se conocen como el API (Application Program Interface) de Windows, y consta de todas las funciones necesarias para la creación de aplicaciones Windows.

Los programas que se ejecutan bajo Windows 3.x pueden ser aplicaciones Windows, y aplicaciones no Windows. Las aplicaciones no Windows pueden ejecutarse bajo MS-DOS y bajo Windows, pero no pueden utilizar las posibilidades de Windows. Las aplicaciones Windows sólo se pueden ejecutar bajo un entorno Windows, ya que dependen de las funciones del API de Windows.

Windows 3.x no es un sistema operativo como tal, porque precisa del soporte del DOS. Por ello es presentado como un entorno de trabajo multitarea sin protocolos de prioridades. Los programas Windows no son interrumpidos por el sistema operativo. En vez de ello, cada programa interrumpe voluntariamente su propio funcionamiento para permitir que se ejecuten otros programas.

Microsoft Windows se compone de tres programas ejecutables:

- El **KERNEL.EXE** es el responsable de la planificación de las tareas, la gestión de la memoria y de los recursos y de la interacción con el DOS.

- El **GDI.EXE** es el responsable de los gráficos, tanto en pantalla como en la impresora.

- El **USER.EXE** proporciona la características de gestión de ventanas, el soporte de las entradas realizadas por el usuario y de las herramientas utilizadas para ello y también el soporte de las comunicaciones.

Cuando el usuario ejecuta el WIN.EXE, los tres módulos se instalan automáticamente y permanecen disponibles para otros programas.[5]

La mayoría del código que históricamente se incluía en los distintos archivos de las aplicaciones del usuario se encuentra ahora en Windows, a disposición de cualquier programa, por lo que:

- Windows utiliza una gran cantidad de memoria RAM cuando está instalado.

- Algunas aplicaciones para Windows muy sofisticadas residen sin embargo en archivos ejecutables increíblemente pequeños.

1.1.2 Tipos de funciones de Windows

El API de Windows tiene una amplia gama de funciones disponibles. Pueden dividirse por comodidad en categorías dependiendo de la parte de Windows donde residan:

- Funciones del **KERNEL.EXE**, que incluyen funciones para gestionar la memoria, interactuar con el sistema operativo, gestionar los recursos y las comunicaciones.

- Funciones del **GDI.EXE**, que incluyen funciones para escribir texto, gráficos y mapas de Bits en pantallas y dispositivos de salida de forma independiente del dispositivo.

- Funciones del **USER.EXE**, que incluyen funciones para manejar mensajes, manipular ventanas y cuadros de diálogo, soportar menús y cursores, controlar el Portapapeles y crear salida del sistema.

1.1.3 Los mensajes de Windows

La forma que tiene Windows de trabajar es mediante la gestión de mensajes, que son atendidos en una cola según el orden de llegada. Los mensajes de Windows pueden enviarse a un objeto de las siguientes maneras:

- Por el usuario, cuando selecciona una opción de un menú, pulsa una tecla, mueve o pulsa un botón del ratón, etc.

- Por el programa, si una de las funciones envía un mensaje al objeto.

- Por el propio Windows, si se realizan llamadas al API que hacen que Windows envíe varios mensajes.

- Por otras aplicaciones, si la aplicación coopera con otras utilizando la característica de intercambio dinámico de datos DDE de Windows.[6]

1.1.4 Los manejadores (handles)

Windows utiliza manejadores para identificar muchos tipos diferentes de objetos, como son los menús, iconos, controladores, reserva de memoria, dispositivos de salida, lápices y pinceles, ventanas e incluso instancias. Windows nos permite ejecutar más de una copia de la misma aplicación a la vez, y mantener el estado de cada una de esas instancias proporcionando a cada una su propio manejador individual. Normalmente utilizamos un manejador como un índice en una tabla interna.

Windows conserva los recursos del sistema utilizando el mismo código para todas las instancias de una aplicación. Sólo se distingue y maneja de forma única el segmento de datos de cada instancia. Además la primera instancia de una aplicación juega un papel muy importante, ya que crea todos los objetos necesarios para el funcionamiento de la aplicación.[2]

1.2 La programación orientada a objetos

La idea principal de la programación estructurada o procedimental es descomponer un programa en unidades más pequeñas denominadas funciones o procedimientos dependiendo del lenguaje. Cada una de las funciones tiene un propósito claramente definido, así como una interfaz a las otras funciones del programa. La información se pasa entre funciones utilizando parámetros, y las funciones pueden tener datos locales a los cuales no se puede acceder fuera del ámbito del procedimiento.

La programación estructurada utiliza el método descendente y el refinamiento sucesivo, y cada una de las funciones se invocan sucesivamente. Sin embargo a medida que el tamaño de los programas aumenta se hacen muy complejos. Los tipos de datos se procesan en muchas funciones, y cuando se producen cambios en ellos, las modificaciones se deben hacer en cada posición que actúa sobre esos tipos de datos dentro del programa. Esta tarea consume mucho tiempo por lo que se hace necesario un método para restringir el acceso a los datos. Otro problema que se suele presentar es que es difícil realizar un diseño estructurado porque las funciones estructuradas de datos no modelizan bien en el mundo real. Existen otros problemas como la dificultad de crear nuevos tipos de datos.

La programación orientada a objetos (POO) se presenta como una alternativa al modelo tradicional y a sus problemas. La orientación a objetos es un modelo de desarrollar software que consiste en construir éste como unión de objetos, cada uno de los cuales aglutina sus datos y su comportamiento.

1.2.1 ¿ Qué son los objetos ?

Los objetos son tipos de datos definidos por el usuario. Se parecen a las estructuras y las uniones del C en que también pueden contener otros objetos. Los objetos también engloban funciones diseñadas para actuar sobre los mismos objetos. Esas funciones se llaman métodos. Además el programador puede especificar cuál será el resultado de aplicar a un objeto un operador matemático como el + o el -, mediante un concepto llamado sobrecarga de operadores.

Una clase es una plantilla, no se reserva memoria para una clase, y sólo puede existir una instancia de cada clase en un mismo programa. Un objeto es una instancia de una clase, necesita reservar memoria y puede coexistir con otros objetos de la misma clase. En la práctica los términos objeto y clase se utilizan indistintamente, excepto cuando se quiere resaltar la diferencia.[8]

1.2.2 Propiedades de los objetos

Para que un lenguaje pueda llamarse orientado a objetos, debe proporcionar las siguientes características:

- **Abstracción de datos.**
- **Encapsulación y ocultación de datos.**
- **Herencia.**
- **Polimorfismo.**
- **Reutilización.**

1.2.2.1 Abstracción de datos

Se define como la capacidad para examinar algo sin preocuparse de sus detalles internos. Por ejemplo si se define un objeto que implementa un número complejo, es razonable que los usuarios de dicho objeto tengan la capacidad de sumar y restar complejos sin conocer los pormenores de las operaciones.

Los distintos niveles de abstracción nos permiten referirnos a grupos más o menos amplios de objetos, eludir detalles innecesarios de un objeto o tratar un objeto del que desconocemos algunas, o muchas, de sus características.[3]

1.2.2.2 Encapsulación y ocultación de datos

Los datos y las funciones deben estar encapsulados en una única entidad, un objeto. La encapsulación es la capacidad de ocultar algunas características de los objetos, a las que únicamente pueden acceder ellos mismos u otros objetos de la misma clase. Cada clase realiza su trabajo y guarda sus datos, sin que el resto del sistema sepa siquiera cuáles son esos datos. Como los objetos que no son de su misma clase no pueden acceder a esos datos, eliminamos el riesgo de que distintos módulos del sistema intenten modificar los datos.

1.2.2.3 Herencia

Un objeto definido similar a otro ya existente debe reutilizar las características del objeto padre. Por ejemplo, se podría decir que una televisión es un objeto con ciertas propiedades (datos) y controles (métodos).

Ejemplos de datos pueden ser la capacidad de cambiar de canal o modificar el volumen. Entonces se podría definir un nuevo objeto, un televisor en color, que tiene las características del televisor normal. Además de las características de una televisión, que no tendrán que redefinirse porque han sido heredadas, un televisor en color tendría métodos para ajustar el color y el tono. La relación entre objetos que comparten datos o métodos se expresa en una estructura en forma de árbol llamada jerarquía de herencia. El televisor en blanco y negro sería un objeto padre, mientras que el televisor en color sería un objeto hijo.

De manera similar, el televisor en blanco y negro sería un objeto hijo respecto del osciloscopio y la radio, como se muestra en la figura 1-2.

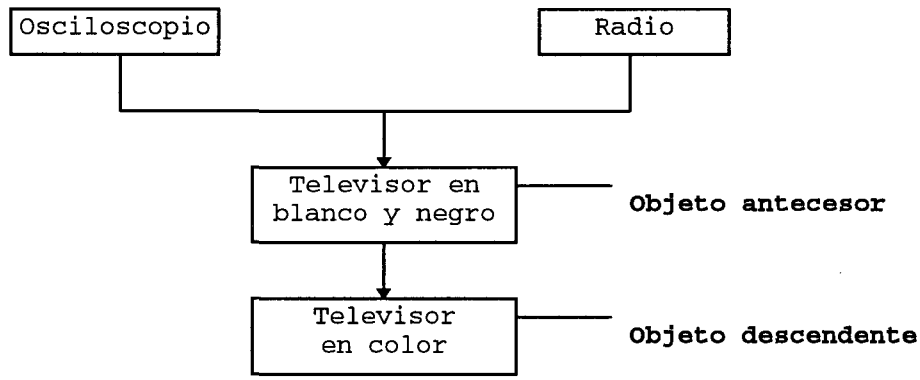


Figura 1-2

La herencia nos permite definir, en una única clase, los atributos y métodos comunes a todas las clases que dependen jerárquicamente de ella. Esto simplifica nuestro trabajo y aumenta la consistencia del sistema.[9]

1.2.2.4 Polimorfismo

En los lenguajes de programación orientados a objeto, las funciones deben formar parte de un objeto. Dichas funciones toman el nombre de métodos del objeto. El polimorfismo es la capacidad que tienen los objetos hijo de redefinir los métodos de sus objetos padre si lo necesitan. En otras palabras, objetos diferentes reaccionan de modo diferente al mismo mensaje. Por ejemplo, supongamos un determinado número de figuras geométricas, en las que todas ellas comprenden el mensaje dibujar. Cada objeto reacciona a este mensaje visualizándose a sí mismo en la pantalla de vídeo. Evidentemente, un objeto rectángulo se dibujará de modo diferente que un objeto círculo.

El polimorfismo juega un papel importante en la simplificación de la sintaxis al realizar la misma operación sobre una colección de objetos.[7]

1.2.2.5 Reutilización

Una vez que una clase se ha escrito, creado, depurado y comprobado, se puede distribuir a otros programadores para utilizarla en sus propios programas. Es decir, se puede utilizar como un bloque básico para construir programas. Esta operación se llama reutilización.

Los programas orientados a objeto se construyen a partir de componentes reutilizables de software. Esta operación es similar al modo en que una biblioteca de funciones en un lenguaje procedimental se puede incorporar en diferentes programas.

1.2.3 Aplicación de la abstracción

El término abstracción significa que un programador puede definir nuevos objetos que muestren un comportamiento similar al de otros objetos predefinidos. Por ejemplo, el C contiene una amplia gama de operadores propios que cubren desde la aritmética básica hasta la manipulación de bits. Un programador esperará utilizar esos operadores de manera que su utilización tenga sentido.

Cuando se define una nueva clase se está creando un nuevo tipo de datos. En C, los nuevos tipos de datos se derivan de los ya existentes, permitiendo que funcionen los operadores diseñados para el tipo base. Pero definir unos tipos basados en tipos predefinidos en C puede llevar a resultados no buscados.

En la programación orientada a objetos, sin embargo, se tiene completo control sobre el comportamiento de los objetos cuando se les aplica un operador. Cada objeto que se crea es un nuevo tipo de datos. Cualquier programador que utilice esos objetos puede esperar que los operadores estándar funcionen con ellos. Si se define un objeto Cadena que presente cadenas de caracteres, otros programadores podrían suponer que se pueden concatenar dos cadenas para formar una sola realizando operaciones parecidas a las que se muestran el la figura 1-3.

```
Cadena      Cadena1;
Cadena      Cadena2 = "Prueba.";
Cadena      Cadena3 = " de Cadenas.";

Cadena1 = Cadena2 + Cadena3;
```

Figura 1-3

Esta posibilidad se proporciona a través de la utilización de la sobrecarga de operadores. Cuando se desarrollan nuevas clases, se deben comprobar todos los operadores del C++ y decidir las características deseadas para el objeto cuando se le aplica cada uno de los operadores. Para los operadores que no estén explícitamente definidos en el código, el compilador trata de utilizar las conversiones implícitas de tipos de datos para determinar el comportamiento del operador.

1.2.4 Aplicación del encapsulamiento

Se puede utilizar el encapsulamiento para mezclar datos y funciones relacionados entre sí dentro de un mismo elemento, un objeto. Normalmente se utiliza el encapsulamiento de datos cuando se hace uso de una estructura (struct) o una unión (union) en C. Cuando se trabaja con programación orientada a objetos sencillamente se amplía la idea de estructura o unión, de forma que incluya las funciones que operan en los datos. Por ejemplo, supongamos que se define una estructura para almacenar la información de los empleados de una empresa de la forma que se muestra en la figura 1-4.

```
struct      Empleado
{
      char Nombre[61];
      char Telefono[9];
      int Sueldo;
};
```

Figura 1-4

La estructura definida encapsula los datos referentes a un empleado, pero necesita de funciones externas (que no forman parte de la estructura de datos con la que se está trabajando) para manipular los datos. Una definición orientada a objetos ampliaría el concepto de estructura para incluir las funciones necesarias para operar con los datos, como podemos apreciar en la figura 1-5.

```
class       Empleado
{
      // Elementos de datos.
      char Nombre[61];
      char Telefono[9];
      int Sueldo;
      //Métodos.
      void MarcarTelefono();
      void ImprimirChequeNomina(float Horastrabajadas);
};
```

Figura 1-5

Ahora esta clase contiene tanto datos como las funciones necesarias para operar con ellos. Esta clase permite mandar al ordenador que marque el teléfono o que imprima el cheque de la nómina de cualquier empleado. Obsérvese que las listas de parámetros de los métodos no incluyen ningún tipo de puntero a los datos que deben manipularse. Los métodos de un objeto tienen automáticamente acceso y capacidad de operar con los elementos de datos de dicho objeto.

Otra razón para utilizar el encapsulamiento es ocultar las dependencias del sistema, especialmente aquellas asociadas a los dispositivos del hardware.

1.2.5 Aplicación de la herencia

A menudo, cuando se definen clases, se descubre que varios objetos tienen especificaciones comunes referentes a los datos o a los métodos.

Por ejemplo supongamos que se definen dos clases que representen empleados y clientes, como se muestra a continuación en la figura 1-6.

```
class       Empleado
{
      // Elementos de datos.
      char Nombre[61];
      char Telefono[9];
      int Sueldo;
      //Métodos.
      void MarcarTelefono();
      void ImprimirChequeNomina(float Horastrabajadas);
};
class       Cliente
{
      // Elementos de datos.
      char Nombre[61];
      char Telefono[9];
      float Balance;
      //Métodos.
      void MarcarTelefono();
      void ImprimirFactura();
};
```

Figura 1-6

Ambos objetos tienen en común dos elementos de datos, *Nombre y Telefono*, y un método, *MarcarTelefono()*. Se puede simplificar la tarea de programación definiendo una clase antecesora llamada *Persona* que almacene los elementos comunes de ambas clases. Las clases *Empleado* y *Cliente* se definirían entonces como hijas de la clase *Persona*. La relación global se expresa mediante una jerarquía de herencias. La nueva implantación podría ser la presentada en la figura 1-7.

```
class       Persona
{
      // Elementos de datos.
      char Nombre[61];
      char Telefono[9];
      //Métodos.
      void MarcarTelefono();
};
class       Empleado : Persona
{
      // Elementos de datos.
      int Sueldo;
      //Métodos.
      void ImprimirChequeNomina(float Horastrabajadas);
};
class       Cliente : Persona
{
      // Elementos de datos.
      float Balance;
      //Métodos.
      void ImprimirFactura();
};
```

Figura 1-7

En este ejemplo, una declaración como:

class Cliente : Persona

Indica que la nueva clase Cliente hereda todos los elementos de datos y métodos de la clase antecesora *Persona*. La clase descendente se encuentra a la izquierda del signo de dos puntos (:), y la o las clases antecesoras, se encuentran a la derecha del mismo signo.

En este ejemplo, la herencia simplifica la implementación de los dos objetos permitiendo escribir una sola vez el método *MarcarTelefono()* para las dos clases *Cliente* y *Empleado*.

1.2.6. Aplicación del polimorfismo

Puede suceder que se quiera definir una clase muy parecida a otra existente, aunque no exactamente igual. En este caso, se puede hacer uso del polimorfismo para redefinir el modo en que trabajan uno o varios métodos ya existentes. El objeto hijo redefine un método de su objeto padre con definir otro nuevo con el mismo nombre de función que el anterior. El compilador selecciona la versión adecuada de la función cuando se la llama a partir del tipo del objeto.

El polimorfismo también es útil cuando se quiere modificar el comportamiento de un objeto muy utilizado en los proyectos de software propios. A veces el modificar un método del objeto, quizá para añadir una nueva característica, hace que el código anterior deje de funcionar. Para evitar este problema se puede definir una nueva clase, derivada del objeto original, que herede todas sus características.

Entonces se puede utilizar el polimorfismo para redefinir la función especificada que se quería modificar en la nueva versión. Por ejemplo, supongamos que se define una clase llamada *PruebaAnterior* con la definición de *Imprimeme() que podemos ver en la figura 1-8.*

```
void PruebaAnterior :: Imprimeme()
{
      printf("Prueba anterior\n");
}
```

Figura 1-8

Existe código que utiliza la clase *PruebaAnterior, como se muestra en la figura 1-9.*

```
PruebaAnterior    Ejemplo1;
PruebaAnterior    Ejemplo2;

Ejemplo1.Imprimeme();
Ejemplo2.Imprimeme();
```

Figura 1-9

Si modificamos la clase *PruebaAnterior* cambia el método *Imprimeme()*. Para evitar cambiar el código ya existente, se puede definir una nueva clase llamada *PruebaNueva*, que herede las características de la clase *PruebaAnterior* pero redefina la función *Imprimeme(). Lo podemos ver en la figura 1-10.*

```
class PruebaNueva : PruebaAnterior
{
public:
      virtual Imprimeme();
};

void PruebaNueva :: Imprimeme()
{
      printf("Prueba nueva\n");
}
```

Figura 1-10

El código que utilice objetos de la clase *PruebaNueva* tendrá el comportamiento nuevo, y el código que utilice la clase *PruebaAnterior* tendrá el comportamiento anterior, figura 1-11.

```
PruebaAnterior    Ejemplo1;
PruebaNueva       Ejemplo2;

Ejemplo1.Imprimeme(); // versión antigua
Ejemplo2.Imprimeme();//versión moderna
```

Figura 1-11

1.3 Introducción al lenguaje C++

Al diseñar C++ se planteó, como premisa inicial, la necesidad de compatibilidad ascendente entre C y C++. Todo lo que se puede hacer en C se puede hacer también en C++.[2],[3],[8]

Esta compatibilidad permitió que los programadores y las empresas que utilizaban C, tuvieran menos reparos a la hora de cambiar a C++, y de esta forma, conseguir una importante cantidad inicial de programadores.

C++ es un lenguaje de programación híbrido, puesto que se puede programar con orientación a objetos o con metodología tradicional. Para usar C++ como lenguaje orientado a objetos debemos basar la programación en un diseño orientado a objetos.

1.3.1 ¿ Por qué C++ ?

C++ es actualmente uno de los lenguajes de mayor difusión y con mejores expectativas de futuro. El modelo de orientación a objetos resulta una alternativa recomendable por sus características de:

- **Difusión:** Gran número de usuarios. Sobre él hay libros revistas, congresos, cursos, etc.

- **Versatilidad:** No está enfocado a un tipo de problema en concreto.

- **Disponibilidad:** Sus especificaciones son públicas y cualquier empresa puede desarrollar un compilador C++.

- **Portabilidad:** El código fuente puede compilarse en distintos ordenadores y entornos.

- **Eficiencia:** Gran velocidad de ejecución heredada de su predecesor C.

- **Herramientas:** Existe un número creciente de entornos de desarrollo, depuradores, librerías de clases.

1.3.2 Semejanzas entre el C y el C++

- La semejanza más significativa es que el C++ contiene al C como un subconjunto. Esto significa, que se pueden escribir programas en ANSI C y utilizar el compilador de C++, compilar el programa y ejecutarlo sin tener ningún conocimiento de las extensiones de C++

- Todo el control lógico del programa es idéntico en ambos lenguajes. Por ejemplo, la sintaxis de los bucles WHILE y FOR, las comprobaciones IF y las instrucciones SWITCH no sufren ningún tipo de cambio.

- Los operadores previamente definidos, incluyendo todas las propiedades para la manipulación de bits, no sufren ninguna modificación.

- El concepto de función y su procedimiento de llamada, básicamente no están modificados. El único cambio que puede apreciarse es que la mayoría de las funciones del C++ están definidas como parte de un objeto específico sin estar disponibles para que cualquiera pueda utilizarlas.

- Se puede trabajar sin ninguna modificación con las rutinas de librería en tiempo de ejecución, de esta forma, funciones tan familiares como printf() y strcpy() se pueden seguir utilizando.

- Todos los tipos específicos de datos en C, incluyendo las estructuras y la uniones, siguen estando disponibles y no sufren ninguna modificación en su significado.

1.3.3 Características introducidas por el C++

Seguridad de tipos de enlace

Los tipos de los parámetros se comprueban en tiempo de enlace detectando errores de tipo durante su desarrollo.

Operador de resolución de ámbito

El operador de resolución de ámbito puede utilizarse para las variables por referencia que de otro modo estarían fuera del ámbito del bloque del programa actual.

Operadores New y Delete

Los operadores new() y delete() reemplazan a los operadores malloc() y free(). Soportan la asignación de memoria para los tipos definidos por el usuario sin necesitar que el programador controle el tamaño de los objetos.

Librería de flujos de E/S

La librería de flujos de E/S soporta la entrada y salida de los objetos.

Prototipos de funciones

Los prototipos de funciones permiten al compilador y al enlazador, realizar una mejor comprobación de los tipos durante el tiempo de compilación.

Conversiones de tipos definidos por el usuario

Las conversiones de tipos definidos por el usuario nos permiten especificar funciones que permitan al compilador realizar conversiones de tipo en sus objetos.

Sobrecarga de funciones

La sobrecarga de funciones permite utilizar el mismo nombre para diferentes funciones que realizan la misma operación lógica sobre diferentes datos.

Clases de almacenamiento volátil y constante

Los datos pueden ser declarados como volátiles o constantes. Los datos volátiles indican al compilador que el valor puede cambiar sin el conocimiento del compilador. Los datos constantes indican que su valor no puede cambiar.

Parámetros por defecto

Se pueden especificar valores por defecto para los parámetros de una aplicación.

Declaraciones dentro de bloques

Se pueden declarar variables allí donde se utilicen y no únicamente al comienzo de las funciones. Las declaraciones tienen un alcance que abarca al bloque donde están definidas.

Variables por referencia

Es más fácil pasar y acceder a los parámetro por referencia.

Sobrecarga de operadores

Los operadores estándar del C pueden trabajar con tipos definidos por el usuario.

Funciones en línea

Las funciones que pueden ser expandidas en línea, evitan la sobrecarga de llamadas a funciones. Al contrario que en las macros, la comprobación de los tipos en las funciones en línea se realiza sobre las funciones, y los parámetros se pasan como en la llamada a una función normal.

Comentarios de línea

El conjunto de caracteres escritos después de //, especifica que se trata de un comentario.

Clases

Las clases se emplean para definir objetos, agrupando juntos a funciones y datos.

Control de acceso

El control de acceso permite hacer que los campos y los métodos de una clase sean visibles en otras clases, visibles solamente en clases derivadas, o completamente privados.

Amigos

Los amigos permiten a una clase poder seleccionar otros objetos o funciones para acceder a campos privados.

Funciones virtuales

Las funciones virtuales permiten el ligamento dinámico de funciones.

Constructores

Los constructores permiten decidir lo que ocurrirá cuando un objeto sea creado.

Destructores

Los destructores permiten decidir lo que ocurrirá cuando un objeto sea destruido.

1.4 El compilador de Borland C++

Con la introducción del compilador de C++ de Borland, tenemos la oportunidad de explorar simultáneamente los mundos de Windows y del C++. Este compilador de Borland permite a los programadores para Windows utilizar el C++ para desarrollar sus programas, y permite a los programadores en C++ diseñar aplicaciones para Windows.[8]

1.4.1 ¿ Por qué Borland C++ ?

Borland C++ aumenta la productividad gracias al sofisticado Entorno Integrado de Desarrollo (IDE) de Borland.

Sus características más importantes son:

- El compilador soporta tanto el lenguaje C como el C++, de acuerdo con la versión 2.0 de la especificación del C++ de AT&T.

- Borland nos provee de todas las herramientas necesarias para producir programas útiles para Windows.

- El compilador se ejecuta en modo protegido, por lo que los programas o librerías grandes no necesitan muchos intercambios.

- Se soportan ficheros de cabecera precompilados. Debido a que las cabeceras para Windows tienden a ser bastante grandes, esta característica de Borland C++ puede mejorar los tiempos de compilación en un factor de diez.

- Se incluye un editor de varios ficheros, arbitrariamente largos, basados en Windows

- El compilador facilita acceso inmediato a una ayuda interactiva que abarca el lenguaje C y su librería en tiempo de ejecución, el lenguaje C++ y las llamadas al API de Windows.

1.4.2 Contenido del Borland C++ 3.x

El Borland C++ 3.x contiene:

- Un compilador optimizador global de ANSI C y C++ versión 2.1.
- Un compilador con interfaz Dos modo protegido y un entorno de programación.
- Un entorno integrado de desarrollo basado en Windows (IDE).
- Un visualizador fuente de gráficos (ObjectBrowser).
- Una utilidad para el seguimiento de mensajes de Windows (WinSight).
- Un depurador de aplicaciones del DOS y de Windows (Turbo Debugger) para encontrar y depurar programas.
- Un perfilador para DOS y Windows (Turbo Profiler), que permite seguir el flujo de mensajes entre ventanas.
- El Turbo Assembler, un compilador para lenguaje ensamblador.
- El Resource Workshop, o taller de recursos, que permite construir y modificar recursos de programas, como iconos, menús, o cuadros de diálogo.

- Una librería de clases de C++ para simplificar el desarrollo de aplicaciones para Windows (ObjectWindows).

- La librería EasyWin para portar programas del DOS al Windows.

- Utilidades para aplicaciones del DOS (Turbo Vision).

- Código fuente de las librerías en tiempo de ejecución.

Para instalar todas las opciones disponibles del compilador de Borland C++ se necesitan aproximadamente 40 Mb de espacio en disco duro. Una configuración normal se compone de un ordenador con procesador 30386, 4 Mb de memoria, un disco duro de 80 Mb, un ratón y una tarjeta gráfica y monitor VGA.

1.4.3 La librería ObjectWindows

La librería ObjectWindows nos proporciona unas poderosas herramientas que nos asistirán en nuestra programación de aplicaciones Windows. Sin una librería semejante, la codificación de aplicaciones Windows se volvería más dificultosa y precisaría de un incrementado proceso de programación. El éxito de la librería ObjectWindows se basa en la realización de la programación orientada a objetos, combinándola con los procesos de programación basada en eventos (base de la filosofía de la programación Windows), y ejemplifica lo bien que estas dos metodologías pueden coexistir.[9]

Todas las aplicaciones para Windows basadas en la librería ObjectWindows de Borland tienen tres componentes básicos: una clase para representar a la aplicación en ejecución, una clase para representar a la ventana principal de la aplicación y un bucle de programación WinMain().

1.4.3.1 La clase de la aplicación

La clase de la aplicación en ejecución se deriva de la clase *TApplication* de la librería ObjectWindows.

Su constructor llama al constructor de la clase *TApplication* pasándole los parámetros adecuados.

Como mínimo esta clase crea una instancia de la clase que representa la ventana principal de la aplicación.

1.4.3.2 La clase de la ventana principal

La clase que representa la ventana de la aplicación se deriva de la clase *TWindow* de la librería ObjectWindows. Esta es la clase en la cuál se selecciona el título de la ventana y su menú, se responde a las opciones elegidas en dicho menú y generalmente se muestra la salida del programa.

1.4.3.3 El bucle principal del programa

La función WinMain() equivalente a la función main() de los programas en C, se utiliza para inicializar todos los objetos creando una instancia de la clase de la aplicación y arrancando su ejecución.

1.4.4 Programación con la librería ObjectWindows

ObjectWindows es una librería de clases del C++ desarrollada por Borland y distribuida con el compilador de Borland C++. Las clases engloban los elementos clave del API de Windows, como ventanas, cuadros de diálogo y diversos tipos de controles. La librería ObjectWindows ofrece tres ventajas a la hora de programar aplicaciones para Windows:

- Encapsulamiento de la información de las ventanas.
- Abstracción de las funciones más importantes del API de Windows.
- Respuesta automática a los mensajes.

1.4.4.1 Encapsulamiento de la información de una ventana

La librería ObjectWindows engloba varios de los campos de Windows directamente en el objeto ObjectWindows. Por ejemplo, muchas de las funciones del API de Windows necesitan un parámetro que es un manejador para una ventana, y esas funciones son llamadas generalmente desde los métodos del propio objeto ventana. Permitiendo a la ventana almacenar su propio manejador, puede pasarlo automáticamente sin necesidad de especificarlo en todas las ocasiones. Otros ejemplos de encapsulamiento de datos con ObjectWindows son los identificadores de las ventanas descendientes, los atributos de las ventanas descendientes, los atributos de las ventanas y sus títulos.

1.4.4.2 Abstracción de las funciones del API de Windows

Microsoft ha definido casi 600 funciones del API que configuran el interfaz de Windows. Aunque se puede llamar a cualquier función del API directamente desde el Borland C++, la librería ObjectWindows permite utilizar métodos de objetos para simplificar muchas de estas llamadas a funciones. Las clases de ObjectWindows agrupan varias llamadas a funciones relacionadas del API en un sólo método para realizar tareas de más alto nivel, y utilizar datos almacenados internamente para proporcionar automáticamente varios parámetros.

1.4.4.3 Respuesta automática a los mensajes

Windows es un entorno basado en mensajes. Tras la creación de una ventana, todo el procesamiento dentro de ella se realiza respondiendo a mensajes recibidos como parámetros por una función de alto nivel llamada el procedimiento de la ventana, Dichos mensajes se denominan mensajes de Windows. Las ventanas descendientes, incluidos los botones, cuadros de texto, casillas de verificación y cuadros de lista, se controlan enviando mensajes a ellas y recibiendo mensajes de ellas. En la programación tradicional en Windows, dichos mensajes se procesaban en una larga sentencia *CASE*. En algunas ventanas de alto nivel, el procedimiento de la ventana puede llegar a tener una sentencia *CASE* con tres niveles de anidamiento y cerca de 200 opciones. Esto hace difícil mantener la longitud de dicha función por debajo del máximo de dos páginas recomendado por la mayoría de los expertos en garantía de calidad.

Con la librería ObjectWindows, sencillamente se define un método para cada mensaje que se quiere gestionar. El objeto llamará automáticamente al método que corresponda a cada mensaje cuando lo reciba.

Para los mensajes de Windows que no tengan asignado un método definido por el usuario, el objeto ObjectWindows gestiona automáticamente el mensaje utilizando cierto tipo de procesamiento por defecto.

2

LA TELEFONÍA MÓVIL AUTOMÁTICA (TMA)

La telefonía móvil consiste en ofrecer un acceso vía radio a un abonado de telefonía, de tal forma que pueda realizar y recibir llamadas dentro del área de cobertura del sistema. Estos abonados pueden acceder al sistema desde terminales instalados en sus vehículos o desde teléfonos portátiles de mano, aumentando las posibilidades de acceso al sistema.

Se basa en dividir la zona de cobertura pretendida, en zonas más pequeñas denominadas células, que serán de igual forma y tamaño. Limitando convenientemente la potencia con que se emite cada frecuencia, permite la reutilización de las mismas a distancias bastante cortas, aumentando la capacidad de los sistemas.

Un sistema de TMA está formado por una superestructura de células, constituida por grupos con encaje mutuo perfecto. Dentro de cada grupo no se repetirá frecuencia alguna y la asignación de frecuencias será la misma para todos ellos.

2.1 Evolución histórica

El primer sistema que apareció fue el NMT-450. Funciona en la banda de 450 MHz y fue realizado por las administraciones telefónicas nórdicas.

Posteriormente aparecieron en los Estados Unidos de América y el Reino Unido las normas TACS y AMPS, muy similares entre ellas y que funcionaban en la banda de 900 MHz. También surgió la versión del sistema nórdico en la banda de los 900 MHz, el NMT-900.

Todos estos sistemas tienen en común que utilizan un interfaz radio analógico, y que a pesar de seguir unas especificaciones más o menos públicas, no es posible interconectar sistemas de distintos suministradores y como consecuencia no existía la posibilidad del seguimiento internacional (posibilidad de llamar o ser llamado cuando el abonado se encuentra en otro país).

Con todos estos problemas a la vista, junto con la posibilidad de que los sistemas existentes agotaran su capacidad en poco tiempo, en 1982 se creó un nuevo grupo de estandarización, cuya tarea sería la especificación de un sistema europeo único de radiocomunicaciones en la banda de los 900 MHz. Tres fueron los objetivos fundamentales que empujaron a emprender esta tarea:

- Disponer de un sistema de comunicaciones móviles común a toda Europa. De esta forma se permite que un usuario pueda tener acceso al servicio en todos los países de la comunidad. Además, se consigue un mercado mucho más amplio que llevará a una disminución en los costes de fabricación y a un menor precio de los equipos y servicios.

- Superar la capacidad de los sistemas móviles precedentes. La demanda de telefonía móvil fue muy infravalorada durante la especificación de los sistemas de primera generación y resultó estar por encima de la capacidad de estos sistemas.

- Aprovechar el estado del arte de la electrónica digital (cada día se dispone de mayores potencias de proceso con costes cada vez menores).

Aproximadamente después de una década de trabajo, en 1991 se pudo hacer la primera demostración pública del sistema desarrollado. Se trataba de un sistema de radiocomunicaciones digitales celular denominado GSM (Global System for Mobile Comunications).

En paralelo con el GSM, se han definido otros dos sistemas de telefonía móvil digital, uno en USA, el ADC, y otro en Japón, el JDC. Sin embargo el GSM está más avanzado que los otros y se está imponiendo en otros países importantes.

Por último destacaremos la continuidad tecnológica que ETSI (Instituto Europeo de Estándares de Telecomunicación) ha dado al GSM, especificando el DCS 1800 , que no es otra cosa que el GSM en la banda de 1800 MHz. Se trata de un sistema celular que utiliza células de pequeño tamaño consiguiendo altas capacidades de usuarios en entornos urbanos.[11]

2.2 El GSM

Los sistemas digitales en general y el GSM en particular se contemplan como una solución al problema de la capacidad de los sistemas analógicos. Esto se consigue con una mejor planificación celular con mayor reutilización de las frecuencias. También tiene como objetivo prioritario el poder ofrecer seguimiento internacional.[10],[11]

Incorpora facilidades específicas para transmisión de datos que reducen algunas dificultades inherentes a los sistemas analógicos, permitiendo el establecimiento de circuitos de hasta 9600 bits/seg. Adicionalmente el sistema incorpora funciones de interconexión con la red telefónica conmutada y con la red pública de conmutación de paquetes (Iberpac en España).

La arquitectura de la red GSM está básicamente dividida en tres partes: el sistema de conmutación, el sistema de estaciones base y el sistema de operación y mantenimiento.

Cada uno de estos sistemas contiene una serie de unidades funcionales en las que se realizan todas las funciones que el sistema GSM es capaz de proporcionar. Las funciones relacionadas con el proceso de llamadas y abonados están implementadas en el sistema de conmutación, mientras que las funciones relacionadas con la radio se concentran en el sistema de estaciones base, todo ello está supervisado por el sistema de operación y mantenimiento.

Al sistema de estaciones base irá conectada la estación móvil vía una interfaz aérea y, a través de esta estación, el abonado de la red móvil será capaz de efectuar y recibir llamadas.

Para la gestión de llamadas hacia o desde los abonados de la red fija, es necesario que el sistema de conmutación tenga implementadas las interfaces apropiadas de interconexión con toda la variedad de redes fijas existentes: red telefónica conmutada, red digital de servicios integrados, red de paquetes, etc.

Para la gestión de llamadas hacia o desde otros abonados móviles es necesario que el sistema de conmutación tenga implementada la interfaz hacia otras entidades de la red GSM.

Dada la complejidad del sistema, la señalización del mismo es digital. Además, el elevado número de usuarios produce un tráfico de señalización entre terminales e infraestructura que aconseja el uso de varios canales dedicados a tal propósito.

El sistema de acceso utilizado es el TDMA (acceso múltiple por división en el tiempo), con una trama TDMA por cada portadora de radio. Cada trama consta de 8 intervalos de tiempo y cada uno de ellos se conoce con el nombre de canal físico, por el que se transmite una ráfaga de información.

Las principales ventajas que presenta el GSM sobre los sistemas analógicos son:

- Disminución de costes de fabricación por la escala del mercado

- Integración de voz y datos gracias a la digitalización de las transmisiones de radio.

- Acceso por tarjeta inteligente. Permite la utilización de cualquier terminal por parte de cualquier usuario, introduciendo el concepto de movilidad personal.

- Compatibilidad con la RDSI (Red Digital de Servicios Integrados).

- Menor consumo del terminal. Permite utilizar terminales más ligeros.

- La calidad de la voz es superior, sobre todo en zonas donde la señal de radio es débil o sufre importantes interferencias.

- Los servicios de datos son una parte integral del sistema dado el carácter digital del radioenlace.

- Se garantiza el control de acceso a los servicios mediante el uso también de técnicas criptográficas. Permitiendo independizar el terminal y la subscripción del usuario gracias a la utilización de un módulo de personalización (Suscriber Identity Module, SIM).

- Autentificación del usuario para prevenir el acceso de usuarios no registrados.

- Cifrado del radioenlance de toda la información de usuario, y de algunos elementos de señalización, para prevenir escuchas por parte de un tercero.

- Protección de la identidad del usuario para imposibilitar el seguimiento de su localización por parte de terceros.

2.3 Conceptos básicos sobre telefonía móvil

Antes de conocer la arquitectura de la red GSM, tenemos que saber diferenciar las siguientes partes comunes a todos los sistemas de telefonía móvil celular:

- **Las estaciones fijas.**
- **Las estaciones móviles.**
- **El registro de localización.**
- **El Handover.**
- **El área de estaciones base.**
- **El área de localización.**
- **El área de conmutación.**
- **El área de servicio.**
- **El área del sistema.**

2.3.1 Las estaciones fijas

Son estaciones radioeléctricas no previstas para su utilización en movimiento. Existen diversos tipos de estaciones fijas:

- **Estación base.**
- **Estación de control.**
- **Estación Repetidora.**

2.3.1.1 Estación base

Una estación base es una estación fija explotada directamente desde una unidad de control situada en un punto de control específico. Dicho control puede ser local o remoto, mediante líneas físicas o radioenlaces. La estación base es la que cumple el objetivo final de dar cobertura radioeléctrica a una zona determinada, controlando la antena para la emisión y recepción.

2.3.1.2 Estación de control

Una estación de control es una estación fija cuyas transmisiones se emplean para controlar automáticamente las transmisiones o el funcionamiento de otra estación de radio en un emplazamiento específico. Las estaciones de control se emplean generalmente para controlar una estación base o repetidora.

2.3.1.3 Estación repetidora

Son estaciones fijas que retransmiten las señales recibidas y que por sus características técnicas y su situación estratégica permiten el logro de una cobertura determinada del sistema. Suelen funcionar de forma desatendida, telecontrolándose en ocasiones desde otro punto fijo. Es importante que las estaciones repetidoras sólo actúen con las señales deseadas y no con otras interferentes. Para ello, el repetidor dispone de un circuito "silenciador" que impide la retransmisión de la señal recibida a menos que esta venga acompañada de una señal "llave" que abra el transmisor. Esta señal se transmite junto con la señal deseada en forma de un tono no audible.

2.3.2 Las estaciones móviles

Una estación móvil es una estación radioeléctrica del servicio móvil prevista para su utilización en un vehículo en movimiento, o que efectúa paradas en puntos indeterminados. También es aplicable el concepto a aquellos equipos portátiles de cierta autonomía que no van instalados en un vehículo o que pueden hacerlo de forma temporal.

El acceso a la red telefónica debe ser igual que en un teléfono normal que accede a través del teléfono a dos hilos. Para ello en el diseño del sistema hay que tener en cuenta que el terminal móvil se puede encontrar en situaciones de difícil comunicación con la estación base. De esta forma el usuario tiene una libertad total de movimiento dentro de la zona de cobertura, que le permite tener un acceso libre al sistema telefónico.

2.3.3 El registro de localización

Dentro del sistema se reservan un número de canales radio como canales de señalización. Adicionalmente la red se divide en un número de áreas de tráfico, cada área consiste en un grupo de células.

La estación base genera un código de identificación correspondiente con el área de tráfico a la que pertenece, como parte de la información transmitida por los canales de señalización.

El abonado móvil que viaja a lo largo de la red, monitoriza el canal de señalización cuya emisión es más potente. Como el móvil se desplaza de una célula a la siguiente, detecta un deterioro en la calidad de recepción en el canal común de señalización utilizado y en consecuencia comienza la búsqueda de otro canal con señal más potente.

Una vez que el móvil ha sintonizado la nueva señal, hay dos opciones posibles. La primera es que habiendo cambiado de célula no se abandone el área de tráfico por lo que el registro no se modificará. La segunda opción es que el móvil también haya cambiado de área de tráfico, en este caso el móvil transmite su identificación a la nueva estación base, que transfiere la información al centro de conmutación. De esta forma el móvil ha registrado su ubicación de manera que la red es capaz de encaminar la llamada hacia el móvil de modo rápido y eficiente.

2.3.4 El handover

Con este término se denomina al problema que se plantea cuando a lo largo de una comunicación, el terminal móvil se desplaza de la cobertura de una célula a otra continua.

Cuando el portátil está desocupado, explora periódicamente los intervalos de tiempo disponibles por las estaciones base, y se engancha en la estación base de la que recibe mayor intensidad de campo.

Si hay una llamada para un portátil, se envía un mensaje de búsqueda, que contiene la identidad de ese portátil, en el canal de señalización de todos los intervalos de tiempo activos en ese momento en el sistema. Este mensaje es leído por el portátil, que si estaba enganchado en un canal libre, enviará una petición de llamada en el intervalo correspondiente en la segunda mitad de la trama TDMA (Acceso Múltiple por División de Tiempo), y si estaba enganchado en un canal de tráfico, tomará el intervalo de tiempo con mejor calidad de la primera mitad de la trama TDMA y enviará la petición de llamada en el intervalo de tiempo correspondiente en su segunda mitad. En ambos casos se indica la identidad de la estación base a la que está enganchado.

Una vez que la estación base y el portátil han acordado qué intercalo de tiempo van a utilizar, la unidad central asignará un canal de voz a la llamada y enviará los dígitos marcados a la interfaz con la red telefónica conmutada.[11],[13]

2.3.5 El área de estaciones base

Es la parte de la red cubierta por una estación base dentro de la cual la posibilidad de efectuar una comunicación estará en función de la calidad del diseño del sistema. Cada estación móvil en el área de una estación base puede conectarse por radio a la misma. A su vez, varias estaciones base estarán bajo el control de una estación de control.

2.3.6 El área de localización

Se define como un área en la cuál una estación móvil puede moverse libremente sin alterar los datos del registro de localización. Puede comprender varias estaciones base y está compuesta por aquellos lugares susceptibles de ser atendidos por el sistema de TMA. En estos lugares el tráfico cursado justifica la cobertura radioeléctrica del mismo. Los ejemplos típicos son ciudades, autopistas principales, aeropuertos, polígonos industriales, etc.

2.3.7 El área de conmutación

Consiste en la parte de la red cubierta por una estación de control. Puede comprender varias áreas de localización y se encarga de controlar las comunicaciones cursadas a través de las estaciones base de una determinada zona. El control del registro y handover debe hacerlo la estación de control correspondiente.

2.3.8 El área de servicio

Área en la cuál una estación móvil puede ser alcanzada por un abonado de la red telefónica fija, sin que tenga que conocer la situación actual del móvil dentro del área. El sistema de registro de localización asociado con cada área de servicio debe contener entonces una lista de todas las estaciones móviles localizadas dentro de la misma.

2.3.9 El área del Sistema

El área del sistema se define como un conjunto de áreas de servicio accesibles para una estación móvil siempre dentro de la compatibilidad completa de diversos sistemas móviles.

2.4 Arquitectura de la red GSM

Como en toda red de comunicaciones móviles celulares, GSM está dividida fundamentalmente en dos partes. Por un lado, la red de acceso conecta a los terminales móviles con la red mediante radioenlaces. Por otro lado, la red fija interconecta el sistema con la red telefónica pública mediante las funciones de conmutación y de gestión de movilidad en el radioacceso. Esta estructura se ve reflejada en la figura 2.1: el subsistema de acceso está formado por las estaciones base (BTS, Base Transceiver Station) y por los controladores de estaciones base (BSC, Base Station Controller); los restantes nodos de la figura forman la red fija. A continuación de la figura 2-1 se explicará el funcionamiento de cada uno de los nodos que conforman la arquitectura GSM.[16],[17]

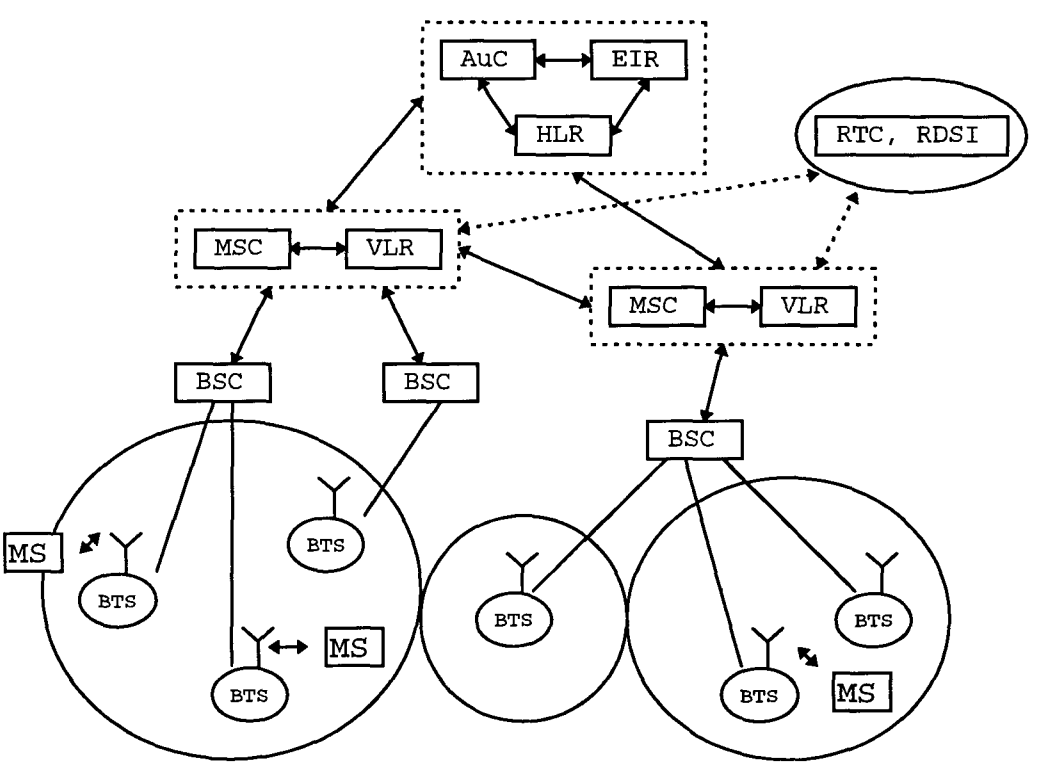


Figura 2-1

2.4.1 Unidades funcionales de la red GSM

En este apartado se resumen las funciones de los nodos del sistema GSM. Siendo dichos nodos:

- MSC: Central de conmutación de servicios móviles.
- HLR: Registro de posición base.
- VLR: Registro de posición visitado.
- AuC: Centro de autentificación.
- EIR: Registro de estaciones móviles.
- BSC: Controlador de estaciones base.
- BTS: Estación base.
- MS: Estación móvil.

2.4.1.1 MSC: Central de Conmutación de Servicios Móviles

El MSC (Mobile Switching Center) o estación de control, constituye el interfaz entre la red GSM y la red telefónica pública conmutada. Sus funciones son básicamente: establecimiento, enrutamiento, control y terminación de las llamadas telefónicas; la asignación de frecuencias, la gestión del handover entre diferentes estaciones base y la tarificación.

Con objeto de obtener una cobertura adecuada en un área geográfica adecuada, se precisan un cierto número de estaciones base (transmisores/receptores de radio). Cada MSC puede disponer interfaces con varias estaciones base; al objeto de cubrir un país pueden precisarse varios MSC.

2.4.1.2 HLR: Registro de posición base

EL HLR (Home Location Register) es una base de datos donde se almacena: información relativa a las suscripciones y a la ubicación actual de los móviles para enrutamiento de llamadas entrantes; número identificador internacional de la estación móvil; número identificador de abonado; etc.

2.4.1.3 VLR: Registro de posición visitado

El VLR (Visited Location Register) se integra habitualmente con el MSC. Es una base de datos relativa a los usuarios que actualmente se encuentran en el área de cobertura del VLR en cuestión. Cuando un abonado entra en el área de un VLR nuevo, éste pide los datos necesarios al HLR asociado a dicho abonado.

2.4.1.4 AuC: Centro de autentificación

El AuC (Authentication Center) se encarga de la prevención de accesos no autorizados. Esto se realiza en dos pasos. Por un lado, el usuario debe identificarse frente al SIM (Suscriber Identity Module) tecleando su número de identificación personal. Por otro lado , la red comprueba la validez del SIM mediante un protocolo de autentificación.

2.4.1.5 EIR: Registro de estaciones móviles

El EIR (Equipment Identity Register) es una base de datos donde se almacena los identificadores internacionales de los equipos móviles clasificados en tres listas(blanca, gris o negra).

2.4.1.6 BSC: Controlador de estaciones base

El BSC (Base Station Controller) hace de interface entre la red fija y las estaciones base, separando las funciones radio de las de conmutación.

2.4.1.7 BTS: Estación base

La BTS (Base Transceiver Station) se encarga de las transmisiones radio en una o varias células. Entre otras cabe citar la función de cifrado/descifrado del radioenlace.

2.4.1.8 MS: Estación móvil

La MS (Mobile Station) es el usuario del sistema GSM, que debe encontrarse dentro de la cobertura de la red para poder comunicarse con otros usuarios de la red telefónica.

2.5 Geometría celular

Para simplificar el estudio, se suponen todos los transmisores de idénticos parámetros, altura de la antena, terreno homogéneo con las mismas condiciones de propagación en toda la zona de cobertura. Esto conduce a esquemas regulares de disposición de canales, basados en celdas homogéneas del mismo tamaño y forma.[10],[12]

Si en cada celda se utilizan antenas omnidireccionales, la zona de cobertura sería aproximadamente circular. Sin embargo, las coberturas circulares o no recubren el plano o producen solapes, esto último implica una reducción de la eficacia espectral, porque en un mismo punto se emplean dos frecuencias.

Se exigen entonces formas poligonales que recubren el plano sin solape. Hay tres formas posibles: triangular, cuadrada y hexagonal, como se muestran en la figura 2-2:

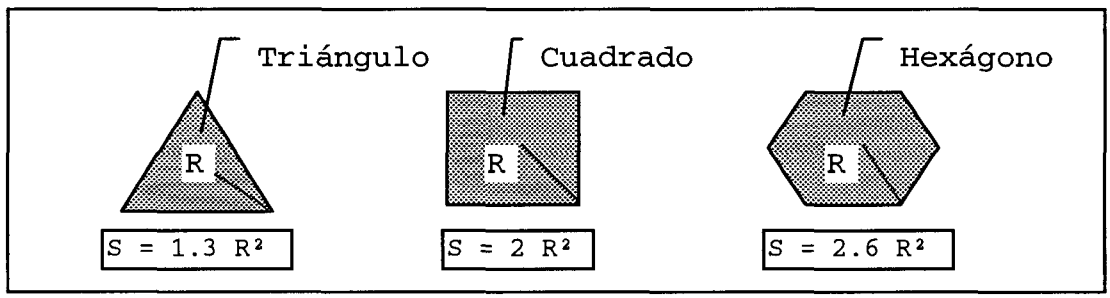


Figura 2-2

Se obtiene que el hexágono proporciona la mayor superficie de celda y por lo tanto el mínimo número de celdas necesario para la cobertura de una zona.

2.6 Estructura de las células

El número de células de un grupo se determina de manera que pueda repetirse de forma no interrumpida en el área de cobertura. Solamente algunas configuraciones lo permiten. Los grupos típicos se basan en 3, 4, 7, 12, o 21 células. En la figura 2-3 podemos observar un grupo de tres células y otro de cuatro.[11],[12],[13]

El número de células en cada grupo tiene significativa importancia en la capacidad total del sistema. Cuanto menor es el número de las células mayor es el número de canales por célula y en consecuencia el tráfico es más alto. Sin embargo debe buscarse el punto de equilibrio, porque si se utilizan más canales por célula y el tamaño del grupo es menor (menos células), la distancia entre las células que utilizan los mismos canales es menor, con la consecuencia de que la interferencia entre grupos adyacentes aumenta (interferencia cocanal).

El tráfico de un área en particular puede ser aumentado si se reduce el tamaño de la célula de manera que aumenta el número total de radiocanales disponibles en el área.

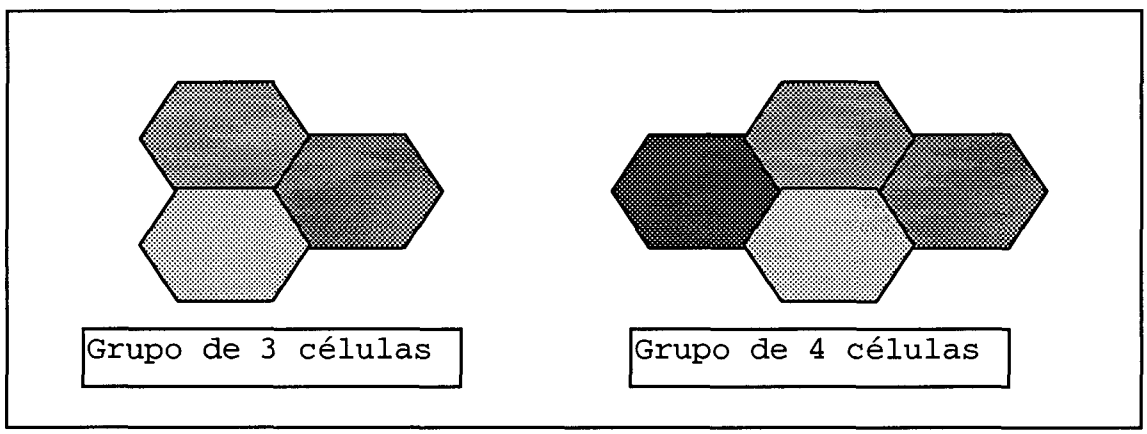


Figura 2-3

2.7 Gestión de las llamadas telefónicas

Las estaciones base ubicadas en el centro de cada célula están unidas a una central de conmutación que en esencia es una central de conmutación modificada para el sistema celular. Una red celular consistirá en la práctica en varias centrales de conmutación interconectadas entre sí. Esta configuración permite la realización completa de los distintos tipos de llamada, como son llamadas de móvil a fijo, de fijo a móvil y de móvil a móvil.[14]

2.7.1 Llamada originada por el móvil

Al "descolgar" el aparato, el terminal manda una señal que debe recoger la estación base en cuya área de cobertura se encuentre el móvil. Esta señal se denomina "tono de señalización".

Cuando la estación base correspondiente detecta el tono de señalización, se comunica a través de un canal de datos con el MSC, que se encargará de analizar las condiciones de la comunicación, es decir, intentará localizar un canal libre, y si hay varios, asignará el que esté en mejores condiciones para la transmisión.

A continuación el MSC comunica a la estación base el canal asignado para la comunicación, ésta a su vez, se lo indica a la estación móvil, quedando de esta forma ajustados los moduladores.

Por otro lado, el MSC transmite a la red conmutada mediante la trama correspondiente la intención de establecer la transmisión desde el móvil.

Si la red y el receptor están preparados, entonces se establece el enlace entre el móvil y el receptor hasta que se indique la petición de liberación de la llamada.

2.7.2 Llamada dirigida a un móvil

Cuando se marca el número correspondiente a un móvil, la red encamina esa petición hacia el MSC en el que se espera encontrar al móvil. En el caso de no encontrarlo, se rastrea por los demás MSC siguiendo un algoritmo adecuado.

Una vez localizada la central, esta buscará entre sus estaciones base para determinar en cuál de ellas se encuentra la estación móvil a la que va dirigida la llamada. Dicha estación base comunica al terminal móvil que se quiere establecer una comunicación.

Si el terminal no está ocupado en ese momento, responde afirmativamente a la estación base, que a su vez se lo comunica a la central (MSC).

En este momento, el MSC puede asignar un canal según el mismo criterio que el caso anterior. Se lo comunica a la estación base, que a su vez se lo hace llegar al móvil.

Con el dato del canal a emplear, los moduladores se ajustan a la frecuencia y en ese momento el terminal avisa al usuario de la petición de comunicación desde el móvil.

2.7.3 Liberación de la llamada

La liberación de la llamada se puede producir tanto desde el terminal fijo, como desde el móvil. En el primer caso se realiza de la siguiente forma:

- El terminal fijo finaliza la comunicación y es detectado por el MSC.
- El MSC comunica a la estación base que la comunicación ha finalizado.
- Una vez informada la estación base, finaliza la transmisión, dejando libre el canal.

La liberación a partir del móvil es como sigue:

- El móvil finaliza la comunicación, dejando de transmitir.
- El MSC detecta la liberación y se lo comunica a la estación base.
- La estación base recibe el mensaje de la central, y deja de transmitir, dejando libre el canal.

2.8 Evolución del Sistema

El método clásico de establecimiento de un sistema celular, es a partir de un primer esquema con pocas células de gran tamaño. La expansión se efectúa mediante subdivisión de las células, lo que provoca un exceso de canales y equipo inicialmente.[10],[15]

Por este motivo la transición debe ser gradual, aplicándose el concepto de recubrimiento, que consiste en ir añadiendo celdas dentro de la zona de servicio requerida. Estas celdas de solape pueden introducirse una a una cuando se requiera y la densidad de usuarios lo haga rentable.

La densidad de tráfico no es homogénea, por lo que la subdivisión celular tampoco lo será. En el caso de una amplia cobertura territorial que comprenda un núcleo urbano, en el centro de éste la densidad de tráfico será máxima e irá decreciendo hacia las afueras.

Un atributo importante de los sistemas celulares es la coexistencia de células de diferentes tamaños.

Se puede proceder a una ulterior subdivisión sin necesidad de emplear más desplazamientos de estaciones base, sectorizando la cobertura, con lo que se mejora notablemente la interferencia cocanal.

2.9 Reutilización de Canales de Frecuencia

Un radiocanal consiste en un par de frecuencias, cada una para una dirección de transmisión utilizadas en un modo de transmisión Full-Duplex.[11]

Un radiocanal F1 utilizado en una zona geográfica llamada célula con un radio de cobertura R, puede ser utilizada en otra célula con el mismo radio de cobertura a una distancia "D" o distancia de reutilización. Esto lo podemos observar en la figura 2-4:

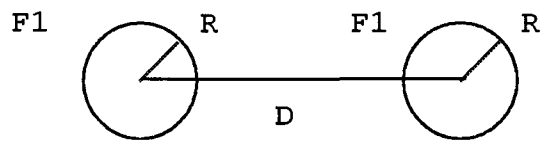


Figura 2-4

La interferencia dual producida por el uso común del mismo canal se denomina interferencia cocanal.

2.10 Planificación de un sistema TMA

A la hora de planificar un sistema móvil se consideran los siguientes factores:

- **Logísticos.**
- **Sistémicos.**
- **Radioeléctricos.**
- **De cobertura.**

2.10.1 Factores logísticos

- Tamaño de la zona de cobertura deseada y naturaleza del medio(rural, urbano, etc).
- Interconexión con otros sistemas, definiendo en su caso los interfaces necesarios.
- Compatibilidad electromagnética.
- Costes.

2.10.2 Factores sistémicos

- Estructura y naturaleza de la red.
- Capacidad en función del número de vehículos y tráfico previsto.
- Calidad y fiabilidad de las comunicaciones.
- Facilidades de control y tipo de señalización.

2.10.3 Factores Radioeléctricos

- Banda de frecuencias a utilizar.
- Tipo de equipos que se emplearán.
- Conseguir la cobertura adecuada.

2.10.4 Factores de cobertura

- Altura de la antena de la estación base sobre el nivel medio del terreno.
- Tipo de terreno, llano, ondulado, montañoso, zona rural o zona urbana.
- Ganancia de las antenas de las estaciones base y de las estaciones móviles.
- Potencia del transmisor (estaciones base y móviles).
- Ruido radioeléctrico en el ambiente, en la ubicación de la estación base o de la móvil.
- Sensibilidad del receptor.
- Pérdidas en el trayecto radioeléctrico entre la salida del transmisor de la estación base y la antena.

3

ANÁLISIS DE LA APLICACIÓN

En este capítulo conoceremos con todo detalle la aplicación TMA. Primero, veremos la partes de que se compone la ventana principal de la aplicación, así como todas las fórmulas utilizadas para realizar la presentación en pantalla de los cálculos obtenidos. A continuación se muestran todas las opciones del menú principal y la utilidad de cada una de ellas. En el tercer tema, se explica el funcionamiento del ratón dentro de esta aplicación. Por último, se muestra el motivo por el que puede aparecer en pantalla un mensaje de información.

Los temas a tratar son:

* **El área de cliente.**
* **El menú principal.**
* **El uso del ratón.**
* **Los mensajes de información.**

En la figura 3-1 nos encontramos con el aspecto que presenta la aplicación TMA al ser ejecutada desde Windows.

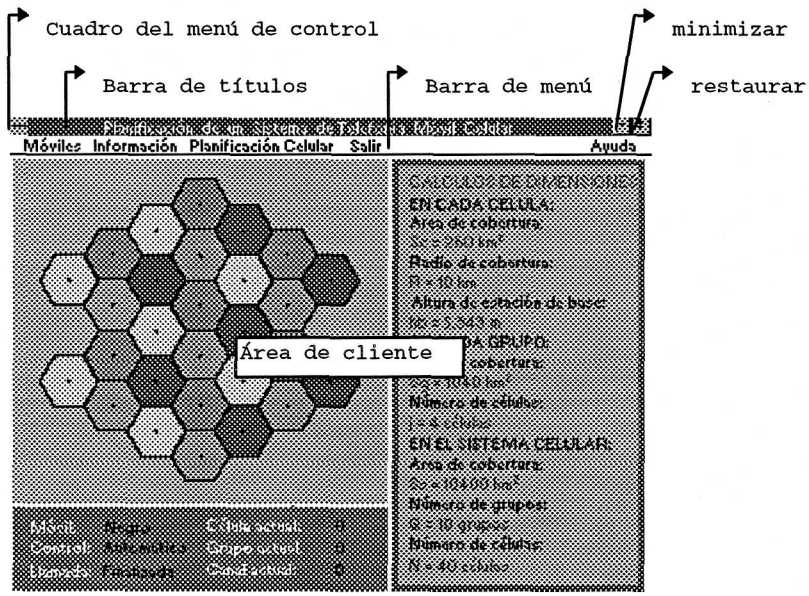


Figura 3-1

3.1 El Área de Cliente

El área de cliente de esta aplicación contiene tres partes diferenciadas y enmarcadas por rectángulos a modo de ventanas hijas:

- **La ventana superior izquierda**
- **La ventana inferior izquierda**
- **La ventana derecha**

3.1.1 La ventana superior izquierda

En esta ventana encontraremos un dibujo bidimensional de la configuración de un sistema celular. Se podrá visualizar un máximo de siete grupos y un mínimo de uno. Cada grupo podrá contener 3, 4, 7 ó 12 células.

Además existe la posibilidad de ver 5 móviles en dicha ventana, los cuales se irán desplazando dentro de ella de forma aleatoria. Uno de esos cinco móviles estará seleccionado y según se vaya desplazando activará la estación de base y el grupo en los que se encuentre localizado.

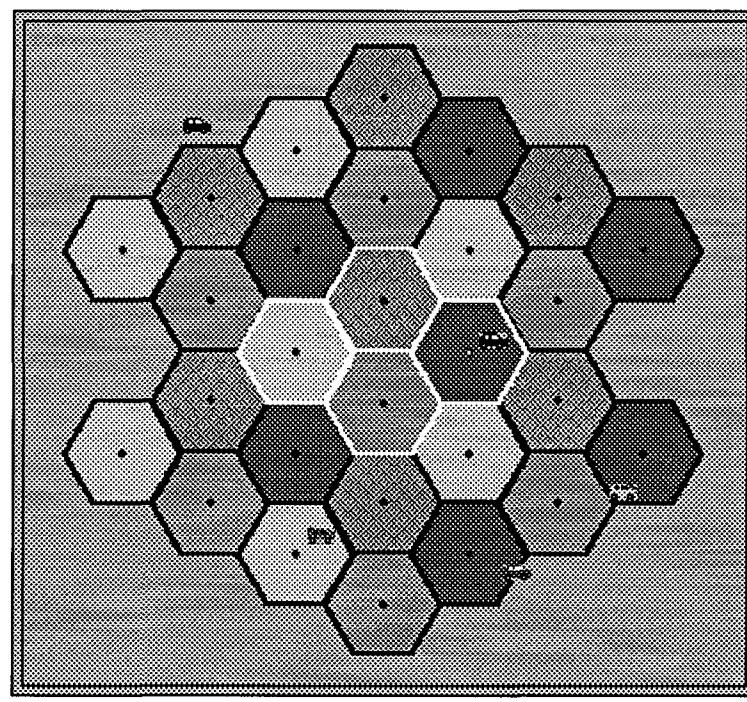


Figura 3-2

En la figura 3-2, se muestra la ventana superior izquierda de la aplicación configurada con siete grupos de 4 células. Podemos observar que la estación de base de la célula en la que se encuentra el móvil seleccionado, así como los bordes del grupo al que pertenece dicha célula, están señalizados con el color blanco. Mientras que el resto del sistema lo está con el color negro.

3.1.2 La ventana inferior izquierda

Esta ventana posee información acerca del móvil seleccionado, para hacer un seguimiento de su localización y del estado de las llamadas que realice o que reciba.

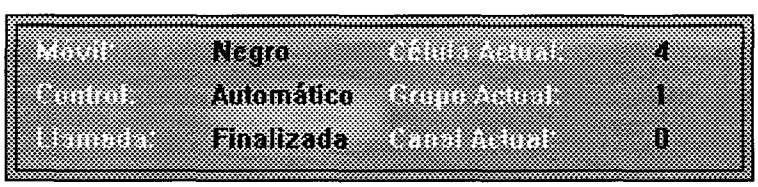


Figura 3-3

La información contenida en la ventana se visualizar con el mismo color que el móvil seleccionado por el usuario, (en este caso es el negro) como se muestra en la figura 3-3.

A continuación explicaremos el significado de cada uno de los datos que se muestran sobre el móvil seleccionado en dicha ventana.

- **Móvil:** Indica cuál es el móvil seleccionado.

- **Control:** Indica como se ha seleccionado la manera de desplazar al móvil seleccionado:

  1. Automático: Se desplaza de forma aleatoria.
  2. Ratón: El usuario controla el desplazamiento pulsando el botón izquierdo del ratón cuando el cursor se encuentre en la posición deseada.

- **Llamada:** Las llamadas pueden pasar por tres estados diferentes:

  1. *En Curso*: Cuando se mantiene la conversación.
  2. *Perdida*: Cuando se pierde la llamada.
  3. *Finalizada*: Cuando se "cuelga" el teléfono.

- **Célula:** Indica el número de la célula en la que se encuentra localizado el móvil seleccionado. El número será cero si se encuentra fuera del área de cobertura.

- **Grupo:** Indica el número del grupo en el que se encuentra localizado el móvil seleccionado. El número será cero si se encuentra fuera del área de cobertura.

- **Canal:** Indica el canal utilizado por el móvil seleccionado. Si la llamada se encuentra "Finalizada", el canal indicará el valor cero.

3.1.3 La ventana derecha

Esta es la ventana de información, cuyo contenido lo podemos cambiar pulsando una de las opciones del menú **Información** o pulsando el botón derecho del ratón siempre que este se encuentre dentro del área de cliente de nuestra aplicación.

Las distintas ventanas de información son las siguientes:

- **Cálculos de Dimensiones**
- **Cálculos Radioeléctricos**
- **Cálculos de Tráfico**
- **Cálculos sobre los Móviles**

3.1.3.1 Cálculos de Dimensiones

En esta ventana se presentan todos los cálculos realizados sobre las dimensiones de los distintos elementos que componen nuestro sistema celular.[10],[13]

- **Área de Cobertura**
- **Radio de cobertura de cada célula**
- **Altura de antena de estación de base**

✸ **Número de células y de grupos**

En la figura 3-4, podemos ver un ejemplo del contenido de esta ventana. Todos los cálculos obtenidos están en función de los valores iniciales de las variables del sistema.

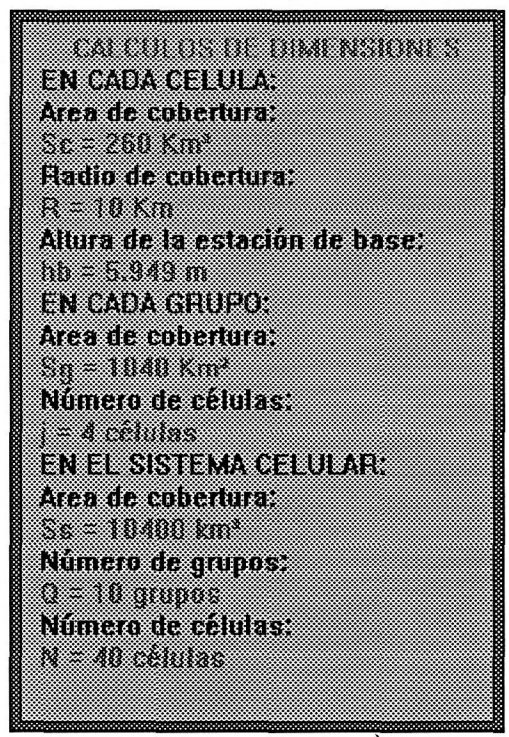


Figura 3-4

Área de Cobertura

El **área de cobertura** es la superficie en la que se puede atender a los abonados móviles y comunicarlos entre sí o con los teléfonos fijos.

✸ **En cada célula (Sc):** Como las células son hexagonales, su área de cobertura será igual a la superficie de un hexágono regular.

$$Sc = 2.6 * Radio^2 \quad km^2$$ **Ecuación 3-1**

- **En cada grupo (Sg):** Se define como el área de una célula multiplicado por el número de células que contiene el grupo (j).

$$Sg = Sc * j \quad km^2$$ **Ecuación 3-2**

- **En el sistema (Ss):** Se define como el área de una célula multiplicado por el número de células que contiene el sistema (N) y también como el área de un grupo multiplicado por el número de grupos que contiene el sistema (Q).

$$Ss = Sc * N = Sg * Q \quad km^2$$ **Ecuación 3-3**

Radio de Cobertura

En nuestra aplicación hemos supuesto que el radio de cobertura de todas las células es el mismo, y puede ser modificado por el usuario.

El radio de cobertura es la distancia máxima de la que puede alejarse un móvil respecto a una estación de base, sin que se pierda la señal. El valor de esta magnitud podrá ser introducido por el usuario en el cuadro de diálogo **Magnitudes Geométricas...** del menú desplegable **Planificación Celular**.

Altura de Antena de Estación de Base

Existe una expresión que relaciona el radio de cobertura de una célula con la altura de la antena de la estación de base de dicha célula (hb) y la altura de la antena de los teléfonos móviles(hm):

$$\text{Radio} = 4.1 * (\sqrt{hb} + \sqrt{hm}) \quad km$$ **Ecuación 3-4**

Podemos despreciar la altura de la antena de los teléfonos móviles (hm) con respecto a la antena de la estación de base (hb) y así obtenemos:

$$hb = (\text{Radio} / 4.1)^2 \quad \text{metros}$$ **Ecuación 3-5**

Número de Células y de Grupos

Si nos referimos al número de células contenidas en un grupo, lo identificamos con la letra "j". En cambio, si nos referimos al número de células contenidas en el sistema, lo haremos con la letra "N".

La relación entre ellas viene dada por:

$$N = j * Q \quad \text{células}$$ **Ecuación 3-6**

Donde "Q" es el número de grupos que conforman el sistema.

3.1.3.2 Cálculos Radioeléctricos

En la figura 3-5 se muestran todos los cálculos radioeléctricos de nuestro sistema celular.

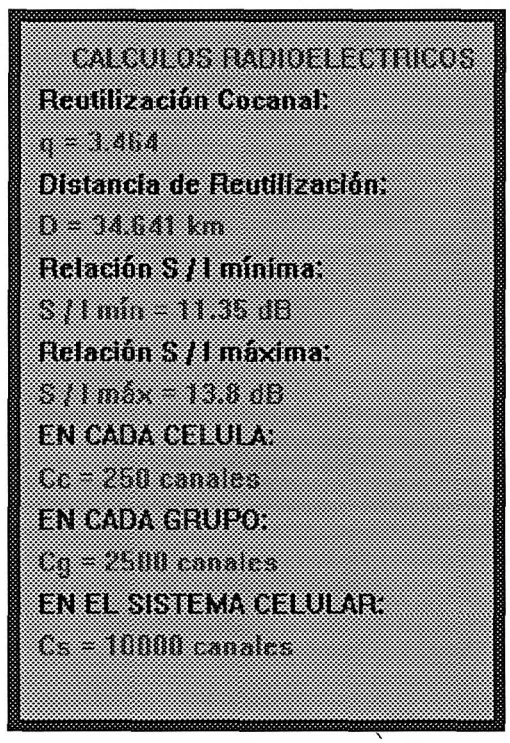


Figura 3-5

Su contenido es el siguiente:

- **Reutilización Cocanal**
- **Distancia de Reutilización**
- **Relación Señal / Interferencia**
- **Número de canales**

Distancia de Reutilización

Es la distancia mínima en la que pueden situarse dos estaciones de base que comparten el mismo juego de radiocanales. Depende tanto del radio de las células como del número de células por cada grupo (j):

$$D = Radio * \sqrt{(3 * j)} \quad Km$$

Ecuación 3-7

Reutilización Cocanal

La relación de reutilización cocanal (q), se define como la distancia de reutilización cocanal (D) dividida por el radio de la célula:

$$q = D / Radio = \sqrt{(3 * j)}$$

Ecuación 3-8

A medida que aumenta el número de células por grupo (j), también aumenta la relación de reutilización cocanal y se mejora la relación Señal / Interferencia por lo que se mejora la calidad de las llamadas telefónicas.

Relación Señal / Interferencia

La interferencia producida por el uso común de un mismo canal, se denomina interferencia cocanal, y es uno de los principales problemas de los sistemas celulares. A medida que reducimos el número de células dentro de un grupo para poder reutilizar el mayor número posible de radiocanales, disminuye la distancia de reutilización y consecuentemente aumenta la interferencia cocanal.

La relación Señal / Interferencia depende también del tipo de antenas utilizadas y del lugar donde se encuentre. Mientras más se aleje el móvil de la estación de base, mayor será la interferencia cocanal y por lo tanto peor será la relación Señal / Interferencia.

Número de Canales

El número de canales (Cc) disponibles dentro de una célula es igual al número de canales disponibles en un grupo (Cg) dividido por el número de células por grupo (j):

$$Cc = Cg / j \quad \text{canales}$$ **Ecuación 3-9**

Estando el número de canales disponibles en cada grupo, en función del número máximo que nos permite el ancho de banda (B) ofrecido para la transmisión de telefonía móvil celular y por la separación entre dichos canales (δf).

$$Cg = B / (2 * \delta f) \quad \text{canales}$$ **Ecuación 3-10**

El número de canales disponibles en el sistema celular es igual al número de canales disponibles en cada célula multiplicado por el número de células que conforman el sistema (N), y también al número de canales disponibles en cada grupo multiplicado por el número de grupos del sistema (Q):

$$Cs = Cc * N = Cg * Q \quad \text{canales}$$ **Ecuación 3-11**

3.1.3.3 Cálculos de Tráfico

En esta ventana podemos ver todos los cálculos realizados sobre el tráfico que podemos obtener a partir de la densidad de móviles por kilómetro cuadrado que requiera el usuario.

Los elementos de esta ventana son:

- **Densidad de tráfico**
- **Tráfico**

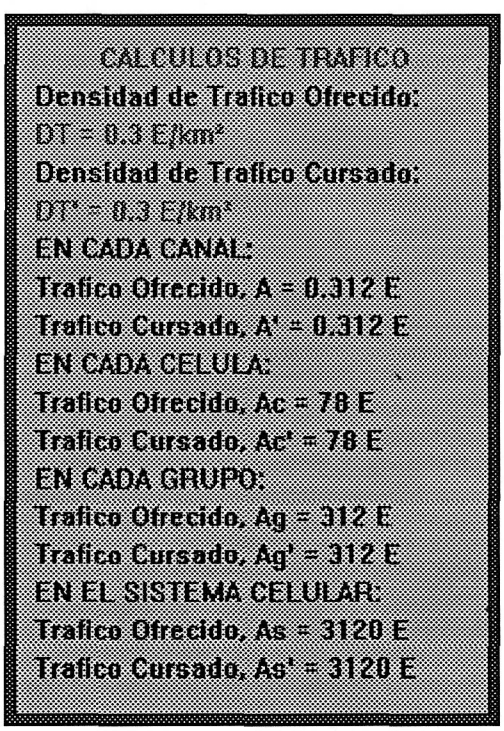


Figura 3-6

Densidad de Tráfico

La densidad de tráfico en cada célula, en cada grupo, y en el sistema es el mismo ya que su definición es la siguiente:

Densidad de tráfico ofrecido (DT): Se define como el tráfico ofrecido en una zona (hemos tomado una célula, Ac) dividido por la superficie de dicha zona (Sc).

$$DT = Ac / Sc \quad Erlangs/km^2$$ **Ecuación 3-12**

Densidad de tráfico cursado (DT'): Se define como el tráfico cursado en una zona (hemos tomado una célula, Ac') dividido por la superficie de dicha zona (Sc).

$$DT' = Ac'/ Sc \quad Erlangs/km^2$$ **Ecuación 3-13**

Se observa que cuando el tráfico ofrecido sea igual al tráfico cursado, las densidades de tráfico ofrecido y cursado también serán iguales, como se muestra en la figura 3-6.

Tráfico

Las magnitudes de tráfico ofrecido y cursado en cada célula, en cada canal, en cada grupo, y en el sistema se indican a continuación:

Tráfico ofrecido en cada célula (Ac): Se define como la densidad de móviles ,o número de móviles por kilómetro cuadrado (DM), multiplicado por el tráfico ofrecido por un móvil (e).

La densidad de móviles se puede especificar en el cuadro de diálogo **Magnitudes de tráfico...** del menú desplegable **Planificación Celular**. Se ha acordado en un valor fijo para indicar el tráfico ofrecido por un móvil (e = 0.03 Erlangs).

$$Ac = DM * e * Sc \quad Erlangs$$ **Ecuación 3-14**

Tráfico cursado en cada célula (Ac'): Se define como el tráfico ofrecido por una célula, multiplicado por el resultado de restarle a la unidad, la probabilidad de que se pierda una llamada telefónica (p).

$$Ac' = Ac * (1 - p) \quad Erlangs$$ **Ecuación 3-15**

Tráfico ofrecido por canal (A): Se define como el tráfico ofrecido por una célula, divido por el número de canales disponibles en cada célula (Cc).

$$A = Ac / Cc \quad Erlangs$$ **Ecuación 3-16**

Tráfico cursado por canal (A'): Se define como el tráfico cursado por una célula, divido por el número de canales disponibles en cada célula (Cc).

$$A' = Ac' / Cc \quad Erlangs$$ **Ecuación 3-17**

Tráfico ofrecido por un grupo (Ag): Se define como el tráfico ofrecido por una célula, multiplicado por el número de células por grupo del sistema (j).

$$Ag = Ac * j \quad Erlangs$$ **Ecuación 3-18**

Tráfico cursado por un grupo (Ag'): Se define como el tráfico cursado por una célula, multiplicado por el número de células por grupo del sistema (j).

$$Ag' = Ac' * j \quad \text{Erlangs}$$ **Ecuación 3-19**

Tráfico ofrecido por el sistema (As): Se define como el tráfico ofrecido por una célula, multiplicado por el número de células que forman el sistema (N).

$$As = Ac * N \quad \text{Erlangs}$$ **Ecuación 3-20**

Tráfico cursado por el sistema (As'): Se define como el tráfico cursado por una célula, multiplicado por el número de células que forman el sistema (N).

$$As' = Ac' * N \quad \text{Erlangs}$$ **Ecuación 3-21**

3.1.3.4 Cálculos sobre los Móviles

En esta ventana obtenemos los datos referentes a los teléfonos móviles de nuestro sistema celular, como se muestra en la figura 3-7.

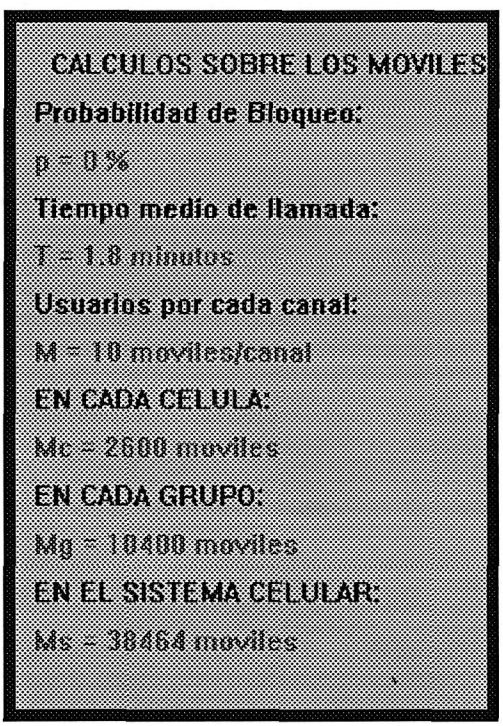


Figura 3-7

Los elementos de esta ventana son:

- **Probabilidad de bloqueo**
- **Tiempo de Llamada**
- **Usuarios por Canal**
- **Móviles**

Probabilidad de bloqueo

La fórmula B de Erlang nos ofrece el valor de la probabilidad de bloqueo de una llamada (p) en función del tráfico ofrecido y del número de canales.

Se define como la probabilidad de que al efectuar una llamada, esta se bloquee, y por lo tanto se pierda la comunicación al estar en ese momento todos los canales ocupados.

Su expresión es la siguiente:

$$p = B(c,a) = (a^c / c!)/(\sum^c_{i=0}(a^i / y!))$$ **Ecuación 3-22**

La probabilidad de bloqueo o de pérdida de expresa en tantos por cien, siendo c el número de canales y a el tráfico ofrecido.

Tiempo de Llamada

El tiempo medio de llamada está en función del tráfico cursado por una célula (Ac') y del número de móviles a los que se puede atender en dicha célula (Mc).

La siguiente expresión calcula el tiempo medio de duración de las llamadas en nuestro sistema celular:

$$T = 60 * Ac' / Mc \quad \text{minutos}$$ **Ecuación 3-23**

Usuarios por Canal

Puesto que estamos haciendo una reutilización de canales, podemos calcular el número de usuarios del sistema que comparten un mismo canal.

El número de móviles por canal (M), es igual al número de móviles que se pueden atender en una célula (Mc), dividido entre el número de canales disponibles en dicha célula (Cc).

$$M = Mc / Cc \quad \text{móviles}$$ **Ecuación 3-24**

Móviles

El número de abonados móviles a los que se puede atender, está en función de la densidad de móviles (DM) y de la superficie a cubrir.

Número de móviles por célula (Mc): Se define como la densidad de móviles multiplicada por la superficie de la célula (Sc).

$$Mc = DM * Sc \quad \text{móviles}$$ **Ecuación 3-25**

Número de móviles por grupo (Mg): Se define como el número de móviles por célula multiplicado por el número de células por grupo (j).

$$Mg = Mc * j \quad \text{móviles}$$ **Ecuación 3-26**

Número de móviles en el sistema (Ms): Se define como el número de móviles por célula multiplicado por el número de células del sistema (N).

$$Ms = Mc * N \quad \text{móviles}$$ **Ecuación 3-27**

3.2 El menú principal

El menú principal de esta aplicación consta de los siguientes menús desplegables:

- **Móviles**
- **Información**
- **Planificación Celular**
- **Salir**
- **Ayuda**

3.2.1 Móviles

En la figura 3-8 se muestra el contenido del menú **Móviles...**, al inicio de esta aplicación.

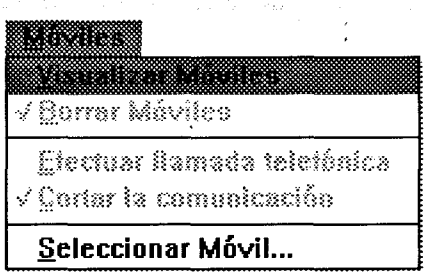


Figura 3-8

Este es el primer menú desplegable de la aplicación, donde podemos seleccionar cualquiera de las siguientes opciones:

- **Visualizar Móviles**
- **Borrar Móviles**
- **Efectuar llamada telefónica**
- **Cortar la comunicación**
- **Seleccionar Móvil...**

3.2.1.1 Visualizar Móviles

Al seleccionar esta opción, podemos ver 5 móviles (coches) en pantalla, que se irán desplazando dentro de la ventana superior izquierda de la aplicación. A partir de ese momento, esta opción no podrá ser nuevamente seleccionada mientras estén presentes los móviles en pantalla.

Este es el primer paso que se debe seguir para poder simular el comportamiento de los abonados móviles en nuestro sistema.

3.2.1.2 Borrar Móviles

El usuario podrá acceder a esta opción siempre que se encuentren los móviles presentes en el área de cliente de la aplicación.

También se activa cuando el usuario modifique el tamaño de la ventana principal de la aplicación ya sea utilizando las cajas de *maximizar* o *minimizar*, o también con el propio cursor del ratón, redimensionando los bordes de la ventana, siempre que estos se encuentren presentes en pantalla.

La última causa por la que se podría activar esta opción es que el usuario mueva alguna ventana, caja de diálogo o de mensajes, por encima de la ventana superior izquierda, donde se encuentran activados los móviles.

3.2.1.3 Efectuar llamada telefónica

Sólo se podrá acceder a esta opción cuando los móviles estén visualizados y la ventana inferior izquierda indique que la llamada anterior está finalizada. A medida que se vayan desplazando los móviles y esta opción esté activada, podrán producirse ciertos mensajes de información en pantalla.

3.2.1.4 Cortar la Comunicación

Sólo se podrá acceder a esta opción, cuando los móviles estén visibles en pantalla y la ventana inferior izquierda indique que la llamada efectuada está en curso o perdida.

3.2.1.5 Seleccionar Móvil...

Esta opción abre el cuadro de diálogo que se muestra en la figura 3-9. Dicho cuadro de diálogo permite hacer ciertas modificaciones sobre los siguientes aspectos que conciernen a los abonados móviles:

- **Movimiento**
- **Mensajes en Pantalla**
- **Móvil Seleccionado**
- **Velocidad**
- **Recibir Llamadas**

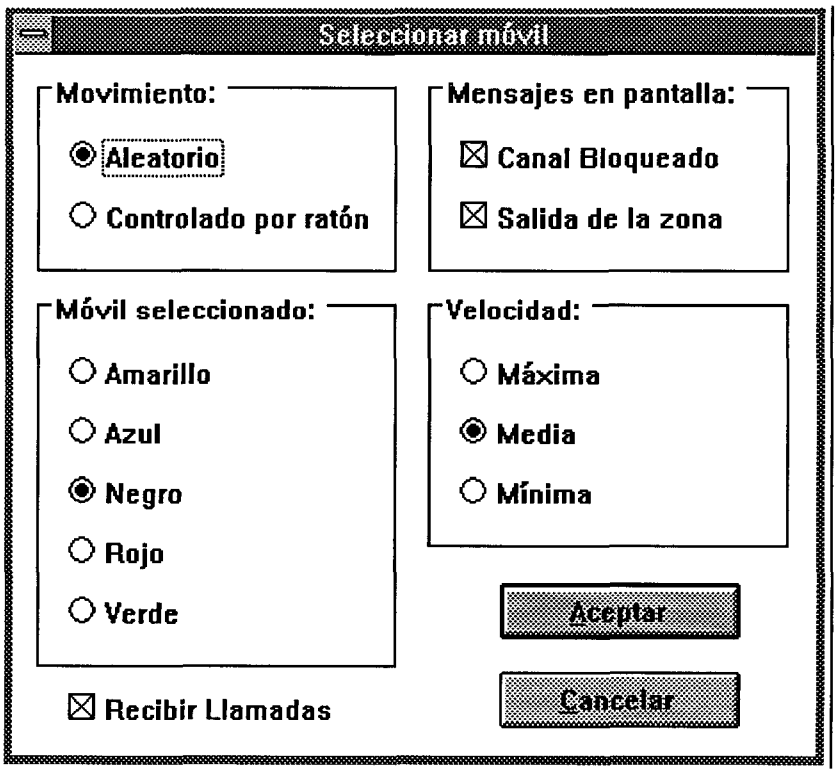


Figura 3-9

Movimiento

Dentro de esta caja de grupo podemos seleccionar el tipo de movimiento que realizará el móvil seleccionado.

Si elegimos la opción **Aleatorio**, el móvil seleccionado se desplazará de forma aleatoria, al igual que el resto de los móviles.

Si seleccionamos la opción **Controlado por Ratón**, el usuario podrá dirigir al móvil seleccionado a la posición que desee dentro de la ventana superior izquierda del área de cliente, desplazando el cursor a la posición deseada y haciendo "clic" con el botón izquierdo del ratón.

Mensajes en Pantalla

Dentro de esta caja de grupo podemos decidir si queremos que la aplicación nos avise cuando el móvil seleccionado ha salido del área de cobertura mientras estaba cursando una llamada telefónica. Esto lo hacemos seleccionado la opción **Salida de la zona** del menú **Móviles**.

También podemos decidir si queremos que la aplicación nos avise cuando se ha perdido una llamada telefónica, seleccionado la opción **Canal Bloqueado** del menú **Móviles**.

Móvil Seleccionado

Dentro de esta caja de grupo podemos seleccionar uno de los 5 móviles que se muestran en la figura 3-10, para hacer un seguimiento de este, a lo largo del sistema celular. En la ventana inferior izquierda de la aplicación podemos observar cuál es el móvil seleccionado.

Figura 3-10

Velocidad

En el interior de esta caja de grupo, podemos seleccionar una de las tres velocidades posibles con la que se desplazarán los móviles, a través de la ventana superior izquierda de la aplicación.

Máxima -> Se desplazan cada 0.5 segundos
Media -> Se desplazan cada segundo
Mínima -> Se desplazan cada 2 segundos

Recibir Llamadas

Esta caja de selección nos permite recibir llamadas de forma aleatoria con un 5% de probabilidad de que se produzca una llamada cada vez que se desplacen los móviles, y siempre que se encuentre la opción **llamada,** de la ventana inferior izquierda, finalizada.

3.2.2 Información

La figura 3-11 contiene el cuadro de diálogo **Información,** indicando su contenido inicial.

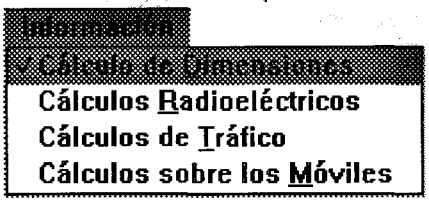


Figura 3-11

3.2.2.1 Cálculo de Dimensiones

Esta opción activa la ventana **Cálculos de Dimensiones** en el área de la ventana derecha de la aplicación.

3.2.2.2 Cálculos Radioeléctricos

Esta opción activa la ventana **Cálculos Radioeléctricos** en el área de la ventana derecha de la aplicación.

3.2.2.3 Cálculos de Tráfico

Esta opción activa la ventana **Cálculos de Tráfico** en el área de la ventana derecha de la aplicación.

3.2.2.4 Cálculos sobre los Móviles

Esta opción activa la ventana **Cálculos sobre los Móviles** en el área de la ventana derecha de la aplicación.

3.2.3 Planificación Celular

Figura 3-12

La figura 3-12 contiene un menú desplegable, que posee las siguientes cajas de diálogo, que permiten seleccionar las condiciones requeridas por cada sistema de telefonía móvil celular:

- **Magnitudes Geométricas**
- **Magnitudes Radioeléctricas**
- **Magnitudes de Tráfico**

3.2.3.1 Magnitudes Geométricas

En esta caja de diálogo, el usuario puede seleccionar o definir los valores de las opciones que se muestran en la figura 3-13.

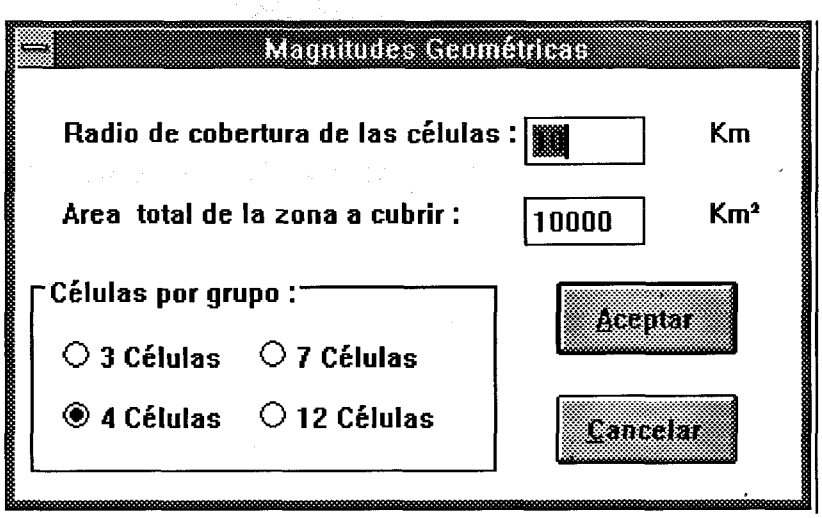


Figura 3-13

Radio de Cobertura de las Células

En este cuadro de edición, el usuario puede introducir un número real para indicar el radio de cobertura de cada una de la células que forman nuestro sistema de telefonía móvil celular.

En la práctica se tiende a hacer dicho radio lo menor posible, siendo el mínimo radio de 1 Km. Además, en el peor de los casos, nunca excede de 30 Km.

Área Total de la Zona a Cubrir

En este cuadro de edición, el usuario puede introducir un número real para indicar el tamaño de la superficie total a cubrir por nuestro sistema de telefonía móvil celular.

La superficie total a cubrir debe ser lo suficientemente grande como para que se pueda instalar al menos una célula en nuestro sistema.

Células por Grupo

Dentro de este cuadro de grupo, podemos seleccionar el número de células que formaran cada grupo. Este número puede ser 3, 4, 7, ó 12.

3.2.3.2 Magnitudes Radioeléctricas

En la caja de diálogo de la figura 3-14, podemos realizar los cambios pertinentes sobre las variables radioeléctricas:

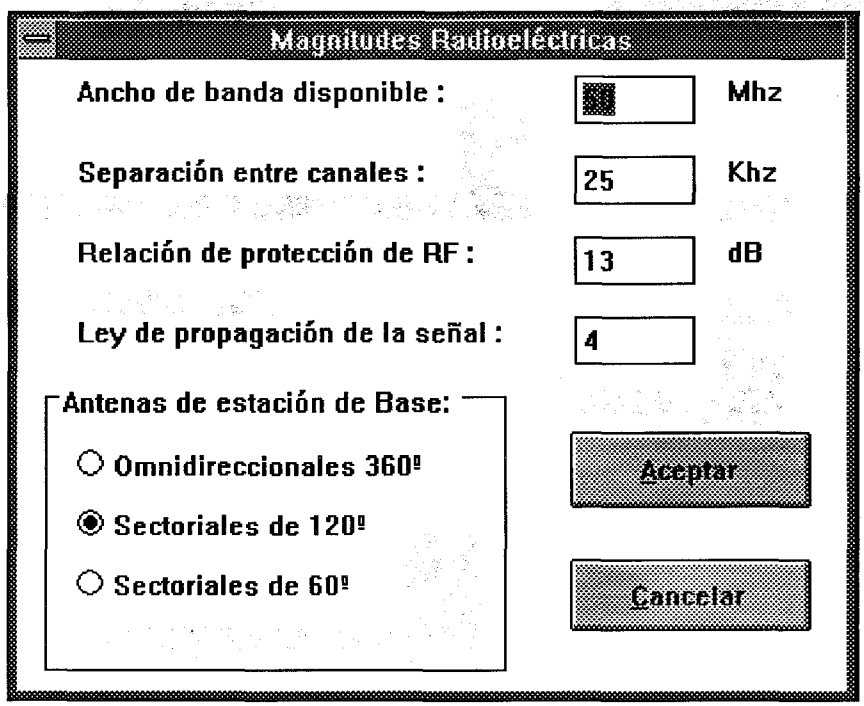


Figura 3-14

Ancho de Banda Disponible

En este cuadro de edición, el usuario puede introducir un número real para indicar el ancho de banda disponible en nuestro sistema de telefonía móvil celular.

El ancho de banda asignado a la telefonía móvil suele estar por debajo de los 50 MHz.

Separación entre canales

En esta caja de edición, el usuario puede introducir un número real para indicar la separación entre los canales asignados a nuestro sistema de telefonía móvil celular.

Una separación adecuada estaría en torno a los 25 KHz.

Relación de Protección de Radiofrecuencia

En esta caja de edición, el usuario puede introducir un número real para indicar el valor mínimo requerido de la relación Señal / Interferencia en nuestro sistema de telefonía móvil celular.

En esta aplicación se puede alcanzar una relación S/I máxima de aproximadamente 32 dB.

Ley de Propagación de la Señal

En este cuadro de edición, el usuario puede introducir un número real para indicar el valor de la Ley de Propagación de la señal donde se ubique nuestro sistema de telefonía móvil celular.

Siendo los valores típicos:

- Antenas elevadas -> 2
- Zona Urbana -> 3.5 - 3.9
- Terreno plano -> 4

Antenas de Estación de Base

A medida que las antenas de estación de base cubren un área o sector menor, disminuye la interferencia cocanal producida por las antenas que comparten un mismo juego de radiocanales.

En esta caja de grupo podemos seleccionar uno de los tres tipos de antena que se utilizan en los sistemas de telefonía móvil celular:

- **Omnidireccionales (360º)**
- **Sectoriales de 120º**
- **Sectoriales de 60º**

Omnidireccionales

Este tipo de antena cubre un área circular de 360º, por lo que sólo se necesita una antena para cubrir el área correspondiente a una célula. Tiene el inconveniente de producir una mayor interferencia cocanal con respecto a las antenas sectoriales.

S/I mínima: $q^4 / 6$ dB **Ecuación 3-28**

S/I máx: $1/(2(q-1)^{-4}+2*q^{-4}+2(q+1)^{-4})$ dB **Ecuación 3-29**

Siendo "q" la relación de reutilización cocanal.

Sectoriales de 120º

Este tipo de antena cubre un sector de 120º dentro del área de un circulo, por lo que se necesitan **3** antenas para cubrir el área correspondiente a una célula.

Se mejora la interferencia cocanal con respecto a las antenas omnidireccionales, pero tiene el inconveniente de multiplicar por tres la posibilidad de que se produzca el Handover.

S/I mínima: $q^4 / 2$ dB **Ecuación 3-30**

S/I máxima: $1 / ((q+0.7)^{-4} + q^{-4})$ dB **Ecuación 3-31**

Siendo "q" la relación de reutilización cocanal.

Sectoriales de 60º

Este tipo de antena cubre un sector de 60º dentro del área de un circulo, por lo que se necesitan **6** antenas para cubrir el área correspondiente a una célula. Se mejora la interferencia cocanal con respecto a las antenas sectoriales de 120º, pero tiene el inconveniente de multiplicar por 6 la posibilidad de que se produzca el Handover.

S/I mínima: q^4 dB **Ecuación 3-32**

S/I máxima: $(q + 0.7)^4$ dB **Ecuación 3-33**

3.2.3.3 Magnitudes de Tráfico

En la caja de diálogo de la figura 3-15, podemos definir la densidad de móviles que requiera nuestro sistema:

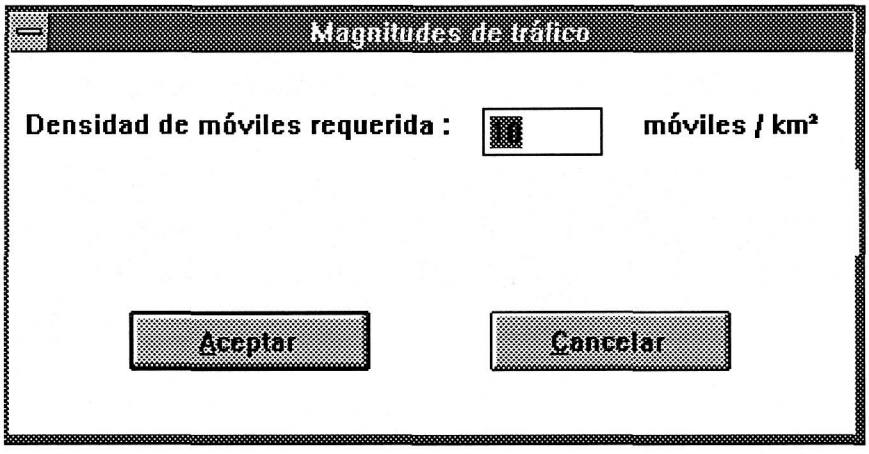


Figura 3-15

Densidad de Móviles Requerida

Esta caja de edición permite introducir un número real para indicar el número de móviles que se va a atender en cada kilómetro cuadrado de nuestro sistema de telefonía móvil.

3.2.4 Salir

Esta opción del menú principal abre una caja de mensajes, donde se le pregunta al usuario si realmente quiere abandonar la aplicación o no, para evitar una salida de la aplicación por error del usuario al activar el comando **Salir** del menú principal. Esta ventana la podemos ver en la figura 3-16:

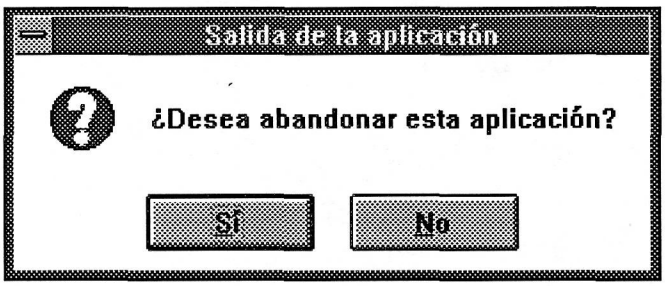


Figura 3-16

3.2.5 Ayuda

En la ventana de la figura 3-17 tenemos todas las diversas opciones del menú **Ayuda** de la aplicación.

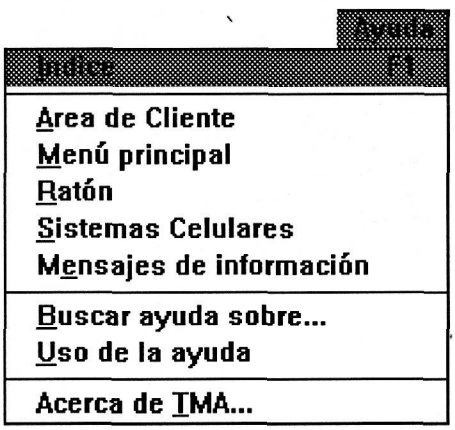


Figura 3-17

3.2.5.1 Índice

Seleccionando esta opción accedemos al índice de la ayuda donde se encuentran los temas principales de la ayuda.

3.2.5.2 Área de cliente

Con esta opción accedemos al tema de ayuda referente al área de cliente de la aplicación.

3.2.5.3 Menú principal

Con esta opción accedemos al tema de ayuda referente al menú de la aplicación.

3.2.5.4 Ratón

Con esta opción accedemos al tema de ayuda referente al uso del ratón en la aplicación.

3.2.5.5 Mensajes de información

Con esta opción accedemos al tema de ayuda referente a los tipos mensajes que se producen en la aplicación.

3.2.5.6 Buscar ayuda sobre

Con esta opción accedemos a un cuadro de diálogo donde podemos buscar ayuda sobre algún término o palabra clave que se utilice en la aplicación. Dicho cuadro de diálogo está representado en la figura 3-18:

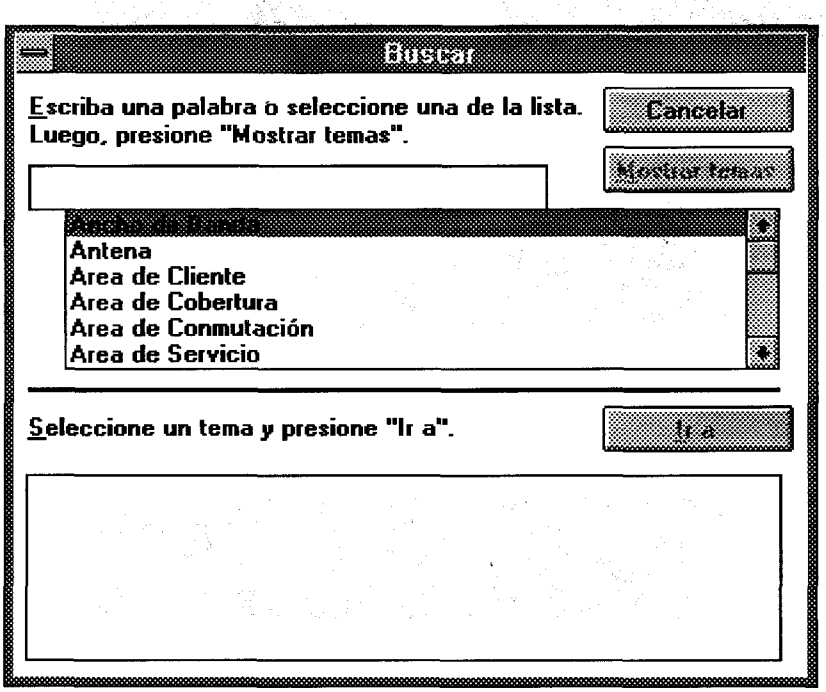


Figura 3-18

3.2.5.7 Uso de la ayuda

Seleccionando esta opción aprenderemos a utilizar la ayuda de Windows.

3.2.5.8 Acerca de...

En la figura 3-19 podemos ver un cuadro de diálogo que contiene el nombre del autor del proyecto y el año en el que realizó esta aplicación.

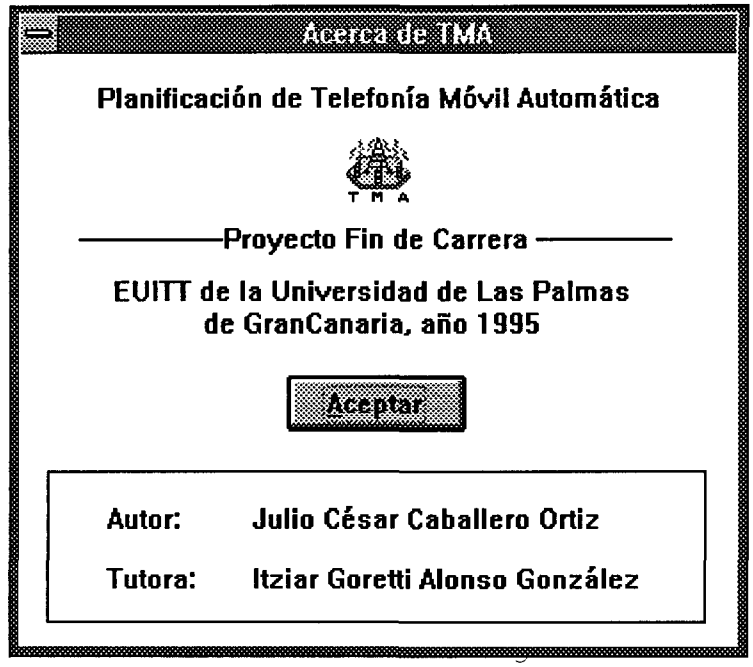


Figura 3-19

3.3 El uso del ratón

Manejar la aplicación con el ratón es muy sencillo, éste se presenta en la pantalla con forma de flecha. Su forma puede variar cuando se realiza una llamada pasando por uno de los siguientes cursores:

Flecha -> Llamada finalizada

Teléfono -> Llamada en curso

Cruz -> Llamada perdida

3.3.1 El botón izquierdo

Utilizaremos este botón para seleccionar alguna opción de los menús o de los cuadros de diálogo en nuestra aplicación.

También podemos desplazar el móvil seleccionado dentro de la ventana superior izquierda al pulsar este botón, cuando esté seleccionada la opción **Controlado por Ratón** del cuadro de diálogo **Seleccionar Móvil**. Este cuadro de diálogo lo podemos encontrar en el menú **Móviles**.

3.3.2 El botón derecho

Mientras el cursor permanezca en el área de cliente de nuestra aplicación, al pulsar este botón se cambiará la opción seleccionada en el menú **Información**, de forma consecutiva. Visualizaremos en pantalla la ventana correspondiente al tema seleccionado.

3.4 Mensajes de Información

En nuestra aplicación, aparecerán estos mensajes cuando ocurra alguno de estos sucesos:

- Pulsar el botón izquierdo del ratón fuera de la ventana superior izquierda cuando el móvil está en el modo Controlado por ratón.

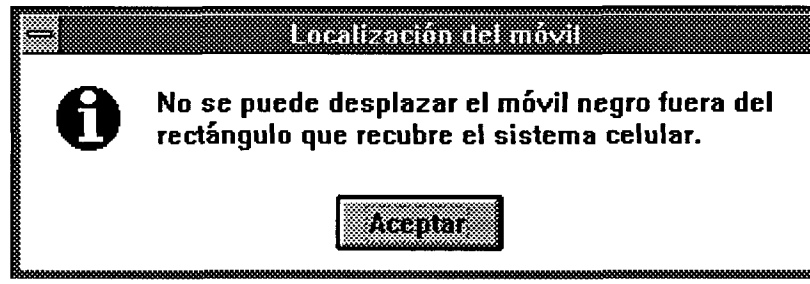


Figura 3-20

- Pulsar el botón izquierdo del ratón dentro del área de cliente cuando el móvil está en el modo Automático.

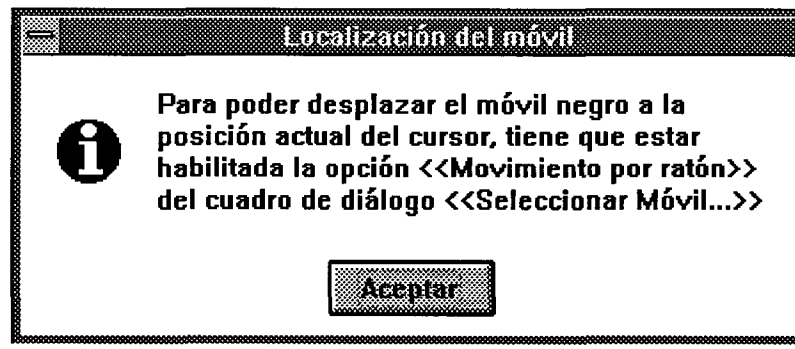


Figura 3-21

- Recibir una llamada telefónica.

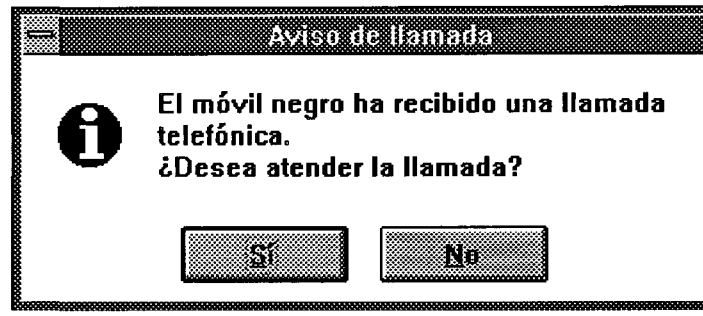


Figura 3-22

- Perder la llamada telefónica por no encontrarse ningún canal disponible, si está habilitada la opción Canal Bloqueado.

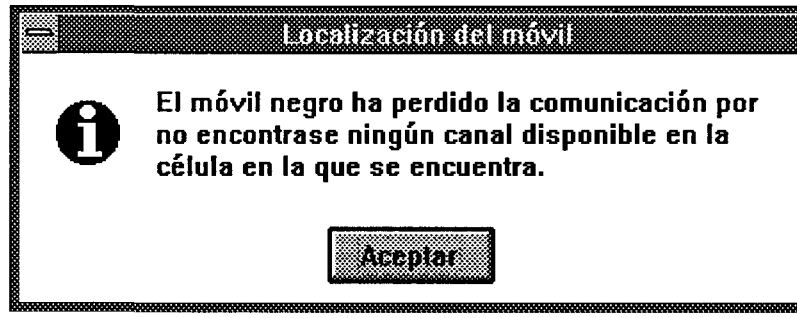


Figura 3-23

- Perder la llamada telefónica por encontrarse el móvil fuera del área de cobertura, si está habilitada la opción Salida Zona.

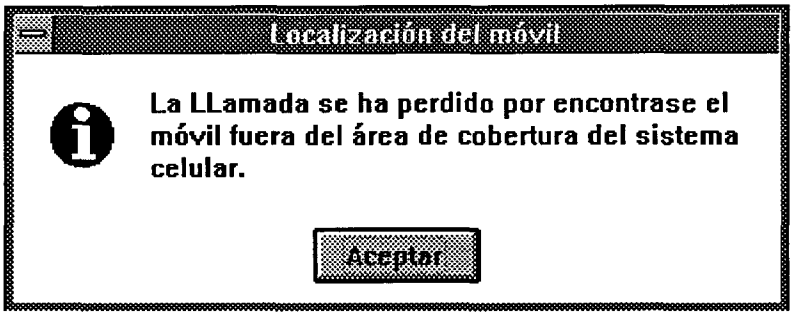


Figura 3-24

- Especificar las magnitudes geométricas de forma contradictoria.

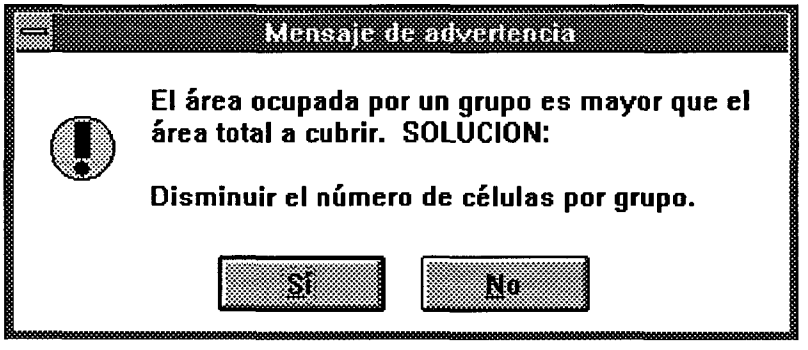


Figura 3-25

Figura 3-26

- Especificar las magnitudes radioeléctricas de forma contradictoria.

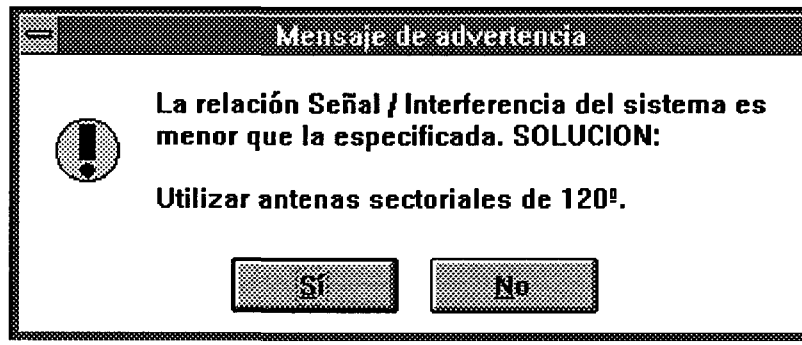


Figura 3-27

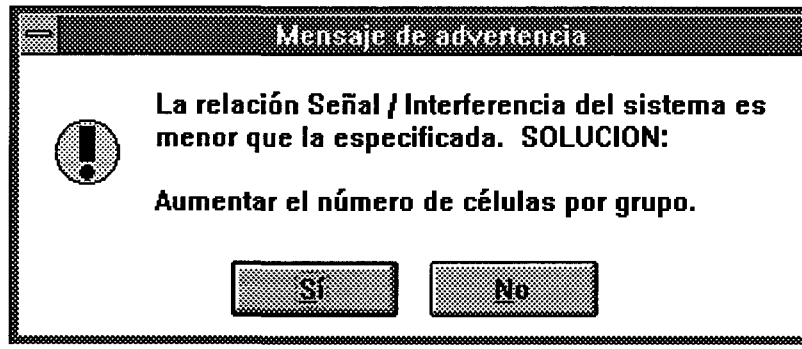


Figura 3-28

4

LAS CLASES Y FUNCIONES DE TMA

En este capítulo se conocerán las clases que conforman la aplicación TMA. Se explicará el funcionamiento de sus funciones miembros y se incluirán diagramas de flujo en aquellas funciones que por su complejidad lo requieran.

La aplicación TMA consta de las siguientes clases:

- **Tapp**
- **TMainWindow**
- **MovilDialog**
- **GeomeDialog**
- **RadioDialog**
- **TraficoDialog**

4.1 La clase Tapp

Esta clase representa a la aplicación en ejecución, y se deriva de la clase TApplication pasándole los parámetros adecuados. TApplication es una clase de ObjectWindows que representa una única instancia de ejecución de un módulo.

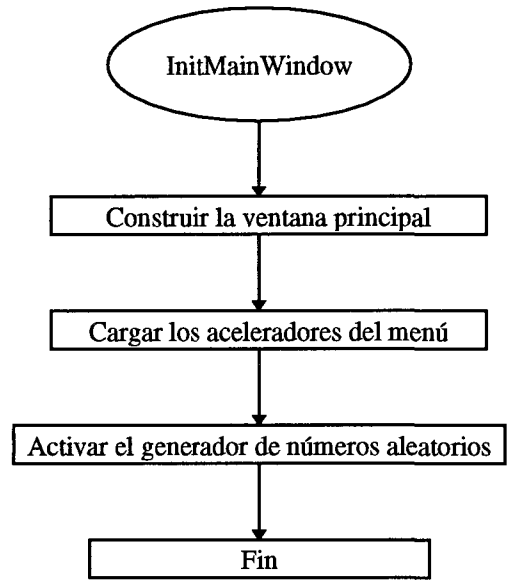
Derivar una clase de aplicación de TApplication, es el método normal de declarar y definir objetos de aplicación específicos para los programas de ObjectWindows.

```
class TApp : public TApplication
{
public:
        //Declaración y definición del constructor de la aplicación.
        TApp (LPSTR AName, HINSTANCE hInstance, HINSTANCE hPrevInstance,
                LPSTR lpCmdLine, int nCmdShow) : TApplication(AName, hInstance, hPrevInstance,
                lpCmdLine, nCmdShow) {};

protected:
        //Crea e inicializa una instancia de la ventana principal.
        virtual void InitMainWindow ();
};
```

4.1.1 Funciones de Tapp

void InitMainWindow()

Esta función es la encargada de construir un nuevo objeto ventana principal. No se le pasa ningún parámetro. Las variables utilizadas son miembro de TApplication.




```
void TApp :: InitMainWindow ()
{
        MainWindow = new TMainWindow(NULL, Name);
        HAccTable = LoadAccelerators(hInstance, "tmaMenu");
        randomize();
}
```


4.2 La clase TMainWindow

TMainWindow se deriva de la clase TWindow y será la ventana principal de esta aplicación. TWindow es una clase de ObjectWindows que representa a las ventanas que no son cuadros de diálogo. En TMainWindow definiremos un conjunto de funciones y variables que nos permitirán trabajar con los mensajes dirigidos a la ventana principal, así como la gestión de los menús de la aplicación.

```
class TMainWindow : public TWindow
{
public:
        //Variables los buffers de transferencia de datos.
        MovilTransferBuffer     MovilBuffer;
        GeomeTransferBuffer     GeomeBuffer;
        RadioTransferBuffer     RadioBuffer;
        TraficoTransferBuffer   TraficoBuffer;

        //Constructor de la ventana principal.
        TMainWindow (PTWindowsObject AParent, LPSTR ATitle);

        //Gestión de apertura de los menús desplegables.
        virtual void WMInitMenuPopup (RTMessage Msg) = [WM_FIRST + WM_INITMENUPOPUP];

        //Gestión de las selecciones realizadas en un menú.
        virtual void WMMenuSelect (RTMessage Msg) = [WM_FIRST + WM_MENUSELECT];

        //Gestión de la activación de la ventana principal.
        virtual void WMActivate (RTMessage Msg) = [WM_FIRST + WM_ACTIVATE];

        //Gestión de los cambios de dimensión de la ventana principal.
        virtual void WMSize (RTMessage Msg) = [WM_FIRST + WM_SIZE];

        //Gestión del movimiento del ratón.
        virtual void WMMouseMove (RTMessage Msg) = [WM_FIRST + WM_MOUSEMOVE];

        //Gestión de las pulsaciones del botón izquierdo del ratón.
        virtual void WMLButtonDown (RTMessage Msg) = [WM_FIRST + WM_LBUTTONDOWN];

        //Gestión de las pulsaciones del botón derecho del ratón.
        virtual void WMRButtonDown (RTMessage Msg) = [WM_FIRST + WM_RBUTTONDOWN];

        //Gestión de la activación del timer.
        virtual void WMTimer (RTMessage Msg) = [WM_FIRST + WM_TIMER];

        //Gestión de la opción de menú "Visualizar móviles".
        virtual void CMVisualizarMoviles (RTMessage Msg) = [CM_FIRST +
                CM_VISUALIZARMOVILES];

        //Gestión de la opción de menú "Borrar móviles".
        virtual void CMBorrarMoviles (RTMessage Msg) = [CM_FIRST + CM_BORRARMOVILES];

        //Gestión de la opción de menú "Efectuar llamada telefónica".
        virtual void CMEfectuarLlamada (RTMessage Msg) = [CM_FIRST +
                CM_EFECTUARLLAMADA];

        //Gestión de la opción de menú "Cortar la comunicación".
        virtual void CMCortarComunicacion (RTMessage Msg) = [CM_FIRST +
                CM_CORTARCOMUNICACION];
```

```
//Gestión de la opción de menú "Seleccionar móvil...".
virtual void CMSeleccionarMovil (RTMessage Msg) = [CM_FIRST +
        CM_SELECCIONARMOVIL];

//Gestión de la opción de menú "Cálculo de Dimensiones".
virtual void CMDimensiones (RTMessage Msg) = [CM_FIRST + CM_DIMENSIONES];

//Gestión de la opción de menú "Cálculos Radioeléctricos".
virtual void CMRadioelectricos (RTMessage Msg) = [CM_FIRST + CM_RADIOELECTRICOS];

//Gestión de la opción de menú "Cálculos de tráfico".
virtual void CMTrafico (RTMessage Msg) = [CM_FIRST + CM_TRAFICO];

//Gestión de la opción de menú "Cálculos sobre los móviles".
virtual void CMMoviles (RTMessage Msg) = [CM_FIRST + CM_MOVILES];

//Gestión de la opción de menú "Magnitudes geométricas...".
virtual void CMMagGeometricas (RTMessage Msg) = [CM_FIRST +
        CM_MAGGEOMETRICAS];

//Gestión de la opción de menú "Magnitudes radioeléctricas...".
virtual void CMMagRadioelectricas (RTMessage Msg) = [CM_FIRST +
        CM_MAGRADIOELECTRICAS];

//Gestión de la opción de menú "Magnitudes de tráfico...".
virtual void CMMagTrafico (RTMessage Msg) = [CM_FIRST + CM_MAGTRAFICO];

//Gestión de la opción de menú "Salir".
virtual void CMSalir (RTMessage Msg) = [CM_FIRST + CM_SALIR];

//Gestión de la opción de menú "Indice".
virtual void CMIndice (RTMessage Msg) = [CM_FIRST + CM_INDICE];

//Gestión de la opción de menú "Area de Cliente".
virtual void CMAyudaAreaCliente (RTMessage Msg) = [CM_FIRST +
        CM_AYUDAAREACLIENTE];

//Gestión de la opción de menú "Menú principal".
virtual void CMAyudaMenu (RTMessage Msg) = [CM_FIRST + CM_AYUDAMENU];

//Gestión de la opción de menú "Ratón".
virtual void CMAyudaRaton (RTMessage Msg) = [CM_FIRST + CM_AYUDARATON];

//Gestión de la opción de menú "Sistemas Celulares".
virtual void CMAyudaSistemas (RTMessage Msg) = [CM_FIRST + CM_AYUDASISTEMAS];

//Gestión de la opción de menú "Mensajes de información".
virtual void CMAyudaMensajes (RTMessage Msg) = [CM_FIRST + CM_AYUDAMENSAJES];

//Gestión de la opción de menú "Buscar Ayuda sobre...".
virtual void CMBuscarAyuda (RTMessage Msg) = [CM_FIRST + CM_BUSCARAYUDA];
```

```
        //Gestión de la opción de menú "Uso de la ayuda".
        virtual void CMUsoAyuda (RTMessage Msg) = [CM_FIRST + CM_USOAYUDA];

        //Gestión de la opción de menú "Acerca de TMA...".
        virtual void CMAcercade (RTMessage Msg) = [CM_FIRST + CM_ACERCADE];

        //Protección de la salida de la aplicación.
        virtual BOOL CanClose ();

protected:
        //Definición de ciertas propiedades y recursos de la aplicación".
        virtual void GetWindowClass (WNDCLASS & WndClass);

        //Gestión de los mensajes de dibujo en la ventana principal".
        virtual void Paint (HDC hdc, PAINTSTRUCT &PS);

private:
        //Variables globales de la clase TMainWindow.
        BYTE    byBaseAnterior,// Indica la base a la que estaba enganchado el móvil seleccionado.
                byCelulasGrupo,// Indica cuántas células conforman un grupo.
                byGrupoAnterior,// Indica el grupo al que estaba enganchado el móvil seleccionado.
                byMovilSeleccionado;// Indica cuál es el móvil seleccionado.

        BOOL    bCortaComunicacion,// Indica si la llamada telefónica se ha cortado.
                bCursorDentro,// Indica si el cursor se encuentra dentro de la ventana superior izquierda.
                bIniciaLlamada,// Indica si se acaba de iniciar la llamada.
                bLlamadaEnCurso,// Indica si la llamada está en curso.
                bLlamadaPerdida,// Indica si se ha perdido la llamada.
                bLlamadaRecibida,// Indica si se ha recibido una llamada.
                bMovilesActivados,// Indica si los móviles están visualizados en pantalla.
                bYaEnganchado;//Si el móvil seleccionado se encontraba dentro del área de cobertura.

        char    sCanal[5],// Contiene el el canal utilizado por el móvil seleccionado.
                sCelula[3],// Contiene la célula en la que se encuentra el móvil seleccionado.
                sControl[11],// Forma en la que se controla el movimiento del móvil seleccionado.
                sGrupo[2],// Contiene el grupo en el que se encuentra el móvil seleccionado.
                sLlamada[11],// Contiene el estado de la llamada telefónica del móvil seleccionado.
                sMovil[9];// Contiene el móvil seleccionado.

        DWORD           dwColor[NumMoviles][xBmp][yBmp];
                        // Almacena el color de los pixels que van a ser redibujados por los móviles.

        float   fAreaCelula,// Valor del área ocupada por cada célula.
                fBloqueo,// Valor de la probabilidad de bloqueo de una llamada.
                fMaximaRSI//Valor máximo de la relación Señal/Interferencia.
                fMinimaRSI,// Valor mínimo de la relación Señal/Interferencia.

        POINT   *pOrigenCelulas,// Apuntará a un vector de las coordenas de los centros de las células.
                ptActual[NumMoviles],// Almacena las coordenadas actuales de los móviles.
                ptAnterior[NumMoviles],// Almacena las coordenadas anteriores de los móviles.
                ptFinal[NumMoviles],//Almacena las coordenadas hacia donde se desplazarán los móviles
                ptVentana;// Almacena las coordenadas de la ventana principal.
```

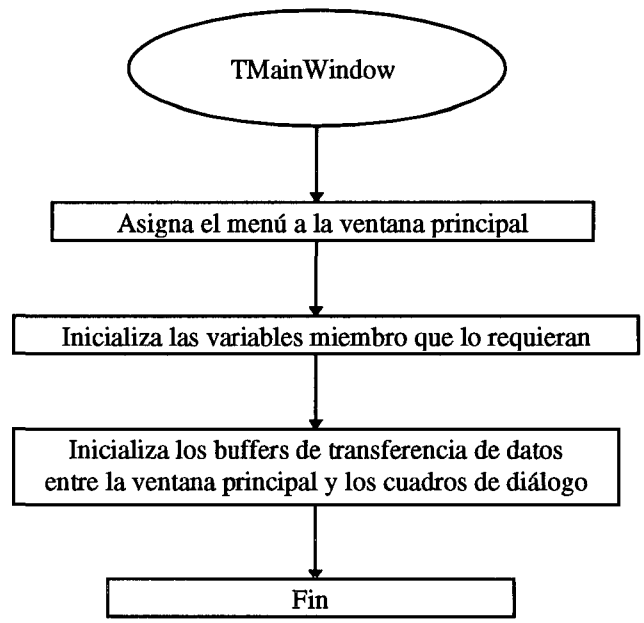
```
WORD  wCanalesCelula,// Indica el número de canales que contiene cada célula.
      wGruposZona,// Indica el número de grupos que conforman el sistema.
      wInformacion,// Indica las magnitudes a ver en la ventana derecha.
      wRadio,// Indica el radio de las células.
      wTiempo;// Indica el tiempo en milisegundos que tardan en moverse los móviles.

// Funciones miembro de uso privado para la clase TMainWindow

//Crea las coordenadas a ocupar por las células de un grupo dado.
void OrigenCelulas (POINT ptOrigen, BYTE byGrupo);

//Crea las coordenadas a ocupar por los grupos del sistema celular.
void OrigenGrupos ();

//Dibuja la estación de base de una célula.
void DibujaBase (POINT ptOrigen, BOOL bActiva);

//Dibuja una célula.
void DibujaCelula (POINT ptOrigen, DWORD dwColorBorde, DWORD dwColorFondo);

//Dibuja un grupo de células.
void DibujaGrupo (BYTE byGrupo, DWORD dwColorBorde);

//Crea de forma aleatoria las coordenadas del móvil identificado por ptMovil.
void CreaPuntoAleatorio (POINT &ptMovil);

//Dibuja un móvil.
void DibujaMovil (POINT ptMovil, BYTE byMovil);

//Borra un móvil.
BorraMovil (POINT ptMovil, BYTE byMovil);

//Envía cuadros de mensaje sobre el móvil seleccionado a la ventana principal.
void MensajeMovil (BYTE byMensaje);

//Visualiza los datos referentes al móvil seleccionado.
void CaracteristicasMovil (BOOL bEnganchado, BOOL bMismaBase, BYTE byBase,
      BYTE byGrupo);

//Localiza al móvil cuyas coordenadas se encuentran en ptMovil
//y llama a las funciones necesarias para indicarselo al usuario.
void LocalizaMovil (POINT ptMovil);

//Calcula si la posición en la que va a dibujarse un móvil está ocupada.
BOOL PosicionOcupada (POINT ptMovil, BYTE byMovil);

//Desplaza un móvil.
void DesplazaMovil (POINT &ptActual, POINT ptFinal);

//Calcula la posición donde debe desplarse un móvil para acercarse a la posición final deseada.
void ActivaMoviles ();
```

```
        //Detiene a todos los móviles en la posición en la que se encuentran.
        void PausaMoviles ();

        //Permite que se desplacen los móviles si antes estaban detenidos.
        void ContinuaMoviles ();

        //Probabilidad de bloqueo de una llamada, fórmula B de Erlang.
        float Erlang_B (float fTrafico, WORD wCanales);

        //Calcula el canal utilizado por el móvil seleccionado.
        WORD CanalUtilizado (BYTE byCelulaActual);
};
```

4.2.1 Funciones de TMainWindow

TMainWindow(PTWindowsObject AParent, LPSTR ATitle)

Esta función es el constructor de la ventana principal y en ella se definirán las condiciones iniciales de dicha ventana.

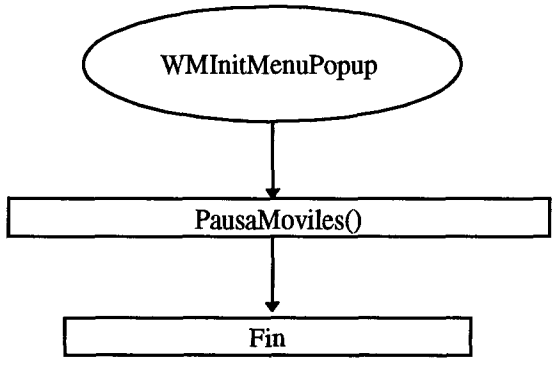


```
TMainWindow :: TMainWindow (PTWindowsObject AParent, LPSTR Atitle)
                        : TWindow (Aparent, ATitle)
{
        // Asigna el menú tmaMenu a la ventana principal.
        AssignMenu("tmaMenu");

        // Inicializa las variables globales que lo requieran.
        bCortaComunicacion = FALSE;
        bCursorDentro = FALSE;
        bIniciaLlamada = FALSE;
        bLlamadaEnCurso = FALSE;
        bLlamadaPerdida = FALSE;
        bLlamadaRecibida = FALSE;
        bMovilesActivados = FALSE;
        bYaEnganchado = FALSE;

        byCelulasGrupo = 4;
        byMovilSeleccionado = 2;

        fAreaCelula = 260;
        fBloqueo = 0;
        fMaximaRSI = 13.80;
        fMinimaRSI = 11.35;
        fTraficoCelula = 78;

        //Asignación dinámica de memoria.
        pOrigenCelulas = new POINT[byCelulasGrupo*7];
        if (pOrigenCelulas == NULL)
        {
                MessageBox(HWindow, "No existe suficiente memoria para ejecutar "
                                    "esta aplicación. Cierre otras aplicaciones "
                                    "y vuelva a ejecutar esta.",
                           "Mensaje de error", MB_OK | MB_ICONQUESTION);
                exit(1);
        }

        strcpy(sCanal, "0");
        strcpy(sCelula, "0");
        strcpy(sControl, "Automático");
        strcpy(sGrupo, "0");
        strcpy(sLlamada, "Finalizada");
        strcpy(sMovil, "Negro");

        wCanalesCelula = 250;
        wGruposZona = 10;
        wInformacion = CM_DIMENSIONES;
        wTiempo = 1000;
```

```
        // Inicializa los buffers de transferencia de datos entre la ventana principal y los cuadros de diálo-
go.
        // Primero se pone el buffer a cero y luego se introducen los datos.
        memset(&MovilBuffer, 0x0, sizeof (MovilBuffer));
        MovilBuffer.wMovimientoAleatorio = BF_CHECKED;
        MovilBuffer.wMovilNegro = BF_CHECKED;
        MovilBuffer.wMsgCanalBloqueado = BF_CHECKED;
        MovilBuffer.wMsgSalidaZona = BF_CHECKED;
        MovilBuffer.wVelocidadMedia = BF_CHECKED;
        MovilBuffer.wRecibirLlamadas = BF_CHECKED;

        memset(&GeomeBuffer, 0x0, sizeof (GeomeBuffer));
        strcpy(GeomeBuffer.sRadioCelula, "10");
        strcpy(GeomeBuffer.sAreaTotal, "10000");
        GeomeBuffer.wGrupo4Celulas = BF_CHECKED;

        memset(&RadioBuffer, 0x0, sizeof (RadioBuffer));
        strcpy(RadioBuffer.sAnchoBanda, "50");
        strcpy(RadioBuffer.sSeparacionCanales, "25");
        strcpy(RadioBuffer.sRelacionProteccion, "13");
        strcpy(RadioBuffer.sLeyPropagacion, "4");
        RadioBuffer.wAntenaSectorial120 = BF_CHECKED;

        memset(&TraficoBuffer, 0x0, sizeof (TraficoBuffer));
        strcpy(TraficoBuffer.sDensidadMoviles, "10");
}
```

void WMInitMenuPopup(RTMessage Msg)

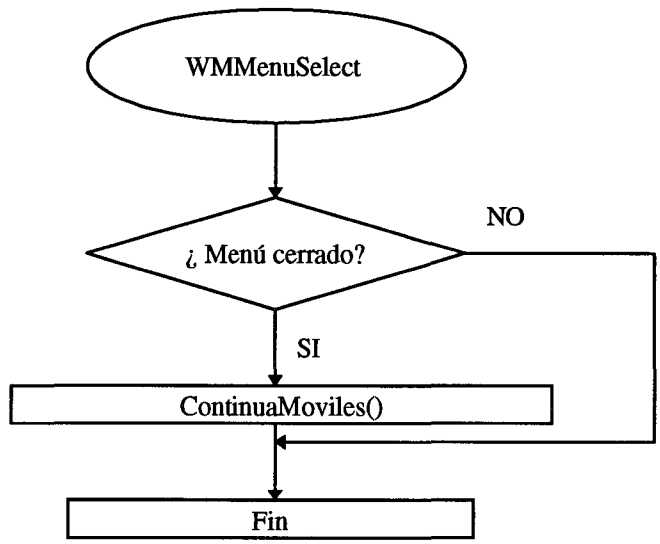
Esta función se activa antes de que se abra un menú desplegable. Se utiliza para que los móviles se paren si están visualizados en pantalla cuando se selecciona una opción del menú principal.



```
void TMainWindow :: WMInitMenuPopup (RTMessage Msg)
{
        PausaMoviles();
}
```

void WMMenuSelect(RTMessage Msg)

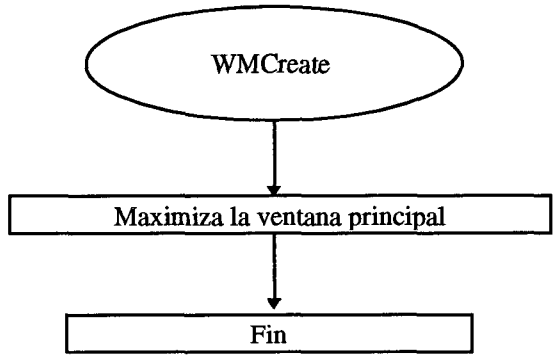
Esta función se activa cuando se selecciona una opción de un menú desplegable, o cuando se sale de él mediante la tecla "Esc". Se utiliza para Permitir que los móviles sigan en movimiento después de que se seleccione una opción de un menú desplegable.




```
void TMainWindow :: WMMenuSelect (RTMessage Msg)
{
        if (Msg.LP.Lo == 65535)
        // Se ha cerrado el menú desplegable.
                ContinuaMoviles();
}
```

void WMCreate(RTMessage Msg)

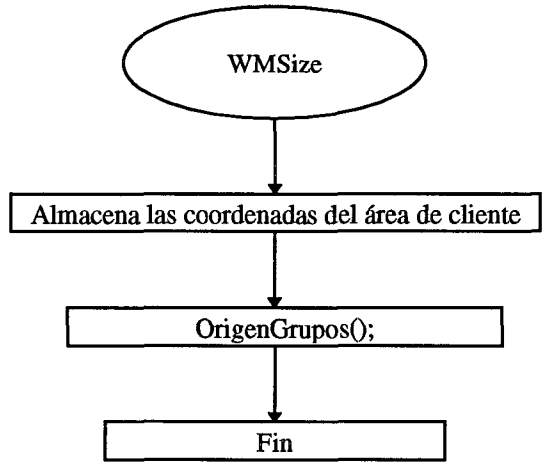
Esta función se activa al abrirse la ventana principal de la aplicación. Se utiliza para maximizar la ventana principal cuando es creada.




```
void TMainWindow :: WMCreate (RTMessage Msg)
{
        ShowWindow(HWindow, SW_MAXIMIZE);
}
```

void WMSize(RTMessage Msg)

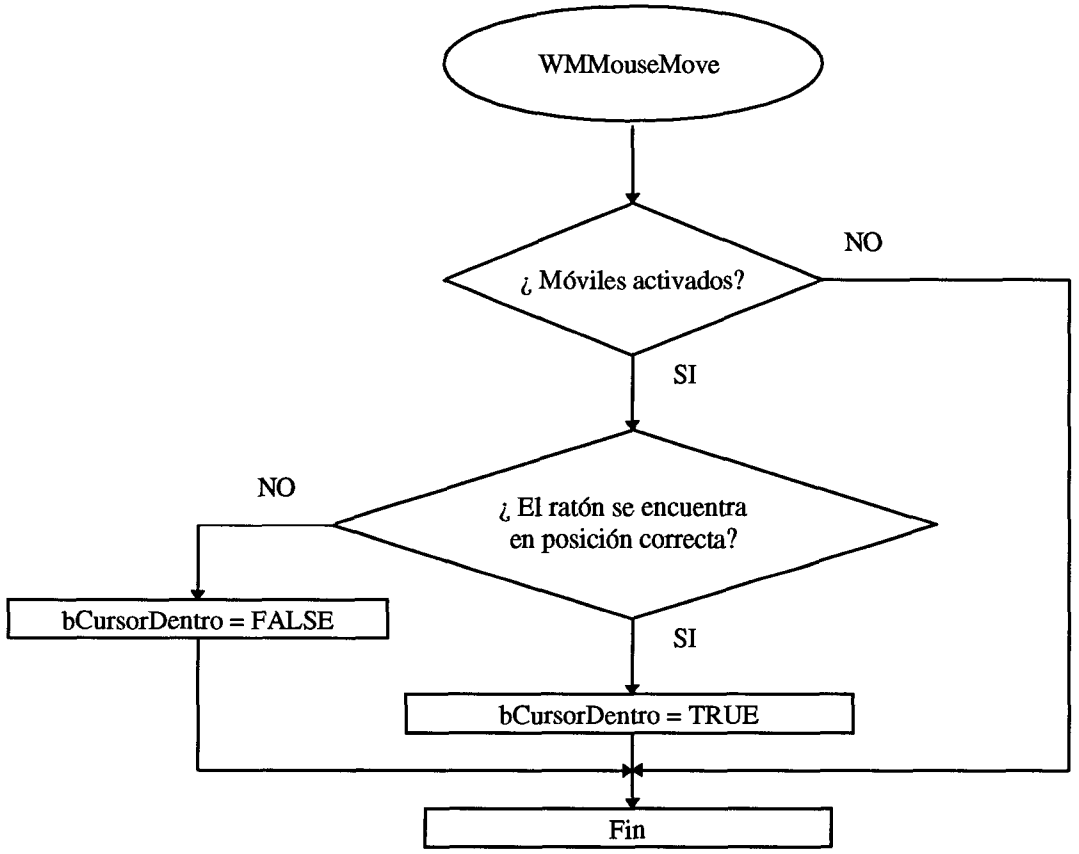
Esta función se activa cuando se redimensiona el tamaño de la ventana principal. Se utiliza para obtener las coordenadas del área de cliente de la aplicación.




```
void TMainWindow :: WMSize (RTMessage Msg)
{
        ptVentana = MAKEPOINT(Msg.LParam);
        OrigenGrupos();
}
```

void WMMouseMove(RTMessage Msg)

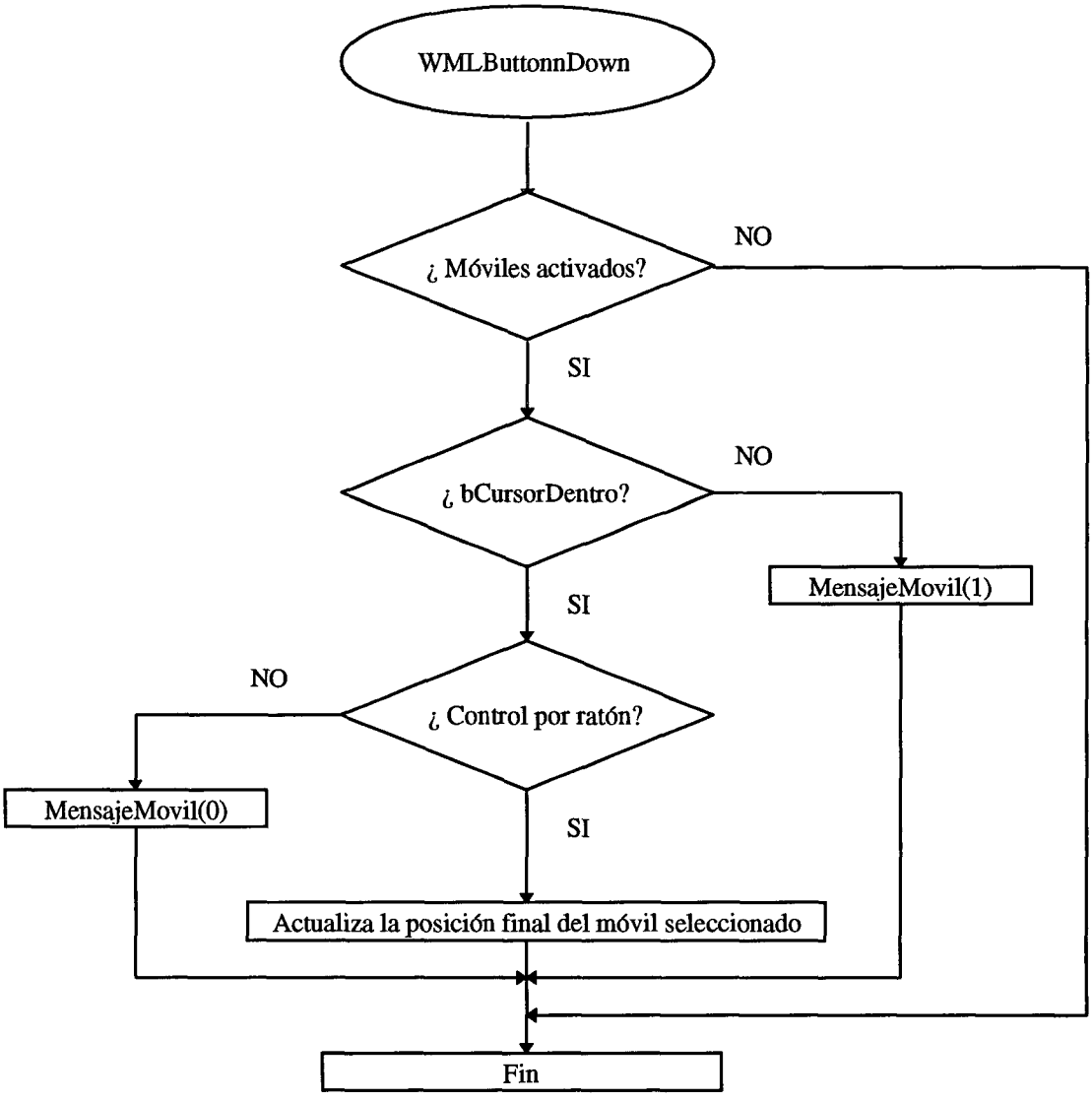
Esta función atiende al movimiento del ratón dentro de la ventana principal de la aplicación. Se utiliza para asegurarse que la posición a donde se quiere desplazar el móvil es correcta.




```
void TMainWindow :: WMMouseMove (RTMessage Msg)
{
        if (bMovilesActivados)
        {
                if ((Msg.LP.Lo >= ptVentana.x*0.6-10)||(Msg.LP.Hi >= ptVentana.y*0.8-10)
                        ||(Msg.LP.Lo < 10)||(Msg.LP.Hi < 10))
                // el cursor se encuentra fuera de la ventana superior izquierda.
                        bCursorDentro = FALSE;
                else
                        // el cursor se encuentra dentro de la ventana superior izquierda.
                        bCursorDentro = TRUE;
        }
}
```

void WMLButtonDown(RTMessage Msg)

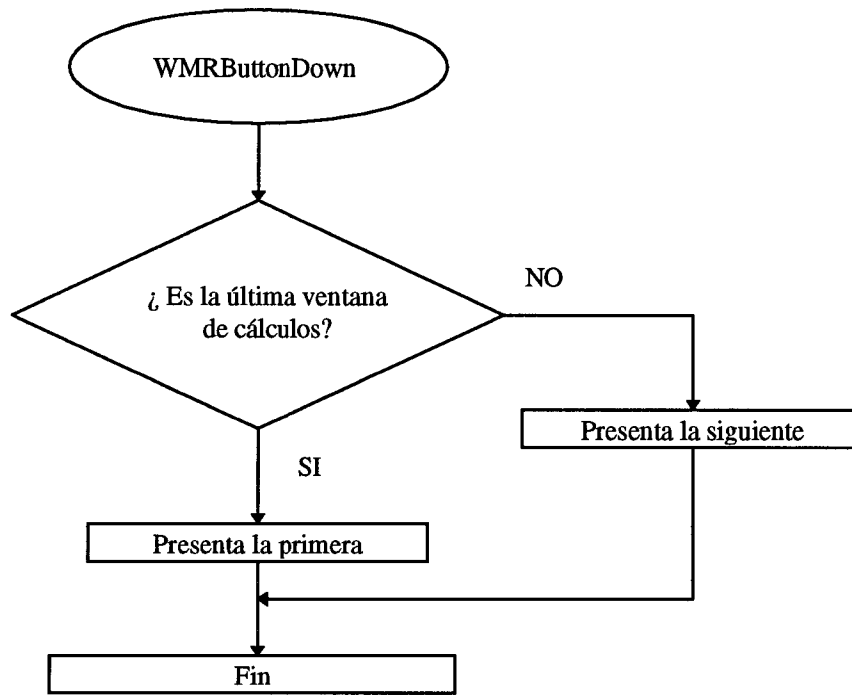
Esta función es llamada cuando se pulsa el botón izquierdo del ratón, siempre que se encuentre en el área de cliente de la ventana principal de la aplicación. Se utiliza para desplazar al móvil seleccionado a la posición del cursor del ratón.



```
void TMainWindow :: WMLButtonDown (RTMessage Msg)
{
        if (bMovilesActivados)
        {
                if (bCursorDentro)
                {
                        if (MovilBuffer.wMovimientoRaton == BF_CHECKED)
                                // móvil controlado por ratón.
                                ptFinal[byMovilSeleccionado] = MAKEPOINT(Msg.LParam);
                        else
                                // móvil controlado por la aplicación.
                                MensajeMovil(0);
                }
                else
                        MensajeMovil(1);
        }
}
```

void WMRButtonDown (RTMessage Msg)

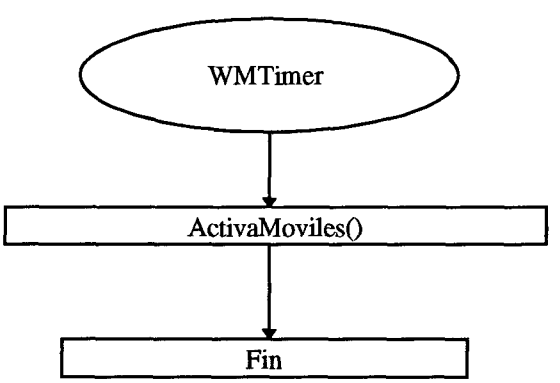
Esta función gestiona la pulsación del botón derecho del ratón cuando se encuentra en el área de cliente de la aplicación. Se utiliza para agilizar la presentación de las diferentes ventanas de cálculos del sistema TMA.



```
void TMainWindow :: WMRButtonDown (RTMessage Msg)
{
        if (wInformacion == CM_MOVILES)
                // la ventana derecha contiene "Cálculos sobre los móviles".
                SendMessage(HWindow, WM_COMMAND, CM_DIMENSIONES, 0);
        else
                SendMessage(HWindow, WM_COMMAND, wInformacion+1, 0);
}
```

void WMTimer (RTMessage Msg)

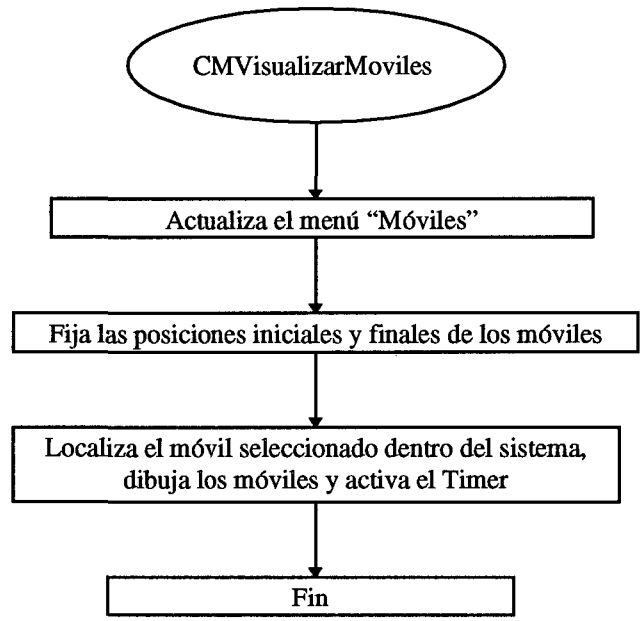
Esta función se activa al producirse un aviso de final de cuenta de un timer que puede ser programado. Se utiliza para desplazar a los móviles con una frecuencia regular que puede ser modificada.




```
void TMainWindow :: WMTimer (RTMessage Msg)
{
        ActivaMoviles();
}
```

void CMVisualizarMoviles (RTMessage Msg)

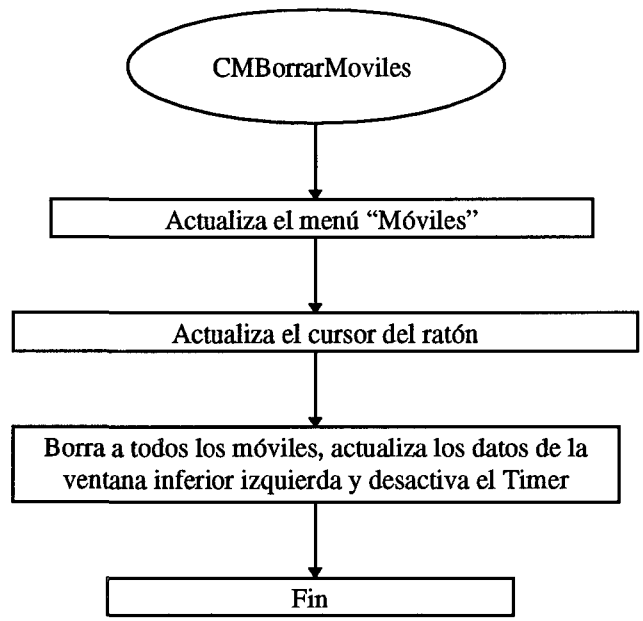
Esta función realiza los pasos necesarios para visualizar los móviles en pantalla.




```
void TMainWindow :: CMVisualizarMoviles (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        EnableMenuItem(hmenu, CM_VISUALIZARMOVILES, MF_GRAYED);
        EnableMenuItem(hmenu, CM_BORRARMOVILES, MF_ENABLED);
        EnableMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_ENABLED);
        EnableMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_GRAYED);
        CheckMenuItem(hmenu, CM_BORRARMOVILES, MF_UNCHECKED);
        // Crea las posiciones iniciales y finales de los móviles.
        for (BYTE by = 0; by < NumMoviles; by++)
        {
                CreaPuntoAleatorio(ptActual[by]);
                CreaPuntoAleatorio(ptFinal[by]);
        }
        // Localiza el móvil seleccionado, los dibuja a todos y actica el timer.
        LocalizaMovil(ptActual[byMovilSeleccionado]);
        for (by = 0; by < NumMoviles; by++)
                DibujaMovil(ptActual[by], by);
        bMovilesActivados = TRUE;
        SetTimer(HWindow, TIMER_MOVIL, wTiempo, NULL);
}
```

void CMBorrarMoviles (RTMessage Msg)

Esta función realiza todos los pasos necesarios para que se borren los móviles de la pantalla y se actualicen los datos correspondientes.

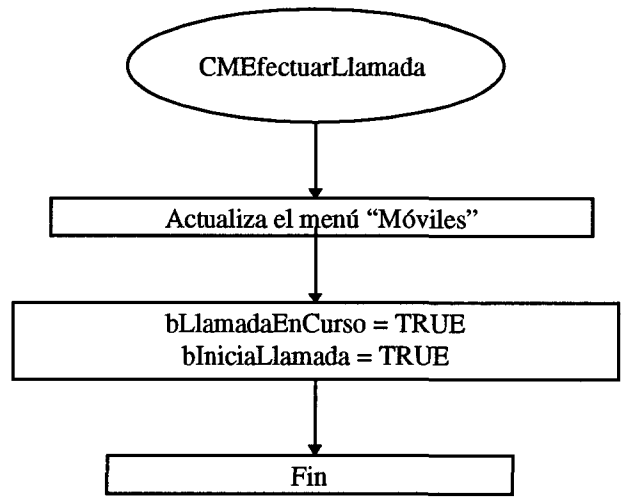



```
void TMainWindow :: CMBorrarMoviles (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        EnableMenuItem(hmenu, CM_VISUALIZARMOVILES, MF_ENABLED);
        EnableMenuItem(hmenu, CM_BORRARMOVILES, MF_GRAYED);
        EnableMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_GRAYED);
        EnableMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_GRAYED);
        EnableMenuItem(hmenu, CM_SELECCIONARMOVIL, MF_ENABLED);
        CheckMenuItem(hmenu, CM_VISUALIZARMOVILES, MF_UNCHECKED);
        CheckMenuItem(hmenu, CM_BORRARMOVILES, MF_CHECKED);
        CheckMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_UNCHECKED);
        CheckMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_CHECKED);
        // Actualiza el cursor y las ventanas de la izquierda.
        SetClassWord(HWindow, GCW_HCURSOR, LoadCursor(NULL, IDC_ARROW));
        for (BYTE by = 0; by < NumMoviles; by++)
                BorraMovil(ptActual[by], by);
        DibujaGrupo(byGrupoAnterior, RGB(0,0,0));
```

```
        bYaEnganchado = FALSE;
        bMovilesActivados = FALSE;
        KillTimer(HWindow, TIMER_MOVIL);
        bLlamadaEnCurso = FALSE;
        bLlamadaPerdida = FALSE;
        bCortaComunicacion = TRUE;
        strcpy(sLlamada, "Finalizada");
        strcpy(sCelula, "0");
        strcpy(sGrupo, "0");
        strcpy(sCanal, "0");
        RECT rect = {10, ptVentana.y*0.8+5, ptVentana.x*0.6-10, ptVentana.y-10};
        InvalidateRect(HWindow, &rect, TRUE);
}
```

void CMEfectuarLlamada (RTMessage Msg)

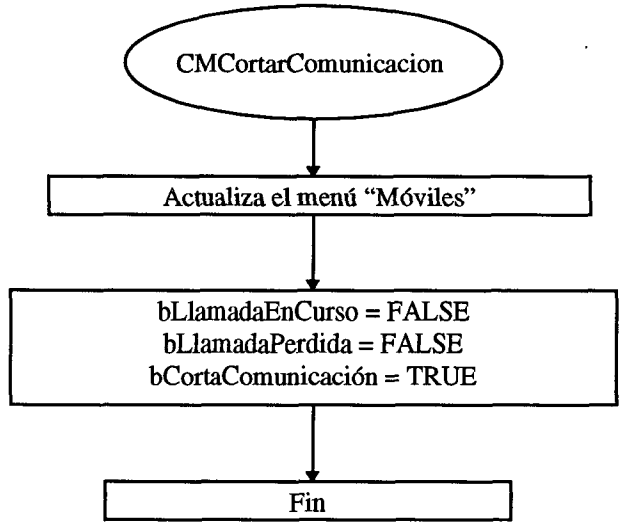
Con esta función conseguimos iniciar una comunicación telefónica para el móvil seleccionado.



```
void TMainWindow :: CMEfectuarLlamada (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        EnableMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_GRAYED);
        EnableMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_ENABLED);
        EnableMenuItem(hmenu, CM_SELECCIONARMOVIL, MF_GRAYED);
        CheckMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_CHECKED);
        CheckMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_UNCHECKED);
        bLlamadaEnCurso = TRUE;
        bIniciaLlamada = TRUE;
}
```

void CMCortarComunicacion (RTMessage Msg)

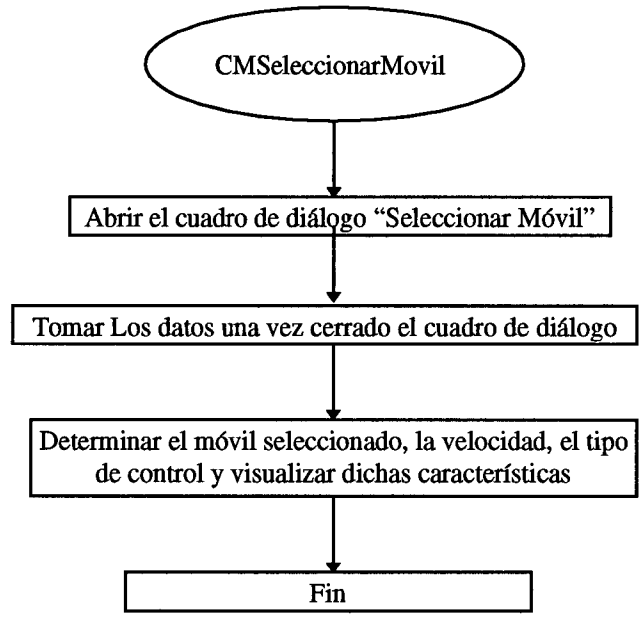
Esta función es la encargada de cortar la comunicación telefónica si la llamada está en curso, o también "colgar" el teléfono móvil si la llamada se ha perdido.



```
void TMainWindow :: CMCortarComunicacion (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        EnableMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_ENABLED);
        EnableMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_GRAYED);
        EnableMenuItem(hmenu, CM_SELECCIONARMOVIL, MF_ENABLED);
        CheckMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_UNCHECKED);
        CheckMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_CHECKED);
        bLlamadaEnCurso = FALSE;
        bLlamadaPerdida = FALSE;
        bCortaComunicacion = TRUE;
}
```

void CMSeleccionarMovil (RTMessage Msg)

Esta función es la encargada de abrir el cuadro de diálogo "Seleccionar móvil...", así como obtener los datos que se han introducido para actualizarlos en la aplicación.

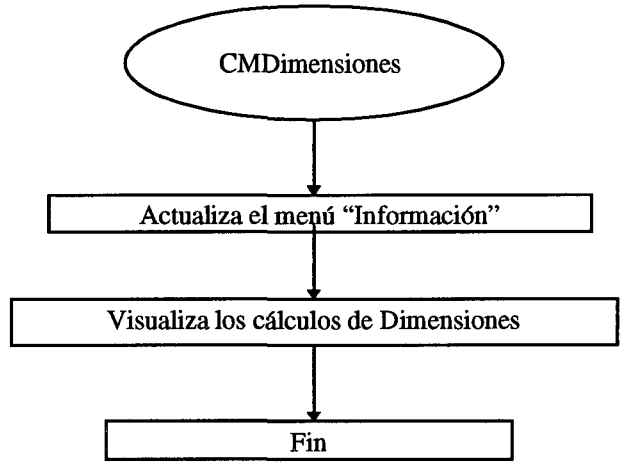


```
void TMainWindow :: CMSeleccionarMovil (RTMessage Msg)
{
        PausaMoviles();
        GetModule()->ExecDialog(new MovilDialog(this, "movilDlg"));
        // Identifica al móvil seleccionado.
        if (MovilBuffer.wMovilAmarillo == BF_CHECKED)
        {
                byMovilSeleccionado = 0;
                strcpy(sMovil, "Amarillo");
        }
        else
        {
                if (MovilBuffer.wMovilAzul == BF_CHECKED)
                {
                        byMovilSeleccionado = 1;
                        strcpy(sMovil, "Azul");
                }
                else
                {
                        if (MovilBuffer.wMovilNegro == BF_CHECKED)
                        {
                                byMovilSeleccionado = 2;
                                strcpy(sMovil, "Negro");
                        }
                        else
                        {
                                if (MovilBuffer.wMovilRojo == BF_CHECKED)
                                {
                                        byMovilSeleccionado = 3;
                                        strcpy(sMovil, "Rojo");
                                }
                                else
                                {
                                        byMovilSeleccionado = 4;
                                        strcpy(sMovil, "Verde");
                                }
                        }
                }
        }
        // Determina la velocidad de los móviles.
        if (MovilBuffer.wVelocidadMaxima == BF_CHECKED)
                wTiempo = 500;
        else
        {
                if (MovilBuffer.wVelocidadMedia == BF_CHECKED)
                        wTiempo = 1000;
                else
                        wTiempo = 2000;
        }
```

```
        // Determina el movimiento del móvil seleccionado.
        if ((MovilBuffer.wMovimientoRaton == BF_CHECKED))
                strcpy(sControl, "Ratón");
        else
                strcpy(sControl, "Automático");
        // Visualiza las características del móvil seleccionado.
        BYTE byEspacioFilas = (ptVentana.y*0.2-20)/3;
        BYTE byEspacioColum = (ptVentana.x*0.6-30)/4;
        HDC hdc = GetDC(HWindow);
        HBRUSH hbrush = SelectObject(hdc, GetStockObject(GRAY_BRUSH));
        HPEN hpen = SelectObject(hdc, CreatePen(PS_SOLID, 1, RGB(128,128,128)));
        Rectangle(hdc, 20+byEspacioColum, ptVentana.y*0.8+10,19+byEspacioColum*2,
                ptVentana.y*0.8+10+byEspacioFilas);
        Rectangle(hdc, 20+byEspacioColum, ptVentana.y*0.8+10+byEspacioFilas,
19+byEspacioColum*2,
                ptVentana.y*0.8+10+byEspacioFilas*2);
        DeleteObject(SelectObject(hdc, hbrush));
        DeleteObject(SelectObject(hdc, hpen));
        DWORD dwMovil[NumMoviles] =
        { RGB(255,255,0), RGB(0,0,255), RGB(0,0,0), RGB(191,0,0), RGB(0,255,0) };
        SetTextColor(hdc, dwMovil[byMovilSeleccionado]);
        SetBkMode(hdc, TRANSPARENT);
        TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10, sMovil, strlen(sMovil));
        TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10+byEspacioFilas, sControl,
                strlen(sControl));
        TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10+byEspacioFilas*2, sLlamada,
                strlen(sLlamada));
        TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10, sCelula, strlen(sCelula));
        TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10+byEspacioFilas, sGrupo,
                strlen(sGrupo));
        TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10+byEspacioFilas*2, sCanal,
                strlen(sCanal));
        ReleaseDC(HWindow, hdc);
        ContinuaMoviles();
}
```

void CMDimensiones (RTMessage Msg)

Esta función se activa al seleccionar la opción "Cálculos de Dimensiones" del menú "Información". Visualiza en pantalla los cálculos obtenidos.



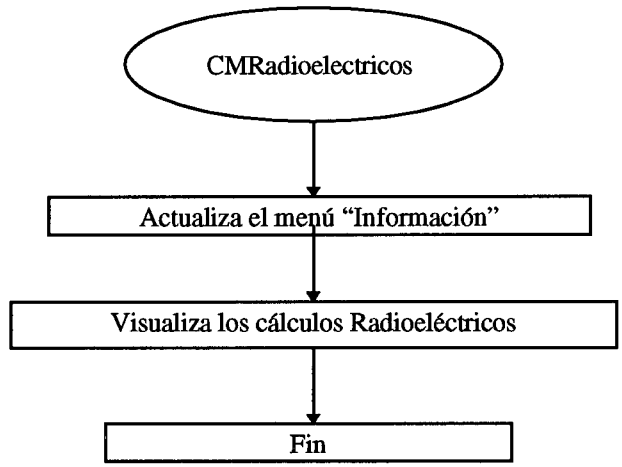

```
void TMainWindow :: CMDimensiones (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        CheckMenuItem(hmenu, wInformacion, MF_UNCHECKED);
        wInformacion = Msg.WParam;
        CheckMenuItem(hmenu, wInformacion, MF_CHECKED);
        // Visualiza la ventana derecha.
        HDC hdc = GetDC(HWindow);
        char sTexto[50];
        char sValor[15];
        HBRUSH hbrush = SelectObject(hdc, CreateSolidBrush(RGB(128,128,0)));
        Rectangle(hdc, ptVentana.x*0.6, 5, ptVentana.x-5, ptVentana.y-5);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, (ptVentana.x*0.6)+5, 10, ptVentana.x-10, ptVentana.y-10);
        DeleteObject(SelectObject(hdc, hbrush));
        // Visualiza los cálculos de dimensiones.
        SetBkMode( hdc, TRANSPARENT);
        BYTE byEspacioFilas = (ptVentana.y-20)/21;
        for (BYTE by = 0; by < 20; by++)
        {
                switch (by)
                {
                case 0:
                        strcpy(sTexto, "    CALCULOS DE DIMENSIONES");
                        SetTextColor(hdc, RGB(128,128,0));
                        break;
                case 1:
                        strcpy(sTexto, "EN CADA CELULA:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 2:
                        strcpy(sTexto, "Area de cobertura:");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
```

```
case 3:
        strcpy(sTexto, "Sc = ");
        gcvt(fAreaCelula, NumChar, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " Km²");
        SetTextColor(hdc,RGB(0,128,128));
        break;
case 4:
        strcpy(sTexto, "Radio de cobertura:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 5:
        strcpy(sTexto, "R = ");
        strcat(sTexto, GeomeBuffer.sRadioCelula);
        strcat(sTexto, " Km");
        SetTextColor(hdc,RGB(0,128,128));
        break;
case 6:
        strcpy(sTexto, "Altura de la estación de base:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 7:
        strcpy(sTexto, "hb = ");
        gcvt(pow(atof(GeomeBuffer.sRadioCelula)/4.1, 2), 4, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " m");
        SetTextColor(hdc,RGB(0,128,128));
        break;
case 8:
        strcpy(sTexto, "EN CADA GRUPO:");
        SetTextColor(hdc, RGB(0,0,255));
        break;
case 9:
        strcpy(sTexto, "Area de cobertura:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 10:
        strcpy(sTexto, "Sg = ");
        gcvt(fAreaCelula*byCelulasGrupo, NumChar, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " Km²");
        SetTextColor(hdc,RGB(0,128,128));
        break;
case 11:
        strcpy(sTexto, "Número de células:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
```

```
                case 12:
                        strcpy(sTexto, "j = ");
                        gcvt(byCelulasGrupo, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " células");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 13:
                        strcpy(sTexto, "EN EL SISTEMA CELULAR:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 14:
                        strcpy(sTexto, "Area de cobertura:");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 15:
                        strcpy(sTexto, "Ss = ");
                        gcvt(fAreaCelula*byCelulasGrupo*wGruposZona, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " km²");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 16:
                        strcpy(sTexto, "Número de grupos:");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 17:
                        strcpy(sTexto, "Q = ");
                        gcvt(wGruposZona, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " grupos");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 18:
                        strcpy(sTexto, "Número de células:");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 19:
                        strcpy(sTexto, "N = ");
                        gcvt(wGruposZona*byCelulasGrupo, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " células");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                }
                TextOut(hdc, (ptVentana.x*0.6)+15, byEspacioFilas*(by+1), sTexto, strlen(sTexto));
        }
        ReleaseDC(HWindow, hdc);
        ContinuaMoviles();
}
```

void CMRadioelectricos (RTMessage Msg)

Esta función se activa al seleccionar la opción "Cálculos Radioeléctricos" del menú "Información". Visualiza en pantalla los cálculos obtenidos.



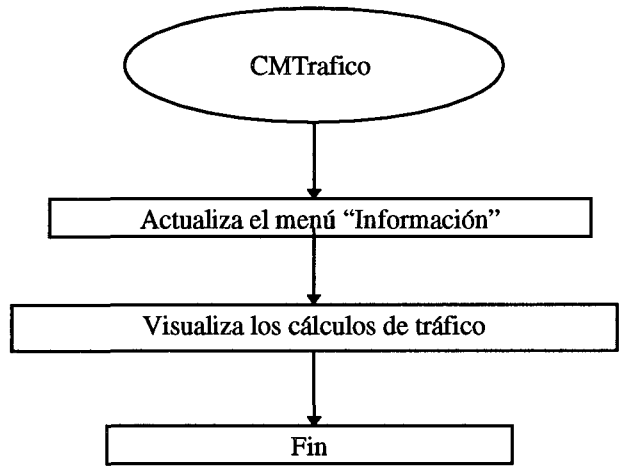

```
void TMainWindow :: CMRadioelectricos (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        CheckMenuItem(hmenu, wInformacion, MF_UNCHECKED);
        wInformacion = Msg.WParam;
        CheckMenuItem(hmenu, wInformacion, MF_CHECKED);
        // Visualiza la ventana derecha.
        HDC hdc = GetDC(HWindow);
        char sTexto[50];
        char sValor[15];
        float fRCocanal = sqrt(3*byCelulasGrupo);
        HBRUSH hbrush = SelectObject(hdc, CreateSolidBrush(RGB(0,128,0)));
        Rectangle(hdc, ptVentana.x*0.6, 5, ptVentana.x-5, ptVentana.y-5);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, (ptVentana.x*0.6)+5, 10, ptVentana.x-10, ptVentana.y-10);
        DeleteObject(SelectObject(hdc, hbrush));

        // Visualiza los cálculos radioeléctricos.
        SetBkMode( hdc, TRANSPARENT);
        BYTE byEspacioFilas = (ptVentana.y-20)/16;
```

```
for (BYTE by = 0; by < 15; by++)
{
        switch (by)
        {
        case 0:
                strcpy(sTexto, "   CALCULOS RADIOELECTRICOS");
                SetTextColor(hdc, RGB(0,128,0));
                break;
        case 1:
                strcpy(sTexto, "Reutilización Cocanal:");
                SetTextColor(hdc, RGB(0,0,0));
                break;
        case 2:
                strcpy(sTexto, "q = ");
                gcvt(fRCocanal, 4, sValor);
                strcat(sTexto, sValor);
                SetTextColor(hdc, RGB(0,128,128));
                break;
        case 3:
                strcpy(sTexto, "Distancia de Reutilización:");
                SetTextColor(hdc, RGB(0,0,0));
                break;
        case 4:
                strcpy(sTexto, "D = ");
                gcvt(atof(GeomeBuffer.sRadioCelula)*fRCocanal, 5, sValor);
                strcat(sTexto, sValor);
                strcat(sTexto, " km");
                SetTextColor(hdc, RGB(0,128,128));
                break;
        case 5:
                strcpy(sTexto, "Relación S / I mínima:");
                SetTextColor(hdc, RGB(0,0,0));
                break;
        case 6:
                strcpy(sTexto, "S / I mín = ");
                gcvt(fMinimaRSI, 5, sValor);
                strcat(sTexto, sValor);
                strcat(sTexto, " dB");
                SetTextColor(hdc, RGB(0,128,128));
                break;
        case 7:
                strcpy(sTexto, "Relación S / I máxima:");
                SetTextColor(hdc, RGB(0,0,0));
                break;
        case 8:
                strcpy(sTexto, "S / I máx = ");
                gcvt(fMaximaRSI, 5, sValor);
                strcat(sTexto, sValor);
                strcat(sTexto, " dB");
                SetTextColor(hdc, RGB(0,128,128));
                break;
```

```
                case 9:
                        strcpy(sTexto, "EN CADA CELULA:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 10:
                        strcpy(sTexto, "Cc = ");
                        gcvt(wCanalesCelula, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " canales");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 11:
                        strcpy(sTexto, "EN CADA GRUPO:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 12:
                        strcpy(sTexto, "Cg = ");
                        gcvt(wCanalesCelula*wGruposZona, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " canales");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 13:
                        strcpy(sTexto, "EN EL SISTEMA CELULAR:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 14:
                        strcpy(sTexto, "Cs = ");
                        gcvt(wCanalesCelula*wGruposZona*byCelulasGrupo, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " canales");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                }
                TextOut(hdc, (ptVentana.x*0.6)+15, byEspacioFilas*(by+1), sTexto, strlen(sTexto));
        }
        ReleaseDC(HWindow, hdc);
        ContinuaMoviles();
}
```

void CMTrafico (RTMessage Msg)

Esta función se activa al seleccionar la opción "Cálculos de tráfico" del menú "Información". Visualiza en pantalla los cálculos obtenidos.



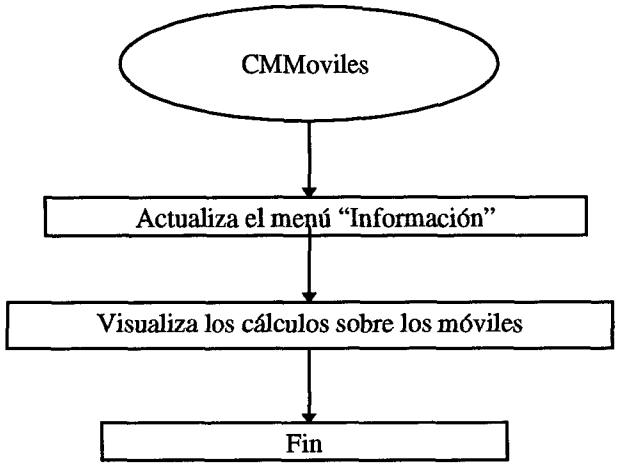

```
void TMainWindow :: CMTrafico (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        CheckMenuItem(hmenu, wInformacion, MF_UNCHECKED);
        wInformacion = Msg.WParam;
        CheckMenuItem(hmenu, wInformacion, MF_CHECKED);
        // Visualiza la ventana derecha.
        HDC hdc = GetDC(HWindow);
        char sTexto[50];
        char sValor[15];
        HBRUSH hbrush = SelectObject(hdc, CreateSolidBrush(RGB(255,0,0)));
        Rectangle(hdc, ptVentana.x*0.6, 5, ptVentana.x-5, ptVentana.y-5);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, (ptVentana.x*0.6)+5, 10, ptVentana.x-10, ptVentana.y-10);
        DeleteObject(SelectObject(hdc, hbrush));
        // Visualiza los cálculos de tráfico.
        SetBkMode( hdc, TRANSPARENT);
        BYTE byEspacioFilas = (ptVentana.y-20)/18;
        for (BYTE by = 0; by < 17; by++)
        {
                switch (by)
                {
                case 0:
                        strcpy(sTexto, "      CALCULOS DE TRAFICO");
                        SetTextColor(hdc, RGB(255,0,0));
                        break;
                case 1:
                        strcpy(sTexto, "Densidad de Trafico Ofrecido: ");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
```

```
case 2:
        strcpy(sTexto, "DT = ");
        gcvt(fTraficoCelula/fAreaCelula, 4, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " E/km²");
        SetTextColor(hdc, RGB(0,128,128));
        break;
case 3:
        strcpy(sTexto, "Densidad de Trafico Cursado: ");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 4:
        strcpy(sTexto, "DT' = ");
        gcvt((fTraficoCelula/fAreaCelula)*(1-fBloqueo), 4, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " E/km²");
        SetTextColor(hdc, RGB(0,128,128));
        break;
case 5:
        strcpy(sTexto, "EN CADA CANAL:");
        SetTextColor(hdc, RGB(0,0,255));
        break;
case 6:
        strcpy(sTexto, "Trafico Ofrecido, A = ");
        gcvt(fTraficoCelula/wCanalesCelula, 4, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " E");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 7:
        strcpy(sTexto, "Trafico Cursado, A' = ");
        gcvt((fTraficoCelula/wCanalesCelula)*(1-fBloqueo), 4, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " E");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 8:
        strcpy(sTexto, "EN CADA CELULA:");
        SetTextColor(hdc, RGB(0,0,255));
        break;
case 9:
        strcpy(sTexto, "Trafico Ofrecido, Ac = ");
        gcvt(fTraficoCelula, 5, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " E");
        SetTextColor(hdc, RGB(0,0,0));
        break;
```

```
                case 10:
                        strcpy(sTexto, "Trafico Cursado, Ac' = ");
                        gcvt((fTraficoCelula)*(1-fBloqueo), 5, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " E");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 11:
                        strcpy(sTexto, "EN CADA GRUPO:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 12:
                        strcpy(sTexto, "Trafico Ofrecido, Ag = ");
                        gcvt(fTraficoCelula*byCelulasGrupo, 5, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " E");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 13:
                        strcpy(sTexto, "Trafico Cursado, Ag' = ");
                        gcvt((fTraficoCelula*byCelulasGrupo)*(1-fBloqueo), 5, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " E");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 14:
                        strcpy(sTexto, "EN EL SISTEMA CELULAR:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 15:
                        strcpy(sTexto, "Trafico Ofrecido, As = ");
                        gcvt(fTraficoCelula*byCelulasGrupo*wGruposZona, 5, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " E");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 16:
                        strcpy(sTexto, "Trafico Cursado, As' = ");
                        gcvt((fTraficoCelula*byCelulasGrupo*wGruposZona)*(1-fBloqueo), 5, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " E");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                }
                TextOut(hdc, (ptVentana.x*0.6)+15, byEspacioFilas*(by+1), sTexto, strlen(sTexto));
        }
        ReleaseDC(HWindow, hdc);
        ContinuaMoviles();
}
```

void CMMoviles (RTMessage Msg)

Esta función se activa al seleccionar la opción "Cálculos sobre los móviles" del menú "Información". Visualiza en pantalla los cálculos obtenidos.



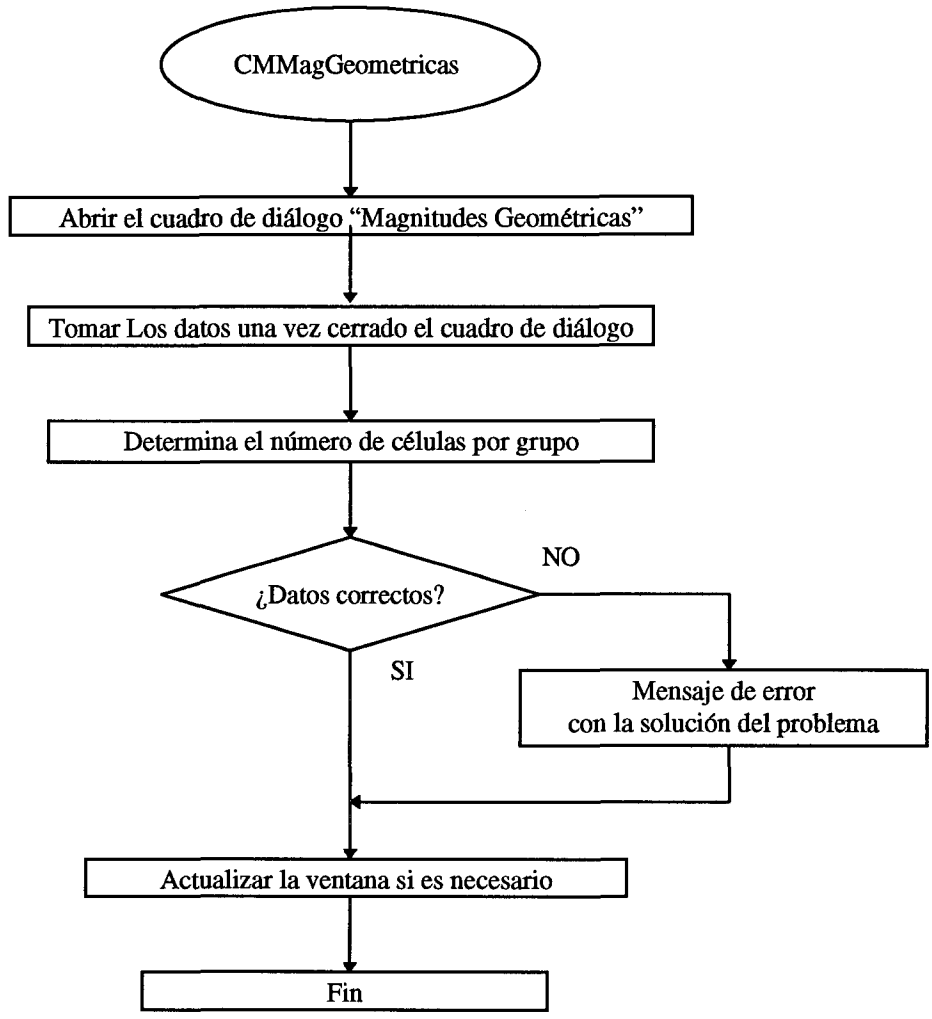

```
void TMainWindow :: CMMoviles (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        CheckMenuItem(hmenu, wInformacion, MF_UNCHECKED);
        wInformacion = Msg.WParam;
        CheckMenuItem(hmenu, wInformacion, MF_CHECKED);
        // Visualiza la ventana derecha.
        HDC hdc = GetDC(HWindow);
        char sTexto[50];
        char sValor[15];
        WORD wMovilesCelula = atof(TraficoBuffer.sDensidadMoviles)*fAreaCelula;
        HBRUSH hbrush = SelectObject(hdc, CreateSolidBrush(RGB(0,0,255)));
        Rectangle(hdc, ptVentana.x*0.6, 5, ptVentana.x-5, ptVentana.y-5);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, (ptVentana.x*0.6)+5, 10, ptVentana.x-10, ptVentana.y-10);
        DeleteObject(SelectObject(hdc, hbrush));
```

```
// Visualiza los cálculos sobre los móviles.
SetBkMode( hdc, TRANSPARENT);
BYTE byEspacioFilas = (ptVentana.y-20)/14;
for (BYTE by = 0; by < 13; by++)
{
        switch (by)
        {
        case 0:
                strcpy(sTexto, "  CALCULOS SOBRE LOS MOVILES");
                SetTextColor(hdc, RGB(0,0,255));
                break;
        case 1:
                strcpy(sTexto, "Probabilidad de Bloqueo:");
                SetTextColor(hdc, RGB(0,0,0));
                break;
        case 2:
                strcpy(sTexto, "p = ");
                gcvt(100*fBloqueo, 4, sValor);
                strcat(sTexto, sValor);
                strcat(sTexto, " %");
                SetTextColor(hdc, RGB(0,128,128));
                break;
        case 3:
                strcpy(sTexto, "Tiempo medio de llamada:");
                SetTextColor(hdc, RGB(0,0,0));
                break;
        case 4:
                strcpy(sTexto, "T = ");
                gcvt((fTraficoCelula*(1-fBloqueo)*60)/wMovilesCelula, 4, sValor);
                strcat(sTexto, sValor);
                strcat(sTexto, " minutos");
                SetTextColor(hdc, RGB(0,128,128));
                break;
        case 5:
                strcpy(sTexto, "Usuarios por cada canal:");
                SetTextColor(hdc, RGB(0,0,0));
                break;
        case 6:
                strcpy(sTexto, "M = ");
                gcvt(wMovilesCelula/wCanalesCelula, NumChar, sValor);
                strcat(sTexto, sValor);
                strcat(sTexto, " moviles/canal");
                SetTextColor(hdc, RGB(0,128,128));
                break;
        case 7:
                strcpy(sTexto, "EN CADA CELULA:");
                SetTextColor(hdc, RGB(0,0,255));
                break;
```

```
                case 8:
                        strcpy(sTexto, "Mc = ");
                        gcvt(wMovilesCelula, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " moviles");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 9:
                        strcpy(sTexto, "EN CADA GRUPO:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 10:
                        strcpy(sTexto, "Mg = ");
                        gcvt(wMovilesCelula*byCelulasGrupo, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " moviles");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 11:
                        strcpy(sTexto, "EN EL SISTEMA CELULAR:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 12:
                        strcpy(sTexto, "Ms = ");
                        gcvt(wMovilesCelula*byCelulasGrupo*wGruposZona, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " moviles");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                }
                TextOut(hdc, (ptVentana.x*0.6)+15, byEspacioFilas*(by+1), sTexto, strlen(sTexto));
        }
        ReleaseDC(HWindow, hdc);
        ContinuaMoviles();
}
```

void CMMagGeometricas (RTMessage Msg)

Esta función abre el cuadro de diálogo "Magnitudes Geométricas...". Obtiene los datos introducidos por el usuario y comprueba que sean correctos.



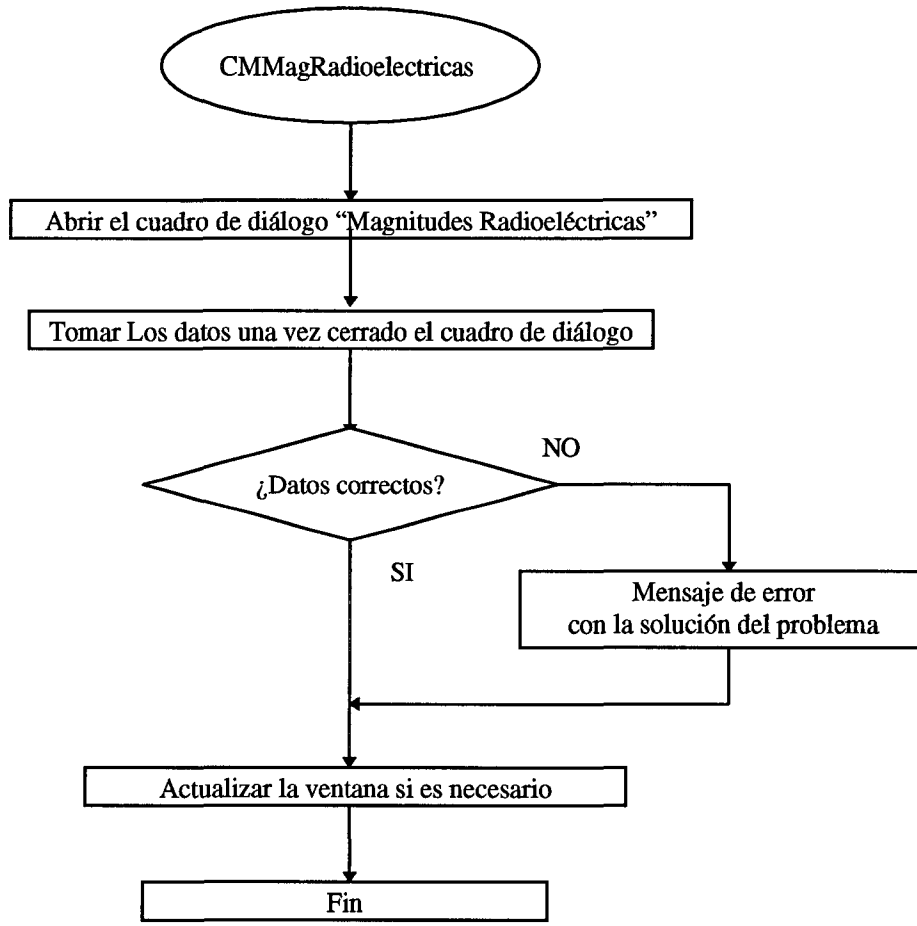

```
void TMainWindow :: CMMagGeometricas (RTMessage Msg)
{
        PausaMoviles();
        BYTE byCGAnterior = byCelulasGrupo;
        WORD wGZAnterior = wGruposZona;
        GetModule()->ExecDialog(new GeomeDialog(this, "geomeDlg"));
        // Identifica el número de células por grupo.
        if (GeomeBuffer.wGrupo3Celulas == BF_CHECKED)
                byCelulasGrupo = 3;
        else
        {
                if (GeomeBuffer.wGrupo4Celulas == BF_CHECKED)
                        byCelulasGrupo = 4;
                else
                {
                        if (GeomeBuffer.wGrupo7Celulas == BF_CHECKED)
                                byCelulasGrupo = 7;
```

```
                else
                        byCelulasGrupo = 12;
        }
}
// Realiza los cálculos necesarios.
wCanalesCelula =
500*atof(RadioBuffer.sAnchoBanda)/(atof(RadioBuffer.sSeparacionCanales)*byCelulasGrupo);
fAreaCelula = 2.6*pow(atof(GeomeBuffer.sRadioCelula),2);
fTraficoCelula = atof(TraficoBuffer.sDensidadMoviles)*TraficoMovil*fAreaCelula;
fBloqueo = Erlang_B(fTraficoCelula, wCanalesCelula);
float fGruposZona = atof(GeomeBuffer.sAreaTotal)/(byCelulasGrupo*fAreaCelula);
wGruposZona = ceil(fGruposZona);
// Comprueba que los datos no se contradigan.
if (fGruposZona < 1)
{
        char sMsg[150] = {"El área ocupada por un grupo es mayor "
                          "que el área total a cubrir. SOLUCION:\n"
                         };
        if (atof(GeomeBuffer.sRadioCelula) > 1)
                strcat(sMsg, "\nDisminuir el radio de las células.");
        else
                strcat(sMsg, "\nDisminuir el número de células por grupo.");
        if (MessageBox(HWindow, sMsg, "Mensaje de advertencia", MB_YESNO |
                MB_ICONEXCLAMATION) == IDYES)
                SendMessage(HWindow, WM_COMMAND, CM_MAGGEOMETRICAS, 0);
}
float fRCocanal = sqrt(3*byCelulasGrupo);
if (RadioBuffer.wAntenaOmnidireccional == BF_CHECKED)
{
        fMinimaRSI = 10*log10(1/((2/pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)))+
                (2/pow(fRCocanal-1, atof(RadioBuffer.sLeyPropagacion)))+
                (2/pow(fRCocanal+1, atof(RadioBuffer.sLeyPropagacion)))));
        fMaximaRSI = 10*log10(pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion))/6);
        if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                if (MessageBox(HWindow, "La relación Señal / Interferencia del sistema "
                                        "es menor que la especificada. SOLUCION:\n"
                                        "\nUtilizar antenas sectoriales de 120°.",
                                        "Mensaje de advertencia", MB_YESNO |
                                        MB_ICONEXCLAMATION) == IDYES)
                        SendMessage(HWindow, WM_COMMAND,
                                CM_MAGRADIOELECTRICAS, 0);
}
```

```
        else
        {
                if (RadioBuffer.wAntenaSectorial120 == BF_CHECKED)
                {
                        fMinimaRSI = 10*log10(pow(fRCocanal,
                                atof(RadioBuffer.sLeyPropagacion))/2);
                        fMaximaRSI = 10*log10(1/((1/pow(fRCocanal+0.7,
                                atof(RadioBuffer.sLeyPropagacion)))+
                                (1/pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)))));
                        if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                                if (MessageBox(HWindow,"La relación Señal/Interferencia del sistema "
                                                "es menor que la especificada.  SOLUCION:\n"
                                                "\nUtilizar antenas sectoriales de 60º.",
                                                "Mensaje de advertencia", MB_YESNO |
                                                MB_ICONEXCLAMATION) == IDYES)
                                SendMessage(HWindow, WM_COMMAND,
                                        CM_MAGRADIOELECTRICAS, 0);
                }
                else
                {
                        fMinimaRSI = 10*log10(pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)));
                        fMaximaRSI = 10*log10(pow(fRCocanal+0.7,
                                atof(RadioBuffer.sLeyPropagacion)));
                        if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                                if (MessageBox(HWindow,"La relación Señal/Interferencia del sistema "
                                                "es menor que la especificada.  SOLUCION:\n"
                                                "\nAumentar el número de células por grupo.",
                                                "Mensaje de advertencia", MB_YESNO |
                                        MB_ICONEXCLAMATION) == IDYES)
                                        SendMessage(HWindow, WM_COMMAND,
                                                CM_MAGGEOMETRICAS, 0);
                }
        }
        // Actualiza la ventana superior izquierda si es necesario.
        if ((byCGAnterior != byCelulasGrupo)||(wGZAnterior != wGruposZona))
        {
                delete pOrigenCelulas;
                BYTE byGrupos = (wGruposZona > 7)? 7: wGruposZona;
                pOrigenCelulas = new POINT[byCelulasGrupo*byGrupos];
                if (pOrigenCelulas == NULL)
                {
                        MessageBox(HWindow, "No existe suficiente memoria para ejecutar "
                                                "esta aplicación. Cierre otras aplicaciones "
                                                "y vuelva a ejecutar esta.",
                                        "Mensaje de error", MB_OK | MB_ICONQUESTION);
                        exit(1);
                }
                OrigenGrupos();
                InvalidateRect(HWindow, NULL, TRUE);
        }
        SendMessage(HWindow, WM_COMMAND, wInformacion, 0);
        ContinuaMoviles();
}
```

void CMMagRadioelectricas (RTMessage Msg)

Esta función abre el cuadro de diálogo "Magnitudes Radioeléctricas...". Obtiene los datos introducidos por el usuario y comprueba que sean correctos.

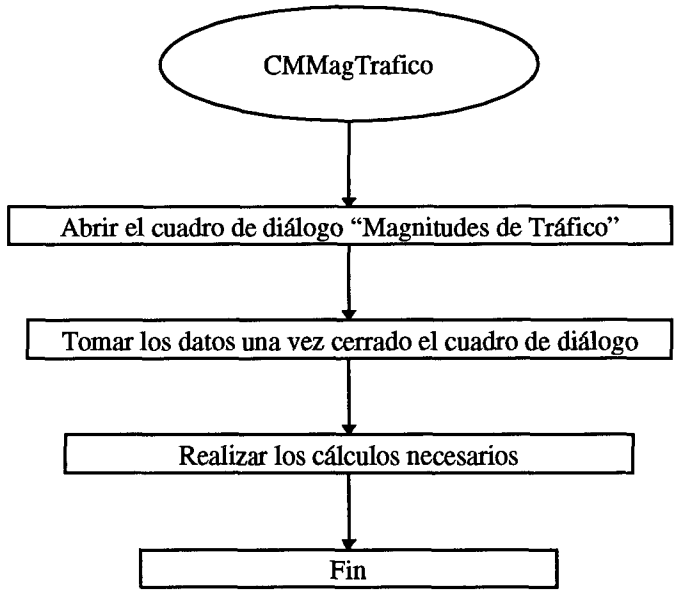



```
void TMainWindow :: CMMagRadioelectricas (RTMessage Msg)
{
        PausaMoviles();
        GetModule()->ExecDialog(new RadioDialog(this, "radioDlg"));
        // Realiza los cálculos necesarios.
        wCanalesCelula =
        500*atof(RadioBuffer.sAnchoBanda)/(atof(RadioBuffer.sSeparacionCanales)*byCelulasGrupo);
        fBloqueo = Erlang_B(fTraficoCelula, wCanalesCelula);
        float fRCocanal = sqrt(3*byCelulasGrupo);
```

```
        // Comprueba que los datos no se contradigan.
        if (RadioBuffer.wAntenaOmnidireccional == BF_CHECKED)
        {
                fMinimaRSI = 10*log10(1/((2/pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)))+
                                (2/pow(fRCocanal-1, atof(RadioBuffer.sLeyPropagacion)))+
                                (2/pow(fRCocanal+1, atof(RadioBuffer.sLeyPropagacion)))));
                fMaximaRSI = 10*log10(pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion))/6);
                if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                        if (MessageBox(HWindow, "La relación Señal / Interferencia del sistema "
                                                "es menor que la especificada. SOLUCION:\n"
                                                "\nUtilizar antenas sectoriales de 120°.",
                                                "Mensaje de advertencia", MB_YESNO |
                                                MB_ICONEXCLAMATION) == IDYES)
                                SendMessage(HWindow, WM_COMMAND,
                                        CM_MAGRADIOELECTRICAS, 0);
        }
        else
        {
                if (RadioBuffer.wAntenaSectorial120 == BF_CHECKED)
                {
                        fMinimaRSI = 10*log10(pow(fRCocanal,
                                atof(RadioBuffer.sLeyPropagacion))/2);
                        fMaximaRSI = 10*log10(1/((1/pow(fRCocanal+0.7,
                                atof(RadioBuffer.sLeyPropagacion)))+
                                (1/pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)))));
                        if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                                if (MessageBox(HWindow,"La relación Señal/Interferencia del sistema "
                                                        "es menor que la especificada.SOLUCION:\n"
                                                        "\nUtilizar antenas sectoriales de 60°.",
                                                        "Mensaje de advertencia", MB_YESNO |
                                                        MB_ICONEXCLAMATION) == IDYES)
                                        SendMessage(HWindow, WM_COMMAND,
                                                CM_MAGRADIOELECTRICAS, 0);
                }
                else
                {
                        fMinimaRSI = 10*log10(pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)));
                        fMaximaRSI = 10*log10(pow(fRCocanal+0.7,
                                atof(RadioBuffer.sLeyPropagacion)));
                        if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                                if (MessageBox(HWindow,"La relación Señal/Interferencia del sistema "
                                                        "es menor que la especificada.  SOLUCION:\n"
                                                        "\nAumentar el número de células por grupo.",
                                                        "Mensaje de advertencia", MB_YESNO |
                                                        MB_ICONEXCLAMATION) == IDYES)
                                        SendMessage(HWindow, WM_COMMAND,
                                                        CM_MAGGEOMETRICAS, 0);
                }
        }
        SendMessage(HWindow, WM_COMMAND, wInformacion, 0);
        ContinuaMoviles();
}
```

void CMMagTrafico (RTMessage Msg)

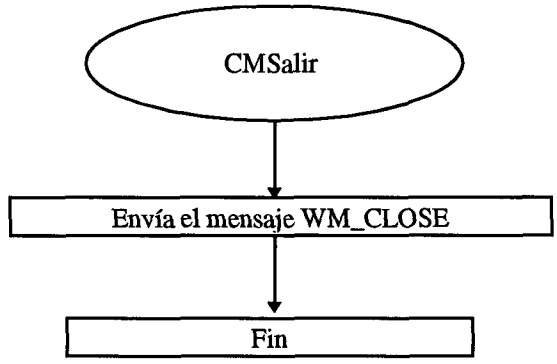
Esta función abre el cuadro de diálogo "Magnitudes de Tráfico...". Obtiene los datos introducidos por el usuario y los actualiza en la aplicación.




```
void TMainWindow :: CMMagTrafico (RTMessage Msg)
{
        PausaMoviles();
        GetModule()->ExecDialog(new TraficoDialog(this, "traficoDlg"));
        // Realiza los cálculos necesarios.
        fTraficoCelula = atof(TraficoBuffer.sDensidadMoviles)*TraficoMovil*fAreaCelula;
        fBloqueo = Erlang_B(fTraficoCelula, wCanalesCelula);
        SendMessage(HWindow, WM_COMMAND, wInformacion, 0);
        ContinuaMoviles();
}
```

void CMSalir (RTMessage Msg)

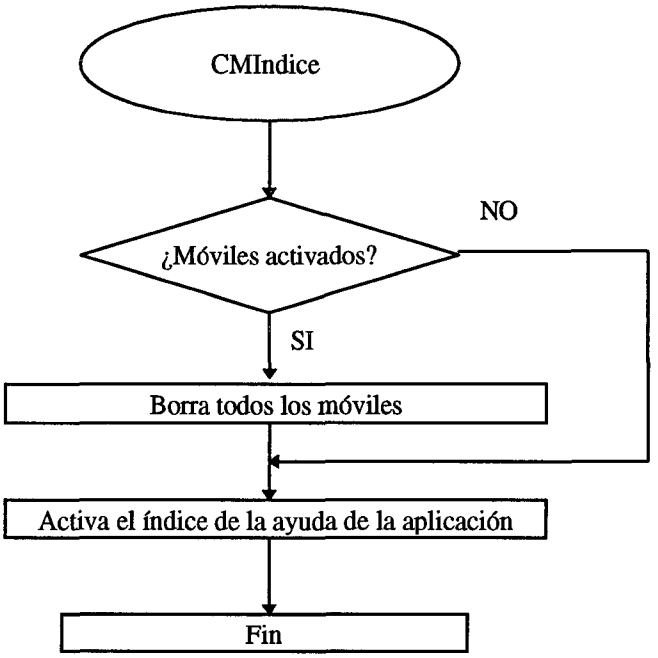
Esta función envía el mensaje de cerrar la ventana principal para salir de la aplicación.




```
void TMainWindow :: CMSalir (RTMessage Msg)
{
        SendMessage(HWindow, WM_CLOSE, 0, 0);
}
```

void CMIndice (RTMessage Msg)

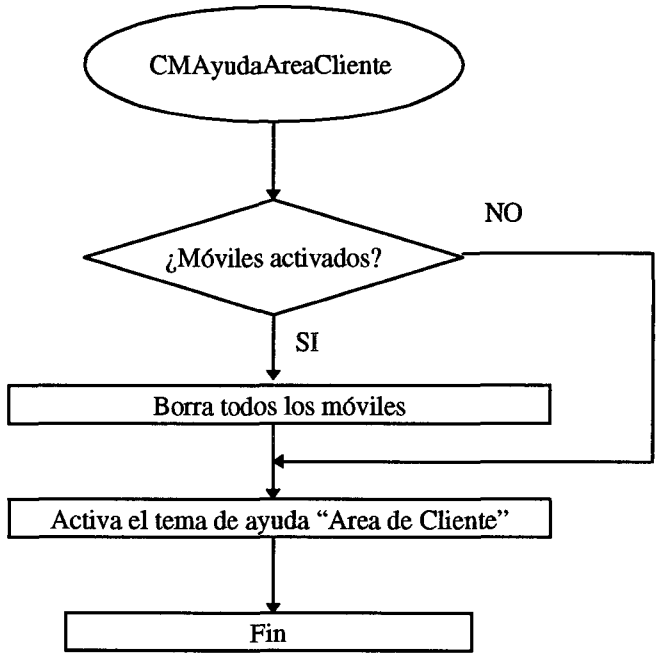
Esta función permite acceder al índice de la ayuda de esta aplicación.




```
void TMainWindow :: CMIndice (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_INDEX, 0);
}
```

void CMAyudaAreaCliente (RTMessage Msg)

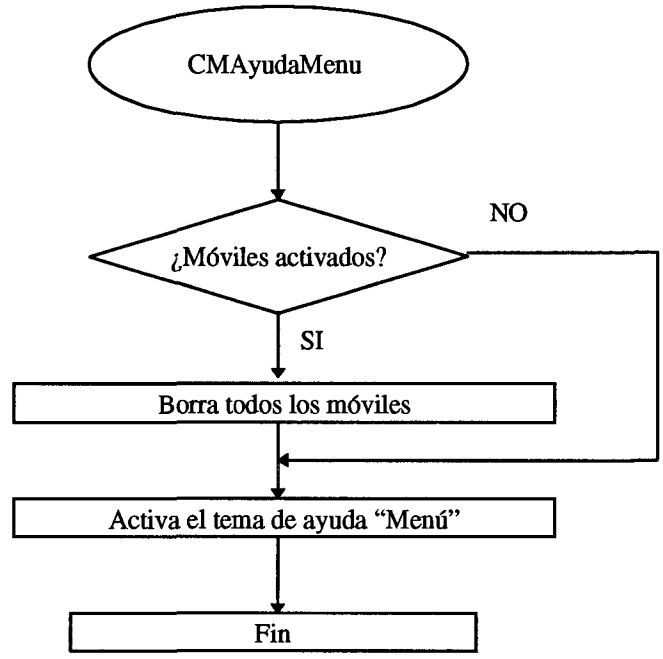
Esta función permite acceder al tema de la ayuda de la aplicación relativo al área de cliente de la aplicación.




```
void TMainWindow :: CMAyudaAreaCliente (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_KEY, (LONG)(LPSTR) "Area de Cliente");
}
```

void CMAyudaMenu (RTMessage Msg)

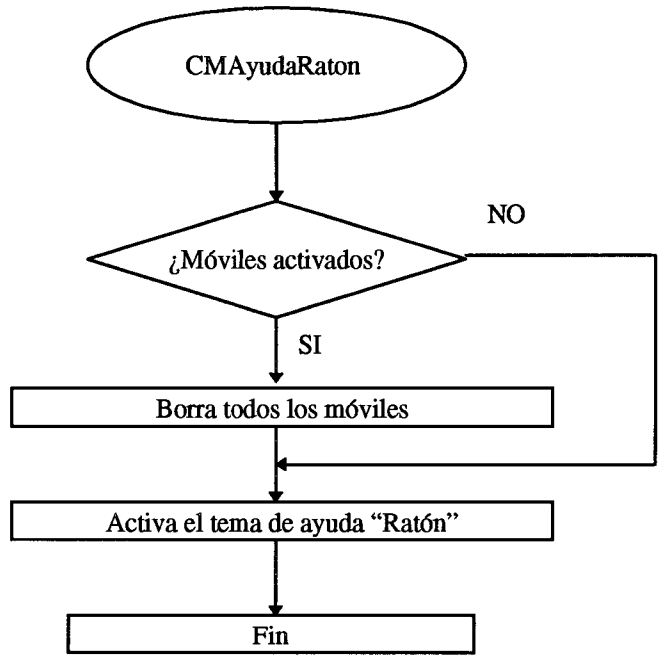
Esta función permite acceder al tema de la ayuda de la aplicación relativo al menú de la aplicación.




```
void TMainWindow :: CMAyudaMenu (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_KEY, (LONG)(LPSTR) "Menú");
}
```

void CMAyudaRaton (RTMessage Msg)

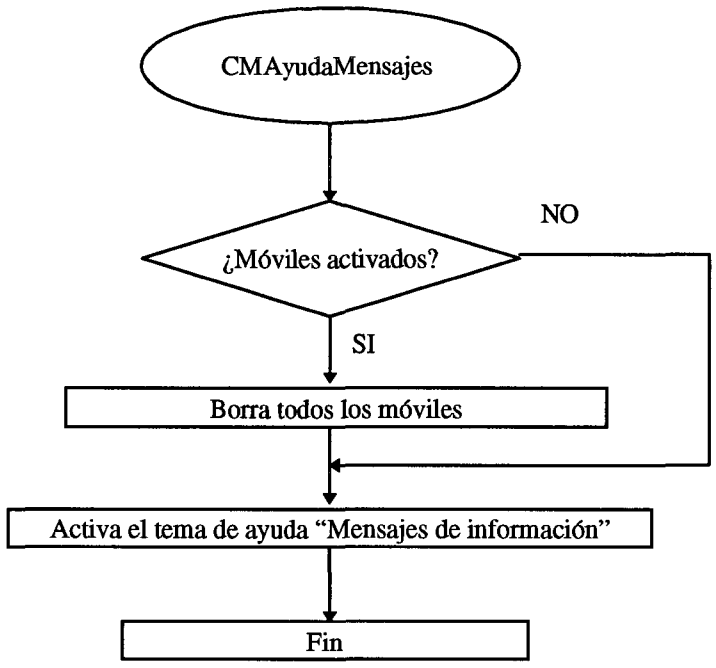
Esta función permite acceder al tema de la ayuda de la aplicación relativo al uso del ratón de la aplicación.




```
void TMainWindow :: CMAyudaRaton (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_KEY, (LONG)(LPSTR) "Ratón");
}
```

void CMAyudaMensajes (RTMessage Msg)

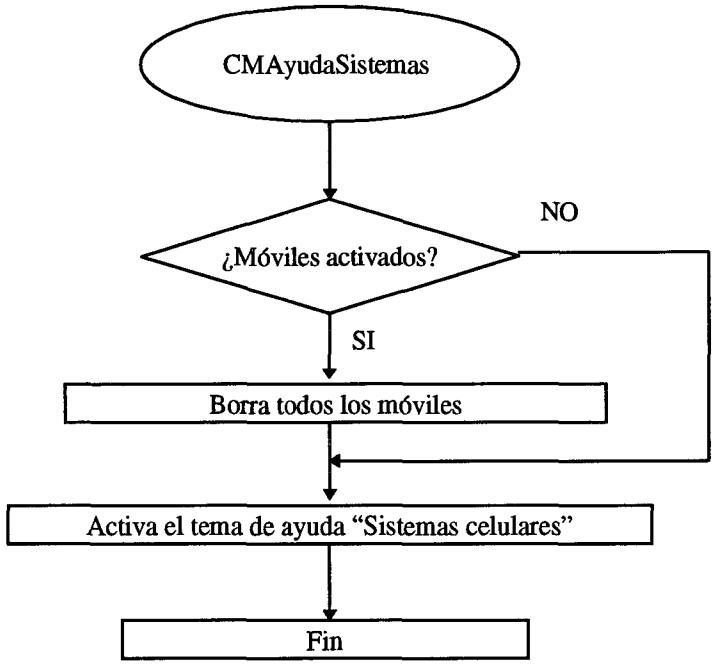
Esta función permite acceder al tema de la ayuda de la aplicación relativo a los mensajes que se pueden ver en la aplicación.




```
void TMainWindow :: CMAyudaMensajes (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_KEY, (LONG)(LPSTR) "Mensajes de información");
}
```

void CMAyudaSistemas (RTMessage Msg)

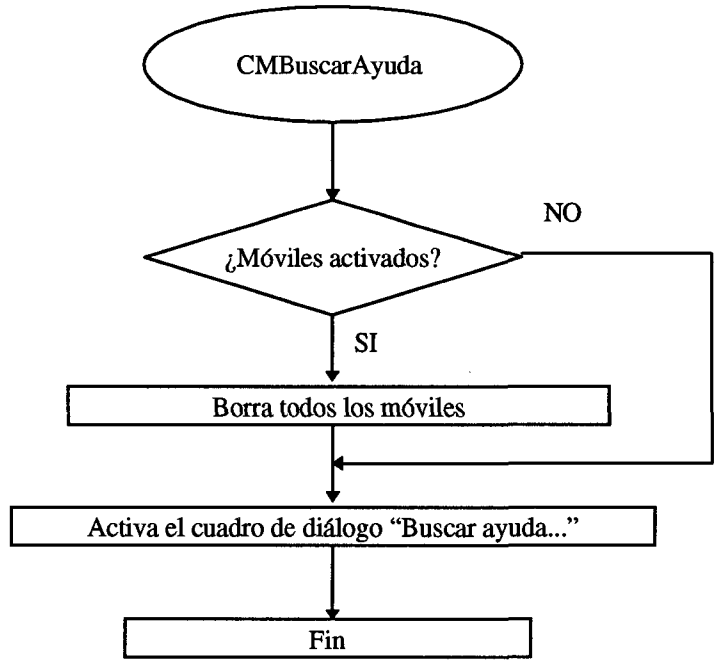
Esta función permite acceder al tema de la ayuda de la aplicación relativo a la telefonía móvil celular.




```
void TMainWindow :: CMAyudaSistemas (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_KEY, (LONG)(LPSTR) "TMA");
}
```

void CMBuscarAyuda (RTMessage Msg)

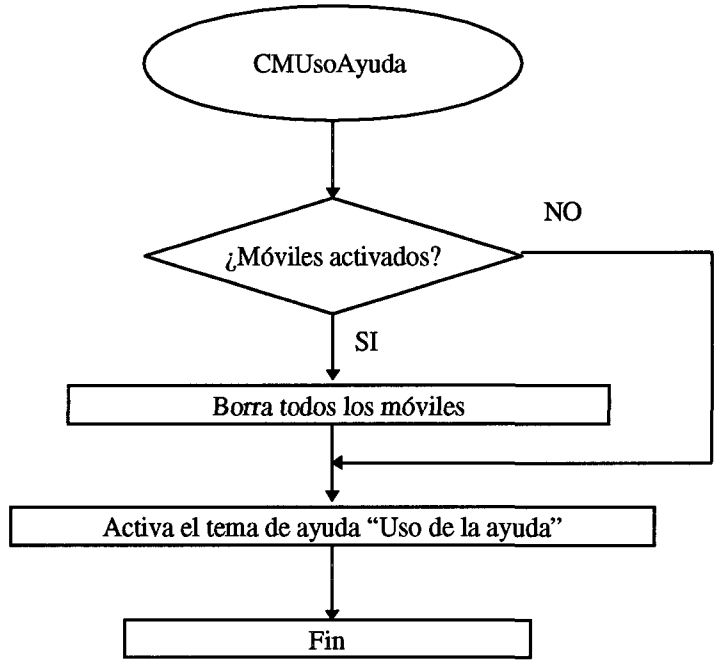
Esta función permite acceder a un cuadro de diálogo que permite seleccionar un tema o una palabra clave que se encuentre en la ayuda de la aplicación.




```
void TMainWindow :: CMBuscarAyuda (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_PARTIALKEY, 0);
}
```

void CMUsoAyuda (RTMessage Msg)

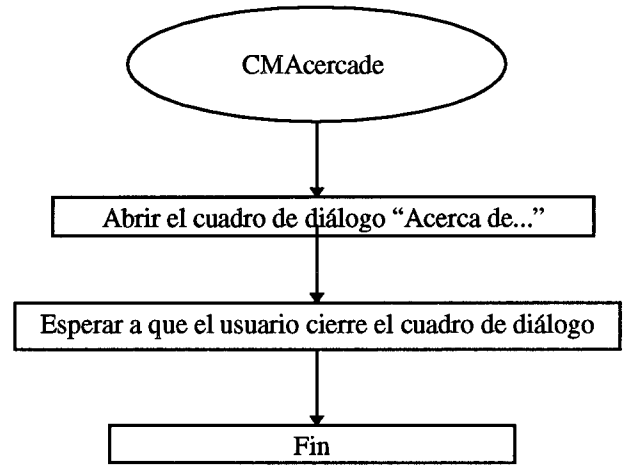
Esta función permite acceder al tema de ayuda de Windows denominado "Uso de la ayuda", donde el usuario puede aprender a utilizar cualquier ventana de ayuda de Windows.




```
void TMainWindow :: CMUsoAyuda (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_HELPONHELP, 0);
}
```

void CMAcercade (RTMessage Msg)

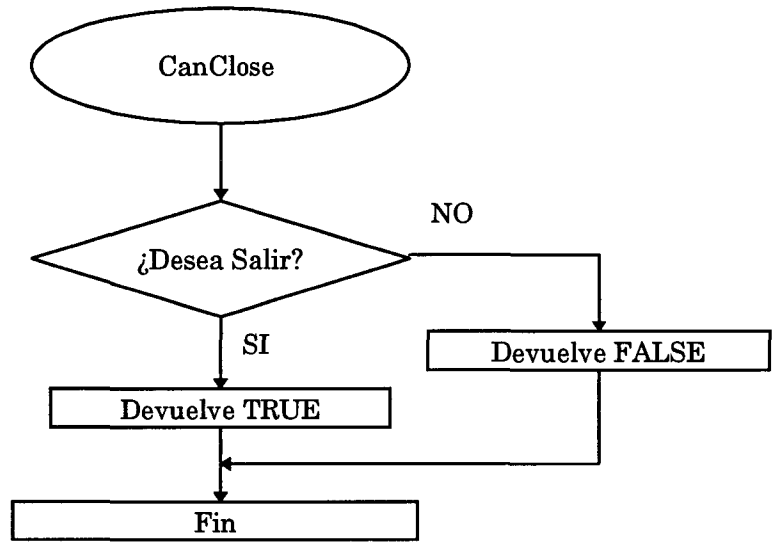
Esta función se encarga de abrir el cuadro de diálogo "Acerca de..." donde se encuentra la información referente al autor de la aplicación.




```
void TMainWindow :: CMAcercade (RTMessage Msg)
{
        PausaMoviles();
        GetModule()->ExecDialog(new TDialog(this, "AcercadeDlg"));
        ContinuaMoviles();
}
```

BOOL CanClose ()

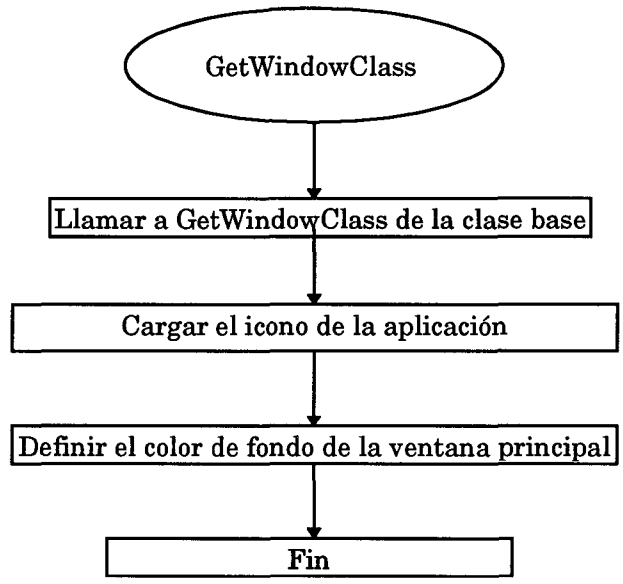
Esta función pertenece a la clase padre TWindow, y la hemos redefinido en TMainWindow para que realice convenientemente el proceso de salida de la aplicación. La función devuelve una variable booleana indicando si hay que cerrar la aplicación (TRUE), o si el usuario desea continuar en ella (FALSE).




```
BOOL TMainWindow :: CanClose ()
{
        PausaMoviles();
        MessageBeep(0);
        // Pregunta al usuario si desea abandonar la aplicación.
        if (MessageBox(HWindow, "¿Desea abandonar esta aplicación?",
                                "Salida de la aplicación",
                                MB_YESNO | MB_ICONQUESTION) == IDYES)
        {
                delete pOrigenCelulas;
                return TRUE;
        }
        else
        {
                ContinuaMoviles();
                return FALSE;
        }
}
```

void GetWindowClass (WNDCLASS & WndClass)

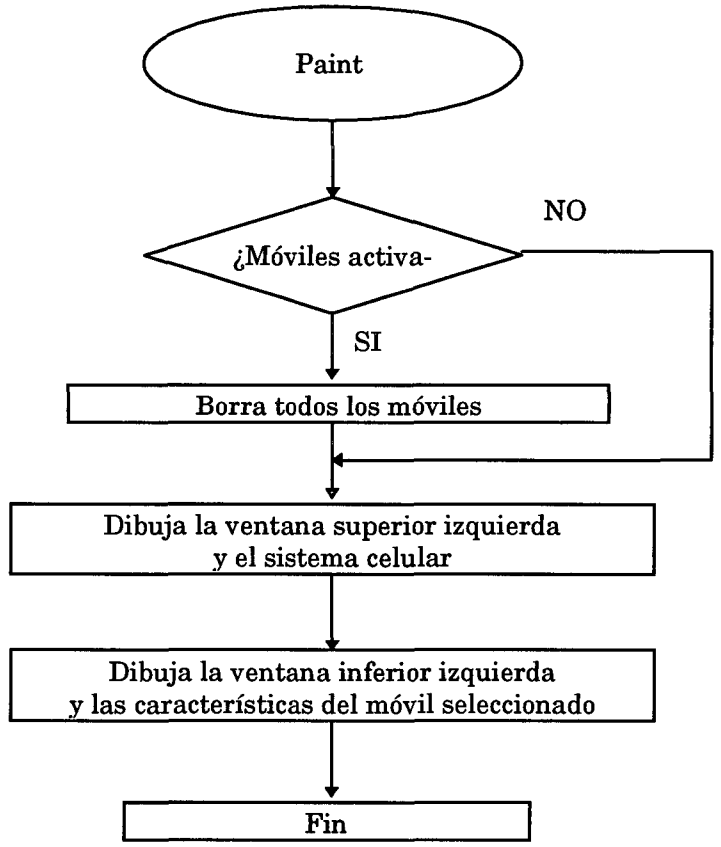
Redefiniendo este método de la clase TWindow, podemos personalizar la ventana principal de TMainWindow. No es necesario redefinir los valores por defecto si queremos utilizarlos.




```
void TMainWindow :: GetWindowClass (WNDCLASS & WndClass)
{
        TWindow :: GetWindowClass(WndClass);
        WndClass.hIcon = LoadIcon(WndClass.hInstance, "tmaIco");
        WndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
}
```

void Paint (HDC hdc, PAINTSTRUCT &PS)

Esta función se activa al recibir un mensaje de dibujar la ventana principal, ya sea todo el área de cliente o parte de él. Cada vez que se redimensiona la ventana principal habrá que volver a dibujar la ventana principal. También cuando se cierra una ventana o se desplaza sobre el área de cliente tenemos que volver a dibujar el contenido anterior de la ventana principal.



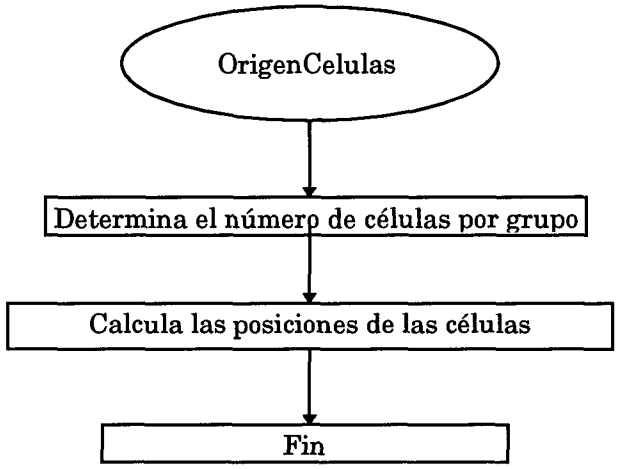
```
void TMainWindow :: Paint (HDC hdc, PAINTSTRUCT &PS)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);

        // Dibuja la ventana superior izquierda.
        BYTE by, byGrupos = (wGruposZona > 7)? 7: wGruposZona;
        HBRUSH hbrush = SelectObject(hdc, GetStockObject(GRAY_BRUSH));
        Rectangle(hdc, 5, 5, ptVentana.x*0.6-5, ptVentana.y*0.8-5);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, 10, 10, ptVentana.x*0.6-10, ptVentana.y*0.8-10);

        // Dibuja el sistema celular.
        for (by = 0; by < byGrupos; by++)
                DibujaGrupo(by, RGB(0,0,0));

        // Dibuja la ventana inferior izquierda.
        SendMessage(HWindow, WM_COMMAND, wInformacion, 0);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, 5, ptVentana.y*0.8, ptVentana.x*0.6-5, ptVentana.y-5);
        SelectObject(hdc, GetStockObject(GRAY_BRUSH));
        Rectangle(hdc, 10, ptVentana.y*0.8+5, ptVentana.x*0.6-10, ptVentana.y-10);
        DeleteObject(SelectObject(hdc, hbrush));

        // Presenta las características del móvil seleccionado.
        SetBkMode(hdc, TRANSPARENT);
        BYTE byEspacioFilas = (ptVentana.y*0.2-20)/3;
        BYTE byEspacioColum = (ptVentana.x*0.6-30)/4;
        DWORD dwMovil[NumMoviles] =
        { RGB(255,255,0), RGB(0,0,255), RGB(0,0,0), RGB(191,0,0), RGB(0,255,0) };
        SetTextColor(hdc, RGB(255,255,255));
        TextOut(hdc, 20, ptVentana.y*0.8+10, "Móvil:", 6);
        TextOut(hdc, 20, ptVentana.y*0.8+10+byEspacioFilas, "Control:", 8);
        TextOut(hdc, 20, ptVentana.y*0.8+10+byEspacioFilas*2, "Llamada:", 8);
        TextOut(hdc, 20+byEspacioColum*2, ptVentana.y*0.8+10, "Célula Actual:", 14);
        TextOut(hdc, 20+byEspacioColum*2, ptVentana.y*0.8+10+byEspacioFilas, "Grupo
Actual:", 13);
        TextOut(hdc, 20+byEspacioColum*2,ptVentana.y*0.8+10+byEspacioFilas*2,"Canal
Actual:",13);
        SetTextColor(hdc, dwMovil[byMovilSeleccionado]);
        TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10, sMovil, strlen(sMovil));
        TextOut(hdc,20+byEspacioColum,ptVentana.y*0.8+10+byEspacioFilas,sControl,strlen
(sControl));
        TextOut(hdc,20+byEspacioColum,ptVentana.y*0.8+10+byEspacioFilas*2,sLlamada,
                strlen(sLlamada));
        TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10, sCelula, strlen(sCelula));
        TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10+byEspacioFilas, sGrupo,
                strlen(sGrupo));
        TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10+byEspacioFilas*2, sCa-
nal,
                strlen(sCanal));
}
```

void OrigenCelulas (POINT ptOrigen, BYTE byGrupo)

Esta función calcula las posiciones que deben ocupar las células de un determinado grupo de nuestro sistema celular y las almacena en el vector de la clase TMainWindow ptOrigenCelulas[]. Se la pasan dos parámetros por valor:

ptOrigen: Contiene el origen del grupo.

byGrupo: Es un índice que identifica al grupo a dibujar.

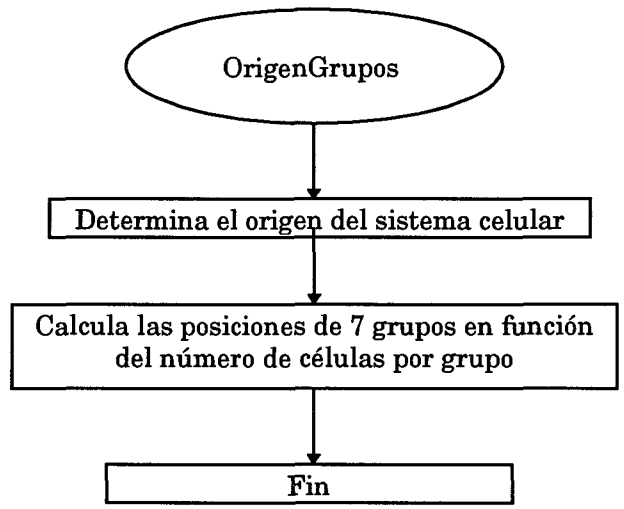



```
void TMainWindow :: OrigenCelulas (POINT ptOrigen, BYTE byGrupo)
{
        BYTE by;
        switch (byCelulasGrupo)
        {
        case 3:
                POINT pt3Celulas[] =
                {
                        ptOrigen.x-(wRadio*0.75), ptOrigen.y,
                        ptOrigen.x+(wRadio*0.75), ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x+(wRadio*0.75), ptOrigen.y+(wRadio*0.866),
                };
                for (by = 0; by < byCelulasGrupo; by++)
                        pOrigenCelulas[by+(byGrupo*byCelulasGrupo)] = pt3Celulas[by];
                break;
```

```
        case 4:
                POINT pt4Celulas[] =
                {
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y,
                        ptOrigen.x, ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x, ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y
                };
                for (by = 0; by < byCelulasGrupo; by++)
                        pOrigenCelulas[by+(byGrupo*byCelulasGrupo)] = pt4Celulas[by];
                break;
        case 7:
                POINT pt7Celulas[] =
                {
                        ptOrigen.x, ptOrigen.y,
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x, ptOrigen.y+(wRadio*1.732),
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x, ptOrigen.y-(wRadio*1.732),
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y-(wRadio*0.866),
                };
                for (by = 0; by < byCelulasGrupo; by++)
                        pOrigenCelulas[by+(byGrupo*byCelulasGrupo)] = pt7Celulas[by];
                break;
        case 12:
                POINT pt12Celulas[] =
                {
                        ptOrigen.x-(wRadio*0.25), ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x+(wRadio*1.25), ptOrigen.y,
                        ptOrigen.x-(wRadio*0.25), ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x-(wRadio*1.75), ptOrigen.y,
                        ptOrigen.x-(wRadio*1.75), ptOrigen.y-(wRadio*1.732),
                        ptOrigen.x-(wRadio*0.25), ptOrigen.y-(wRadio*2.598),
                        ptOrigen.x+(wRadio*1.25), ptOrigen.y-(wRadio*1.732),
                        ptOrigen.x+(wRadio*2.75), ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x+(wRadio*2.75), ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x+(wRadio*1.25), ptOrigen.y+(wRadio*1.732),
                        ptOrigen.x-(wRadio*0.25), ptOrigen.y+(wRadio*2.598),
                        ptOrigen.x-(wRadio*1.75), ptOrigen.y+(wRadio*1.732),
                };
                for (by = 0; by < byCelulasGrupo; by++)
                        pOrigenCelulas[by+(byGrupo*byCelulasGrupo)] = pt12Celulas[by];
                break;
        }
}
```

void OrigenGrupos ()

Esta función calcula las posiciones de los grupos que componen el sistema celular, en función del tamaño de la ventana principal y del número de células por grupo.




```
void TMainWindow :: OrigenGrupos ()
{
        POINT ptOrigen;
        ptOrigen.x = (ptVentana.x*0.6-20)/2+10;
        ptOrigen.y = (ptVentana.y*0.8-20)/2+10;
        WORD wMenor = (ptOrigen.x < ptOrigen.y)? ptOrigen.x: ptOrigen.y;
        BYTE by, byGrupos = (wGruposZona > 7)? 7: wGruposZona;
        switch (byCelulasGrupo)
        {
        case 3:
                wRadio = wMenor/5;
                POINT pt7Grupos3Celulas[] =
                {
                        ptOrigen.x, ptOrigen.y,
                        ptOrigen.x+(wRadio*3),          ptOrigen.y,
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y+(wRadio*2.598),
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y+(wRadio*2.598),
                        ptOrigen.x-(wRadio*3), ptOrigen.y,
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y-(wRadio*2.598),
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y-(wRadio*2.598)
                };
                for (by = 0; by < byGrupos; by++)
                        OrigenCelulas(pt7Grupos3Celulas[by], by);
                break;
```

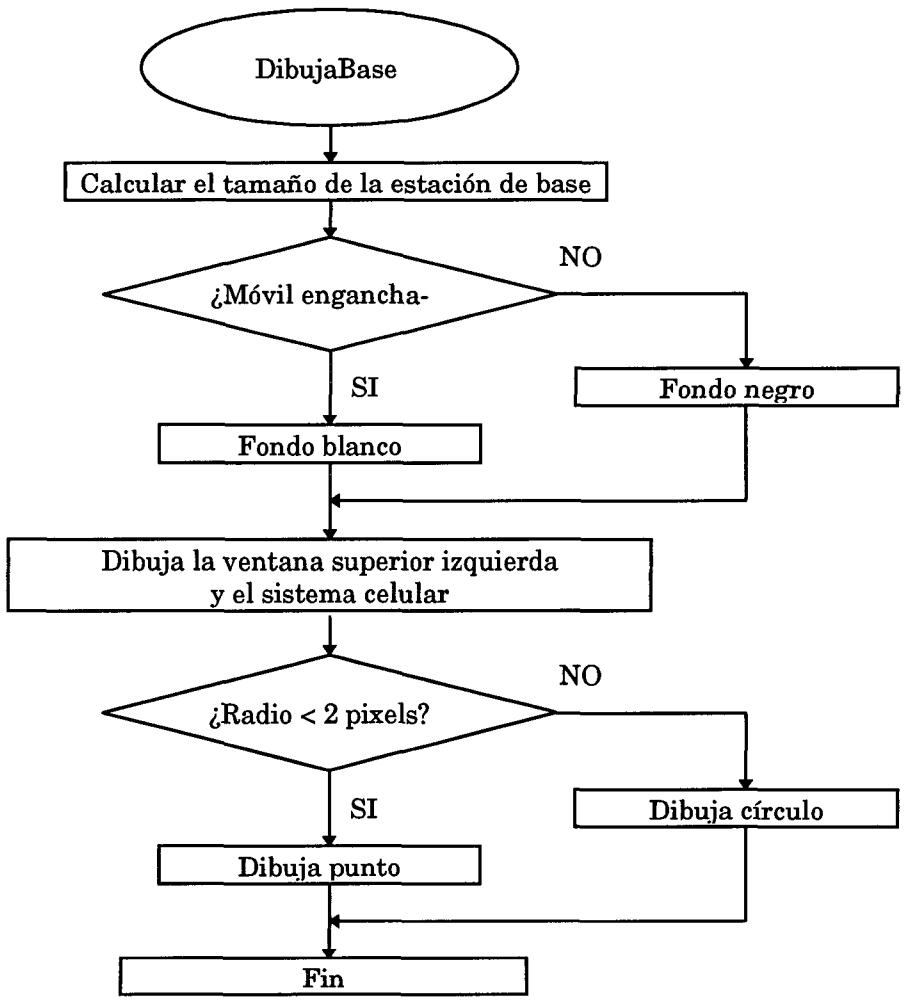
```
        case 4:
                wRadio = wMenor/6;
                POINT pt7Grupos4Celulas[] =
                {
                        ptOrigen.x, ptOrigen.y,
                        ptOrigen.x+(wRadio*3), ptOrigen.y+(wRadio*1.732),
                        ptOrigen.x, ptOrigen.y+(wRadio*3.464),
                        ptOrigen.x-(wRadio*3), ptOrigen.y+(wRadio*1.732),
                        ptOrigen.x-(wRadio*3), ptOrigen.y-(wRadio*1.732),
                        ptOrigen.x, ptOrigen.y-(wRadio*3.464),
                        ptOrigen.x+(wRadio*3), ptOrigen.y-(wRadio*1.732)
                };
                for (by = 0; by < byGrupos; by++)
                        OrigenCelulas(pt7Grupos4Celulas[by], by);
                break;
        case 7:
                wRadio = wMenor/7.5;
                POINT pt7Grupos7Celulas[] =
                {
                        ptOrigen.x, ptOrigen.y,
                        ptOrigen.x+(wRadio*4.5), ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y+(wRadio*4.33),
                        ptOrigen.x-(wRadio*3), ptOrigen.y+(wRadio*3.464),
                        ptOrigen.x-(wRadio*4.5), ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y-(wRadio*4.33),
                        ptOrigen.x+(wRadio*3), ptOrigen.y-(wRadio*3.464)
                };
                for (by = 0; by < byGrupos; by++)
                        OrigenCelulas(pt7Grupos7Celulas[by], by);
                break;
        case 12:
                wRadio = wMenor/10;
                POINT pt7Grupos12Celulas[] =
                {
                        ptOrigen.x, ptOrigen.y,
                        ptOrigen.x+(wRadio*6), ptOrigen.y,
                        ptOrigen.x+(wRadio*3), ptOrigen.y+(wRadio*5.196),
                        ptOrigen.x-(wRadio*3), ptOrigen.y+(wRadio*5.196),
                        ptOrigen.x-(wRadio*6), ptOrigen.y,
                        ptOrigen.x-(wRadio*3), ptOrigen.y-(wRadio*5.196),
                        ptOrigen.x+(wRadio*3), ptOrigen.y-(wRadio*5.196)
                };
                for (by = 0; by < byGrupos; by++)
                        OrigenCelulas(pt7Grupos12Celulas[by], by);
                break;
        }
}
```

void DibujaBase (POINT ptOrigen, BOOL bActiva)

Esta función dibuja un círculo en el centro de una célula a modo de estación de base. El círculo será de color negro cuando el móvil seleccionado no se encuentre en dicha célula, y de color blanco en caso afirmativo. Utiliza dos parámetros por valor:

ptOrigen: Contiene las coordenadas centrales del círculo

bActiva: Indica si hay que dibujar la estación de base de color blanco (TRUE) o de color negro (FALSE).



```
void TMainWindow :: DibujaBase (POINT ptOrigen, BOOL bActiva)
{
        HDC hdc = GetDC(HWindow);
        BYTE byRadioBase = ceil(wRadio/10);
        if (bActiva)
        {
                if (byRadioBase < 2)
                        SetPixel(hdc, ptOrigen.x, ptOrigen.y, RGB(255, 255, 255));
                else
                        Ellipse(hdc, ptOrigen.x-byRadioBase, ptOrigen.y-byRadioBase,
                                ptOrigen.x+byRadioBase, ptOrigen.y+byRadioBase);
        }
        else
        {
                if (byRadioBase < 2)
                        SetPixel(hdc, ptOrigen.x, ptOrigen.y, RGB(0, 0, 0));
                else
                {
                        HBRUSH hbrush = SelectObject(hdc, GetStockOb-
ject(BLACK_BRUSH));
                        Ellipse(hdc, ptOrigen.x-byRadioBase, ptOrigen.y-byRadioBase,
                                ptOrigen.x+byRadioBase, ptOrigen.y+byRadioBase);
                        DeleteObject(SelectObject(hdc, hbrush));
                }
        }
        ReleaseDC(HWindow, hdc);
}
```

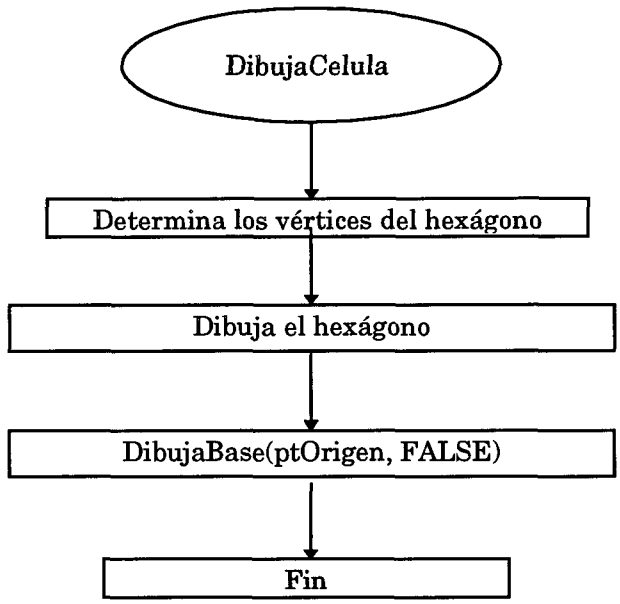
void DibujaCelula (POINT ptOrigen, DWORD dwColorBorde, DWORD dwColorFondo)

Esta función es la encargada de dibujar una célula o hexágono y de llamar a la función DibujaBase() para dibujar la estación de base. Necesita tres parámetros por valor:

ptOrigen: Indica el centro de la célula a dibujar.

dwColorBorde: Contiene el color con el que dibuja el borde del hexágono.

dwColorFondo: Contiene el color de fondo del hexágono.



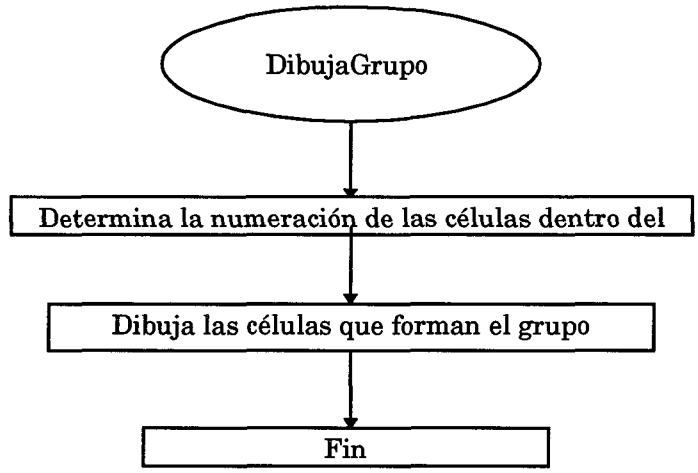

```
void TMainWindow :: DibujaCelula (POINT ptOrigen, DWORD dwColorBorde, DWORD
dwColorFondo)
{
        POINT  ptCelula[] =
        {
                ptOrigen.x+wRadio, ptOrigen.y,
                ptOrigen.x+(wRadio/2), ptOrigen.y+(wRadio*0.866),
                ptOrigen.x-(wRadio/2), ptOrigen.y+(wRadio*0.866),
                ptOrigen.x-wRadio, ptOrigen.y,
                ptOrigen.x-(wRadio/2), ptOrigen.y-(wRadio*0.866),
                ptOrigen.x+(wRadio/2), ptOrigen.y-(wRadio*0.866),
        };
        HDC hdc = GetDC(HWindow);
        HPEN hpen = SelectObject(hdc, CreatePen(PS_SOLID, 2, dwColorBorde));
        HBRUSH hbrush = SelectObject(hdc, CreateSolidBrush(dwColorFondo));
        Polygon(hdc, ptCelula, 6);
        DeleteObject(SelectObject(hdc, hpen));
        DeleteObject(SelectObject(hdc, hbrush));
        ReleaseDC (HWindow, hdc);
        DibujaBase(ptOrigen, FALSE);
}
```

void DibujaGrupo (BYTE byGrupo, DWORD dwColorBorde)

Esta función se encarga de llamar a la función DibujaCelula() para dibujar tantas células como sean necesarias para crear un grupo. Utiliza dos parámetros por valor:

byGrupo: Es un índice que identifica al grupo a dibujar.

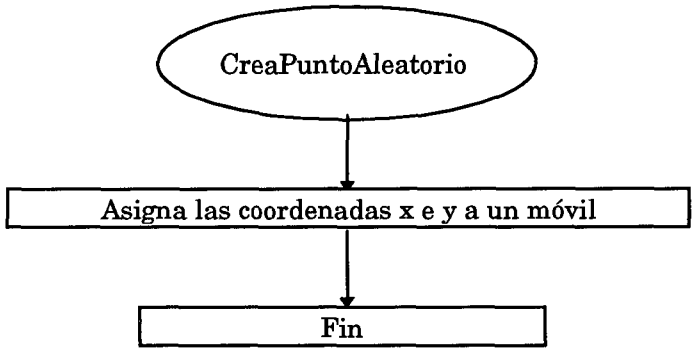
dwColorBorde: Contiene el color del borde con el cuál dibujar todas las células del grupo.




```
void TMainWindow :: DibujaGrupo (BYTE byGrupo, DWORD dwColorBorde)
{
        DWORD dwColorFondo[] =
        {
                RGB(255,255,0), RGB(0,255,128), RGB(0,255,255), RGB(0,128,128),
                RGB(0,0,255), RGB(255,0,128), RGB(128,0,128), RGB(128,128,0),
                RGB(255,0,0), RGB(128,128,128), RGB(255,0,255), RGB(128,255,0)
        };
        BYTE by, byColor = 0, byPrimeraCelula = byGrupo*byCelulasGrupo,
                byUltimaCelula = byPrimeraCelula+byCelulasGrupo;
        for (by = byPrimeraCelula; by < byUltimaCelula; by++)
        {
                DibujaCelula(pOrigenCelulas[by], dwColorBorde, dwColorFondo[byColor]);
                byColor++;
        }
}
```

void CreaPuntoAleatorio (POINT &ptMovil)

Esta función crea las coordenadas de un móvil identificado por el parámetro **ptMovil** de forma aleatoria dentro de la ventana superior izquierda de la aplicación. El parámetro ptMovil se pasa por referencia.



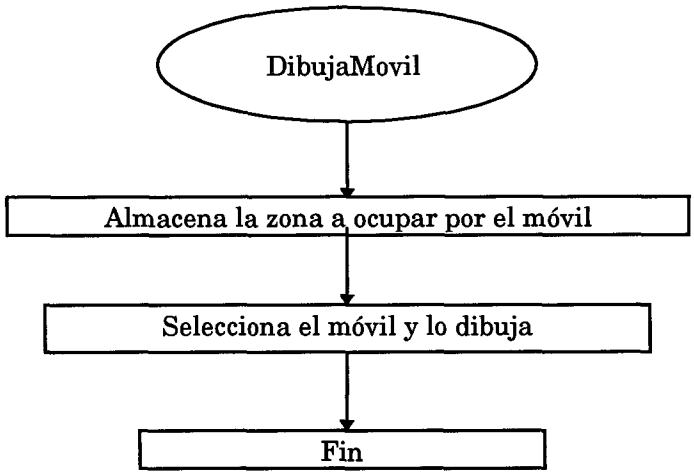

```
void TMainWindow :: CreaPuntoAleatorio (POINT &ptMovil)
{
        ptMovil.x = 10+random(ptVentana.x*0.6-20);
        ptMovil.y = 10+random(ptVentana.y*0.8-20);
}
```

void DibujaMovil (POINT ptMovil, BYTE byMovil)

Esta función dibuja un bitmap en pantalla almacenando la imagen que había anteriormente en el rectángulo ocupado por el bitmap, donde el bitmap es un automóvil. Utiliza dos parámetros por valor:

ptMovil: Contiene la posición donde se va a dibujar el móvil.

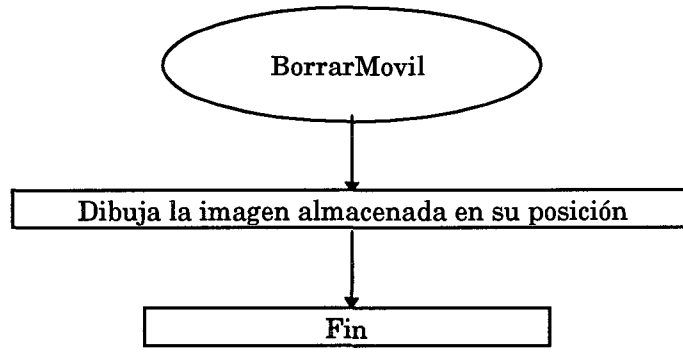
byMovil: Identifica al móvil que se quiere dibujar.




```
void TMainWindow :: DibujaMovil (POINT ptMovil, BYTE byMovil)
{
        HDC hdc = GetDC(HWindow), hdcMem = CreateCompatibleDC(hdc);
        for (WORD x = 0; x < xBmp; x++)
                for (WORD y = 0; y < yBmp; y++)
                        dwColor[byMovil][x][y] = GetPixel(hdc, ptMovil.x+x, ptMovil.y+y);
        HBITMAP hbitmap;
        switch (byMovil)
        {
        case 0:
                hbitmap = LoadBitmap(GetModule()->hInstance,"amarilloBmp");
                break;
        case 1:
                hbitmap = LoadBitmap(GetModule()->hInstance,"azulBmp");
                break;
        case 2:
                hbitmap = LoadBitmap(GetModule()->hInstance,"negroBmp");
                break;
        case 3:
                hbitmap = LoadBitmap(GetModule()->hInstance,"rojoBmp");
                break;
        case 4:
                hbitmap = LoadBitmap(GetModule()->hInstance,"verdeBmp");
                break;
        }
        SelectObject(hdcMem, hbitmap);
        BitBlt(hdc, ptMovil.x, ptMovil.y, xBmp, yBmp, hdcMem, 0, 0, SRCCOPY);
        DeleteDC(hdcMem);
        DeleteObject(hbitmap);
        ReleaseDC(HWindow, hdc);
}
```

void BorraMovil (POINT ptMovil, BYTE byMovil)

Esta función recupera la imagen guardada al dibujar un móvil en pantalla y la sitúa en su lugar consiguiendo borrar el móvil.

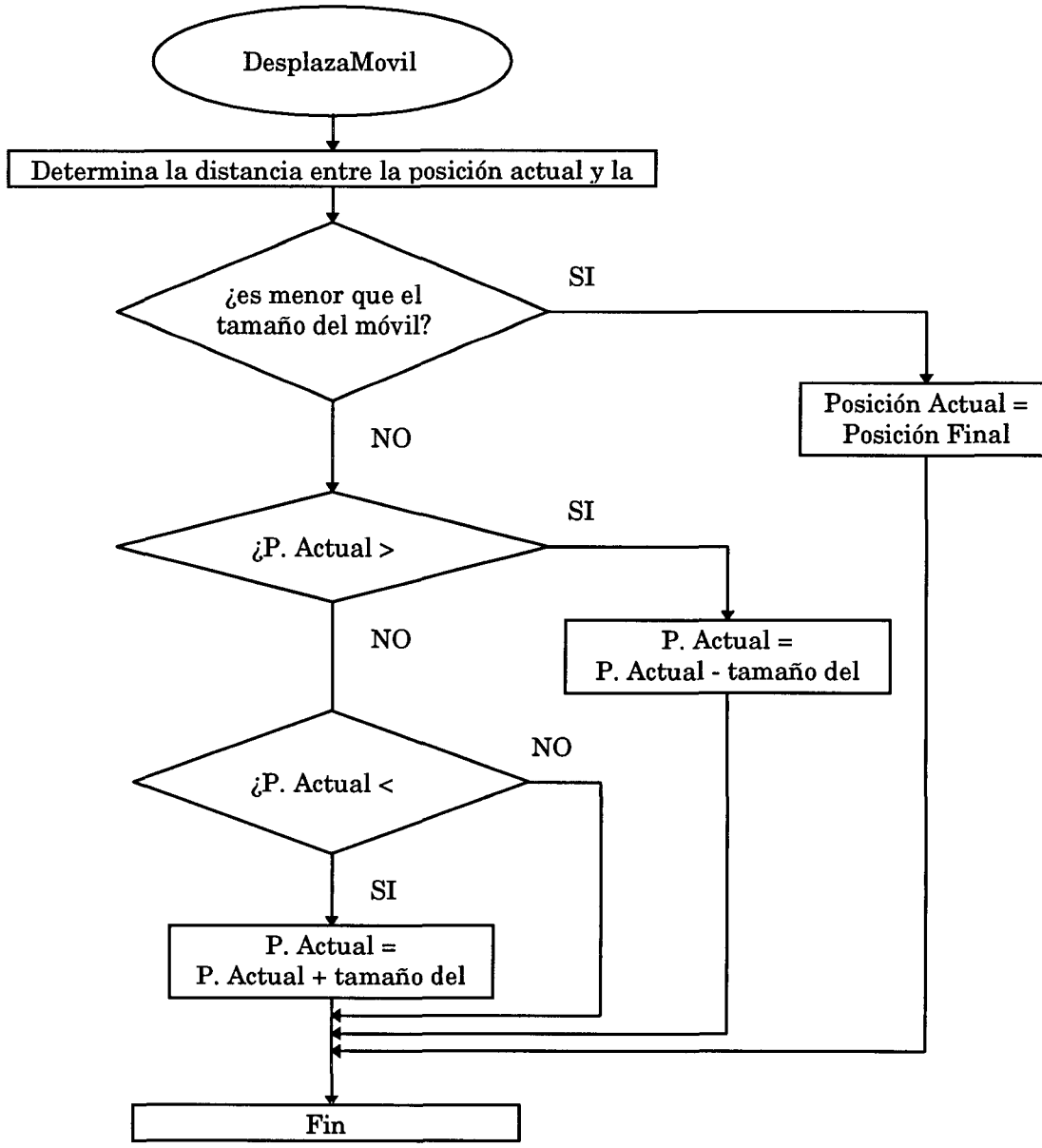



```
void TMainWindow :: BorraMovil (POINT ptMovil, BYTE byMovil)
{
        HDC hdc = GetDC(HWindow);
        for (WORD x = 0; x < xBmp; x++)
                for (WORD y = 0; y < yBmp; y++)
                        SetPixel(hdc, ptMovil.x+x, ptMovil.y+y, dwColor[byMovil][x][y]);
        ReleaseDC(HWindow, hdc);
}
```

void DesplazaMovil (POINT &ptActual, POINT ptFinal)

Esta función desplaza un móvil desde la posición en la que se encuentra hacia la posición final a la que se dirige. Utiliza una variable por referencia **ptActual**, y otra por valor, **ptFinal**:

ptActual: Posición en la que se encuentra el móvil.
ptFinal: Posición hacia la que se dirige el móvil.

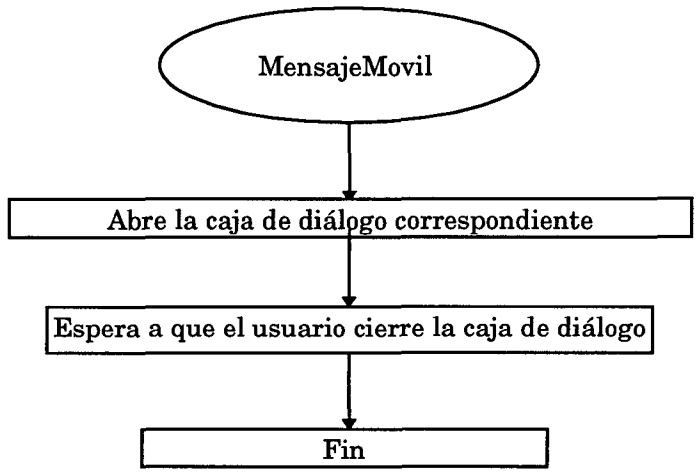


```
void TMainWindow :: DesplazaMovil (POINT &ptActual, POINT ptFinal)
{
        POINT ptDistancia;
        ptDistancia.x = (ptActual.x > ptFinal.x)? ptActual.x-ptFinal.x: ptFinal.x-ptActual.x;
        if (ptDistancia.x <= yBmp-1)
                ptActual.x = ptFinal.x;
        else
        {
                if (ptActual.x > ptFinal.x)
                        ptActual.x -= yBmp;
                else
                        if (ptActual.x < ptFinal.x)
                                ptActual.x += yBmp;
        }
        ptDistancia.y = (ptActual.y > ptFinal.y)? ptActual.y-ptFinal.y: ptFinal.y-ptActual.y;
        if (ptDistancia.y <= yBmp-1)
                ptActual.y = ptFinal.y;
        else
        {
                if (ptActual.y > ptFinal.y)
                        ptActual.y -= yBmp;
                else
                        if (ptActual.y < ptFinal.y)
                                ptActual.y += yBmp;
        }
}
```

void MensajeMovil (BYTE byMensaje)

Esta función se encarga de enviar los mensajes de información relativos a los móviles presentes en la pantalla principal. Utiliza una variable por valor:

byMensaje: Identifica al mensaje a enviar.




```
void TMainWindow :: MensajeMovil (BYTE byMensaje)
{
        MessageBeep(0);
        PausaMoviles();
        char sMovil[5][9] =
        {"amarillo", "azul", "negro", "rojo", "verde"};
        char sMsg[200];
        if (byMensaje == 4)
        {
                strcpy(sMsg, "El móvil ");
                strcat(sMsg, sMovil[byMovilSeleccionado]);
                strcat(sMsg, " ha recibido una llamada telefónica.\n" "¿Desea atender la lla-
mada?");
                if (MessageBox(HWindow, sMsg, "Aviso de llamada", MB_YESNO |
                        MB_ICONINFORMATION) == IDYES)
                {
                        SendMessage(HWindow, WM_COMMAND,
CM_EFECTUARLLAMADA, 0);
                        bLlamadaRecibida = TRUE;
                }
        }
        else
        {
                switch (byMensaje)
                {
                case 0:
                        strcpy(sMsg, "Para poder desplazar el móvil ");
                        strcat(sMsg, sMovil[byMovilSeleccionado]);
                        strcat(sMsg, " a la posición actual del cursor, tiene que "
                                "estar habilitada la opción <<Movimiento por ratón>> "
                                "del cuadro de diálogo <<Seleccionar Móvil...>>");
                        break;
                case 1:
                        strcpy(sMsg, "No se puede desplazar el móvil ");
                        strcat(sMsg, sMovil[byMovilSeleccionado]);
                        strcat(sMsg, " fuera del rectángulo que recubre el sistema celular.");
```

```
                case 2:
                        strcpy(sMsg, "El móvil ");
                        strcat(sMsg, sMovil[byMovilSeleccionado]);
                        strcat(sMsg, " ha perdido la comunicación por no encontrase ningún "
                                "canal disponible en la célula en la que se encuentra.");
                        break;
                case 3:
                        strcpy(sMsg, "La LLamada se ha perdido por encontrase el móvil "
                                "fuera del área de cobertura del sistema celular.");
                        break;
                }
                MessageBox(HWindow, sMsg, "Localización del móvil", MB_OK |
                        MB_ICONINFORMATION);
        }
        ContinuaMoviles();
}
```

void CaracteristicasMovil (BOOL bEnganchado, BOOL bMismaBase, BYTE byBase, BYTE byGrupo)

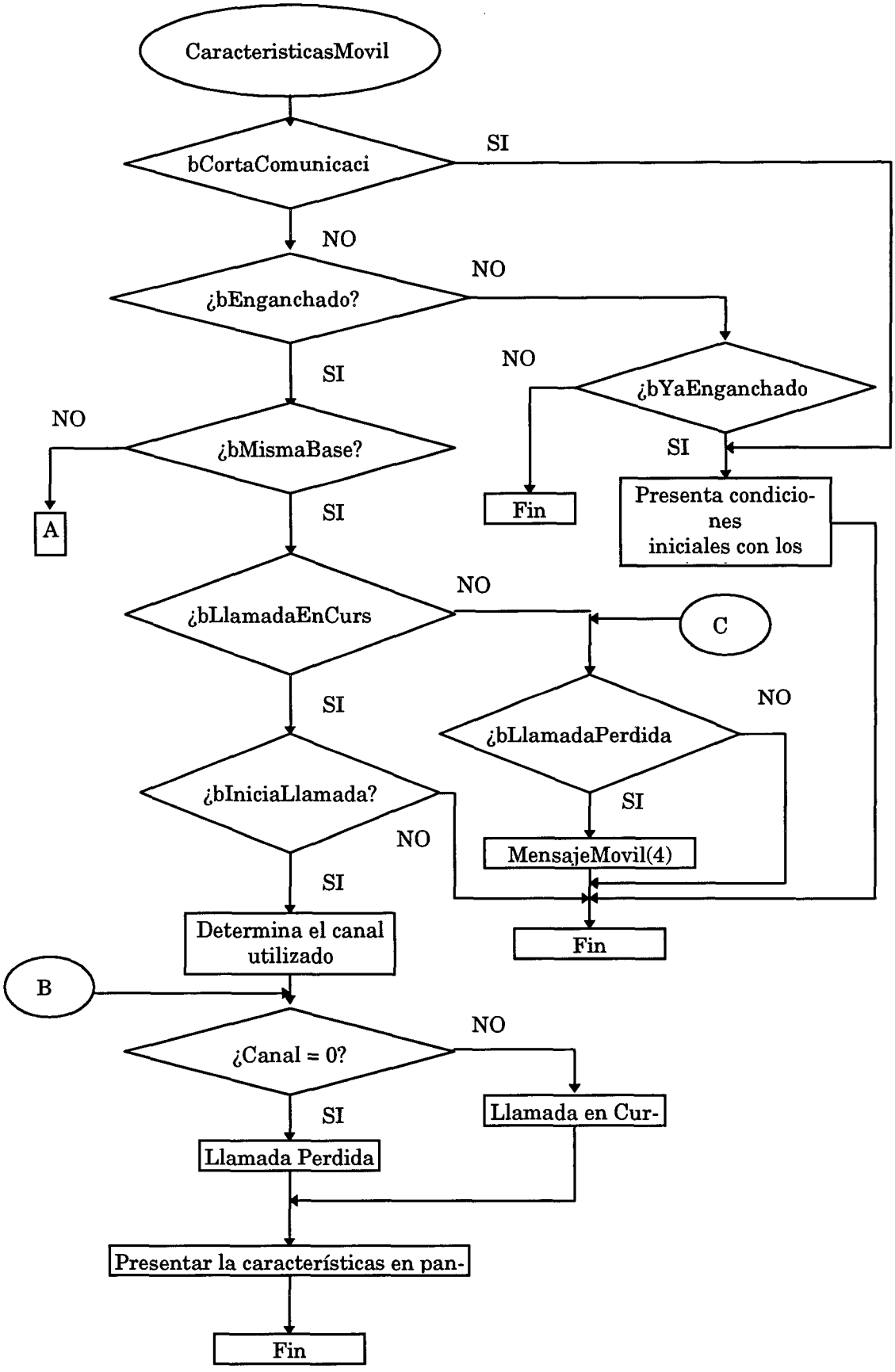
Esta función presenta en la ventana inferior izquierda las características del móvil seleccionado en función de cuatro parámetros que obtiene por valor:

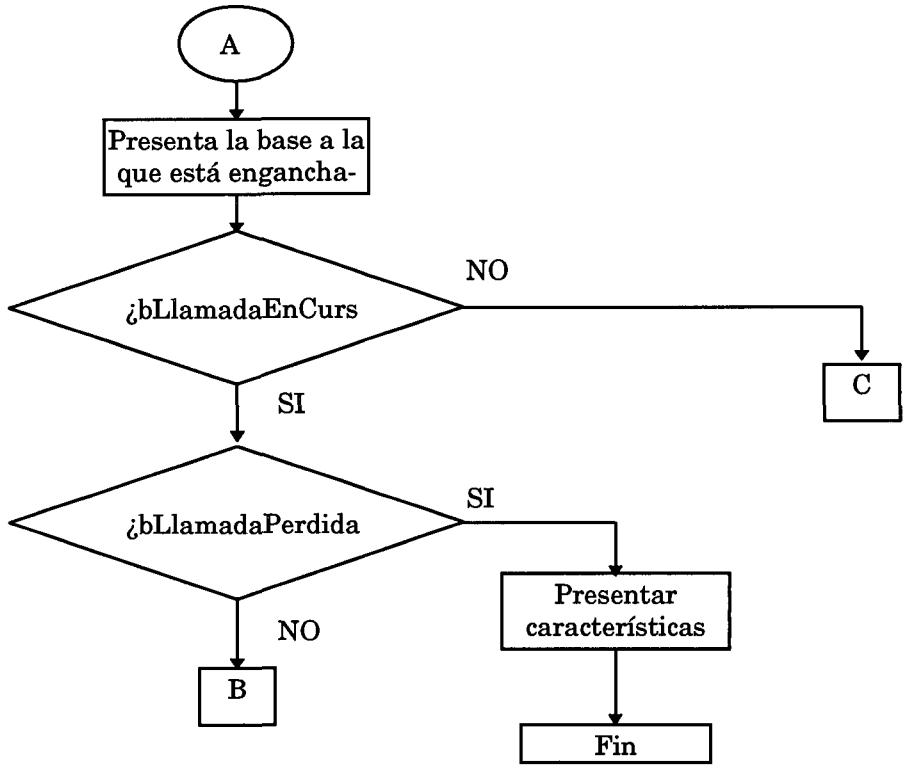
bEnganchado: Indica si el móvil seleccionado está enganchado a alguna estación de base del sistema celular.

bMismaBase: Indica si el móvil seleccionado se encontraba en la misma base que se encuentra ahora.

byBase: Indica la base en la que se encuentra el móvil seleccionado.

byGrupo: Indica el grupo en el que se encuentra el móvil seleccionado.






```
void TMainWindow :: CaracteristicasMovil (BOOL bEnganchado, BOOL bMismaBase, BYTE
byBase
                        BYTE byGrupo)
{
        HDC hdc = GetDC(HWindow);
        BYTE byEspacioFilas = (ptVentana.y*0.2-20)/3;
        BYTE byEspacioColum = (ptVentana.x*0.6-30)/4;
        WORD wCanal;
        HBRUSH hbrush = SelectObject(hdc, GetStockObject(GRAY_BRUSH));
        HPEN hpen = SelectObject(hdc, CreatePen(PS_SOLID, 1, RGB(128,128,128)));
        DWORD dwMovil[NumMoviles] =
        { RGB(255,255,0), RGB(0,0,255), RGB(0,0,0), RGB(191,0,0), RGB(0,255,0) };
        SetTextColor(hdc, dwMovil[byMovilSeleccionado]);
        SetBkMode(hdc, TRANSPARENT);
        if (bCortaComunicacion)
        {
                bCortaComunicacion = FALSE;
                SetClassWord(HWindow, GCW_HCURSOR, LoadCursor(NULL,
IDC_ARROW));
                Rectangle(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+byEspacioFilas*2+10,
                        19+byEspacioColum*4, ptVentana.y*0.8+10+byEspacioFilas*3);
                strcpy(sCanal, "0");
                TextOut(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+10+byEspacioFilas*2, sCanal,
```

```
                strcpy(sLlamada, "Finalizada");
                TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10+byEspacioFilas*2,
sLlamada,
                        strlen(sLlamada));
                }
        else
        {
                if (bEnganchado)
                {
                        if (bMismaBase)
                        {
                                if (bLlamadaEnCurso)
                                {
                                        if (bIniciaLlamada)
                                        {
                                                bIniciaLlamada = FALSE;
                                                wCanal = CanalUtilizado(byBase);
                                                if (wCanal == 0)
                                                {
                                                        strcpy(sLlamada, "Perdida");
                                                        bLlamadaPerdida = TRUE;
                                                        SetClassWord(HWindow,
GCW_HCURSOR,
                                                                LoadCursor(NULL,
IDC_CROSS));
                                                }
                                                else
                                                {
                                                        strcpy(sLlamada, "En Curso");
                                                        SetClassWord(HWindow,
GCW_HCURSOR,
                                                        LoadCursor(GetModule()->hInstance,
                                                                "tmaCur"));
                                                }
                                                Rectangle(hdc, 20+byEspacioColum*3.5,
                                                        ptVentana.y*0.8+byEspacioFilas*2+10,
                                                        19+byEspacioColum*4,
                                                        ptVentana.y*0.8+10+byEspacioFilas*3);
                                                itoa(wCanal, sCanal, 10);
                                                TextOut(hdc, 20+byEspacioColum*3.5,
                                                ptVentana.y*0.8+10+byEspacioFilas*2, sCanal,
                                                        strlen(sCanal));
                                                Rectangle(hdc, 20+byEspacioColum,
                                                        ptVentana.y*0.8+10+byEspacioFilas*2,
                                                        19+byEspacioColum*2,
                                                        ptVentana.y*0.8+10+byEspacioFilas*3);
                                                TextOut(hdc, 20+byEspacioColum,
                                                        ptVentana.y*0.8+10+byEspacioFilas*2,
                                                        sLlamada, strlen(sLlamada));
                                        }
                                }
                                else
                                        if (MovilBuffer.wRecibirLlamadas == BF_CHECKED)
```

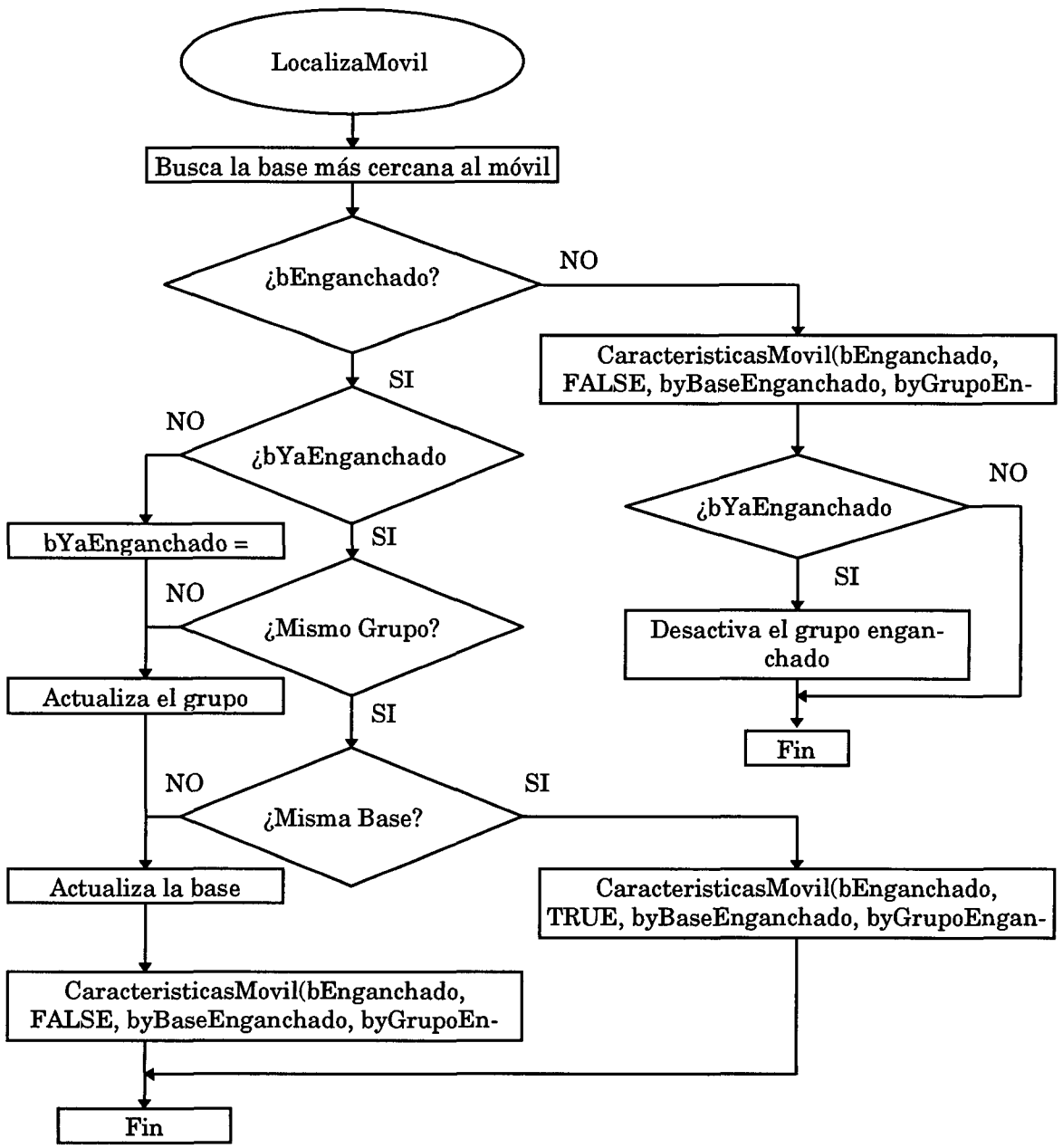
```
                    else
                    {
                        Rectangle(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10,
                            19+byEspacioColum*4, ptVenta-
na.y*0.8+10+byEspacioFilas);
                        itoa(byBase-(byGrupo*byCelulasGrupo)+1, sCelula, 10);
                        TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10,
sCelula,
                            strlen(sCelula));
                        Rectangle(hdc, 20+byEspacioColum*3.5,
                        ptVentana.y*0.8+10+byEspacioFilas,
                        19+byEspacioColum*4, ptVenta-
na.y*0.8+10+byEspacioFilas*2);
                        itoa(byGrupo+1, sGrupo, 10);
                        TextOut(hdc, 20+byEspacioColum*3.5,
                            ptVentana.y*0.8+10+byEspacioFilas, sGrupo,
strlen(sGrupo));
                        if (bLlamadaEnCurso)
                        {
                            if (!bLlamadaPerdida)
                            {
                                wCanal = CanalUtilizado(byBase);
                                if (wCanal == 0)
                                {
                                    strcpy(sLlamada, "Perdida");
                                    bLlamadaPerdida = TRUE;
                                    SetClassWord(HWindow,
GCW_HCURSOR,
                                        LoadCursor(NULL,
IDC_CROSS));
                                }
                                else
                                {
                                    strcpy(sLlamada, "En Curso");
                                    SetClassWord(HWindow,
GCW_HCURSOR,
                                    LoadCursor(GetModule()->hInstance,
                                        "tmaCur"));
                                }
                                Rectangle(hdc, 20+byEspacioColum*3.5,
                                    ptVentana.y*0.8+byEspacioFilas*2+10,
                                    19+byEspacioColum*4,
                                    ptVenta
        na.y*0.8+10+byEspacioFilas*3);
                                itoa(wCanal, sCanal, 10);
                                TextOut(hdc, 20+byEspacioColum*3.5,
                                    ptVentana.y*0.8+10+byEspacioFilas*2,
                                    sCanal, strlen(sCanal));
                                Rectangle(hdc, 20+byEspacioColum,
                                    ptVentana.y*0.8+10+byEspacioFilas*2,
                                    19+byEspacioColum*2,
                                    ptVentana.y*0.8+10+byEspacioFilas*3);
```

```
                                else
                                        if (MovilBuffer.wRecibirLlamadas == BF_CHECKED)
                                                if (random(21) == 0)
                                                        MensajeMovil(4);
                                }
                        }
                }
                else
                {
                        if (bYaEnganchado)
                        {
                                bYaEnganchado = FALSE;
                                Rectangle(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10,
                                        19+byEspacioColum*4, ptVenta-
na.y*0.8+10+byEspacioFilas);
                                strcpy(sCelula, "0");
                                TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10,
                                        sCelula, strlen(sCelula));
                                Rectangle(hdc, 20+byEspacioColum*3.5,
                                        ptVentana.y*0.8+10+byEspacioFilas,
19+byEspacioColum*4,
                                        ptVentana.y*0.8+10+byEspacioFilas*2);
                                strcpy(sGrupo, "0");
                                TextOut(hdc, 20+byEspacioColum*3.5,
                                        ptVentana.y*0.8+10+byEspacioFilas, sGrupo,
strlen(sGrupo));
                        }
                        if ((bLlamadaEnCurso)&&(!bLlamadaPerdida))
                        {
                                bLlamadaPerdida = TRUE;
                                bIniciaLlamada = FALSE;
                                SetClassWord(HWindow, GCW_HCURSOR, LoadCur-
sor(NULL,
                                        IDC_CROSS));
                                Rectangle(hdc, 20+byEspacioColum,
                                ptVentana.y*0.8+10+byEspacioFilas*2, 19+byEspacioColum*2,
                                ptVentana.y*0.8+10+byEspacioFilas*3);
                                strcpy(sLlamada, "Perdida");
                                TextOut(hdc, 20+byEspacioColum,
                                        ptVentana.y*0.8+10+byEspacioFilas*2,
                                        sLlamada, strlen(sLlamada));
                                Rectangle(hdc, 20+byEspacioColum*3.5,

        ptVentana.y*0.8+10+byEspacioFilas*2,19+byEspacioColum*4,
                                        ptVentana.y*0.8+10+byEspacioFilas*3);
                                strcpy(sCanal, "0");
                                TextOut(hdc, 20+byEspacioColum*3.5,
                                ptVentana.y*0.8+10+byEspacioFilas*2, sCanal,
strlen(sCanal));
                                if (MovilBuffer.wMsgSalidaZona == BF_CHECKED)
                                        MensajeMovil(3);
                        }
                }
```

void LocalizaMovil (POINT ptMovil)

Esta función es la que calcula la posición en la que se encuentra el móvil indicado por **ptMovil** y llama a las funciones necesarias para que visualicen los cambios producidos por el movimiento del móvil. Utiliza un parámetro por valor:

ptMovil: Identifica al móvil que se quiere estudiar.



```
void TMainWindow :: LocalizaMovil (POINT ptMovil)
{
        WORD wMenorDistancia, wDistancia, wDistanciax, wDistanciay;
        BYTE by, byGrupos, byBaseEnganchado = 0, byGrupoEnganchado;
        BOOL bEnganchado;
        wDistanciax = (pOrigenCelulas[0].x > ptMovil.x)? pOrigenCelulas[0].x-ptMovil.x:
                ptMovil.x-pOrigenCelulas[0].x;
        wDistanciay = (pOrigenCelulas[0].y > ptMovil.y)? pOrigenCelulas[0].y-ptMovil.y:
                ptMovil.y-pOrigenCelulas[0].y;
        wMenorDistancia = sqrt(pow((wDistanciax),2)+pow((wDistanciay),2));
        byGrupos = (wGruposZona > 7)? 7: wGruposZona;
        for (by = 1; by < byCelulasGrupo*byGrupos; by++)
        {
                wDistanciax = (pOrigenCelulas[by].x > ptMovil.x)? pOrigenCelulas[by].x-
ptMovil.x:
                        ptMovil.x-pOrigenCelulas[by].x;
                wDistanciay = (pOrigenCelulas[by].y > ptMovil.y)? pOrigenCelulas[by].y-
ptMovil.y:
                        ptMovil.y-pOrigenCelulas[by].y;

        if((wMenorDistancia)>=(wDistancia=sqrt(pow((wDistanciax),2)+pow((wDistanciay),2))
))
                {
                        byBaseEnganchado = by;
                        wMenorDistancia = wDistancia;
                }
        }
        bEnganchado = (wMenorDistancia > wRadio)? FALSE: TRUE;
        if (bEnganchado)
        {
                byGrupoEnganchado = byBaseEnganchado/byCelulasGrupo;
                if (bYaEnganchado)
                {
                        if (byGrupoEnganchado == byGrupoAnterior)
                        {
                                if (byBaseEnganchado != byBaseAnterior)
                                {
                                        DibujaBase(pOrigenCelulas[byBaseAnterior], FALSE);
                                        DibujaBase(pOrigenCelulas[byBaseEnganchado],
TRUE);
                                        byBaseAnterior = byBaseEnganchado;

        CaracteristicasMovil(bEnganchado,FALSE,byBaseEnganchado,
                                                byGrupoEnganchado);
                                }
                                else
```

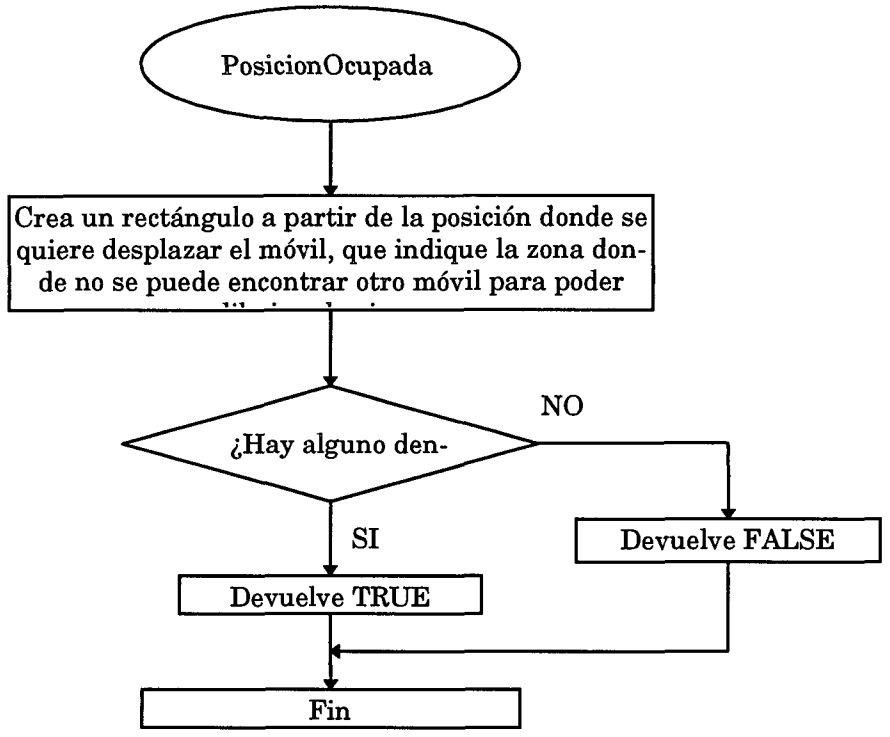
```
                    else
                    {
                        DibujaGrupo(byGrupoAnterior, RGB(0,0,0));
                        DibujaGrupo(byGrupoEnganchado, RGB(255,255,255));
                        DibujaBase(pOrigenCelulas[byBaseEnganchado], TRUE);
                        byGrupoAnterior = byGrupoEnganchado;
                        byBaseAnterior = byBaseEnganchado;
                        CaracteristicasMovil(bEnganchado, FALSE, byBaseEnganchado,
                                byGruoEnganchado);
                    }
                }
                else
                {
                    DibujaGrupo(byGrupoEnganchado, RGB(255,255,255));
                    DibujaBase(pOrigenCelulas[byBaseEnganchado], TRUE);
                    byGrupoAnterior = byGrupoEnganchado;
                    byBaseAnterior = byBaseEnganchado;
                    bYaEnganchado = TRUE;
                    CaracteristicasMovil(bEnganchado, FALSE, byBaseEnganchado,
                        byGrupoEnganchado);
                }
            }
            else
            {
                CaracteristicasMovil(bEnganchado, FALSE, byBaseEnganchado, byGrupoEnganchado);
                if (bYaEnganchado)
                {
                    DibujaGrupo(byGrupoAnterior, RGB(0,0,0));
                    bYaEnganchado = FALSE;
                }
            }
```

BOOL PosicionOcupada (POINT ptMovil, BYTE byMovil)

Esta función evita que se dibuje un móvil sobre otro ocupando las mismas posiciones en la pantalla. Devuelve un valor booleano, indicando si la posición a la que se desea desplazar un móvil ya está ocupada (TRUE), o por el contrario no lo está (FALSE). Utiliza dos parámetros por valor:

ptMovil: Contiene la posición que se va a comprobar.

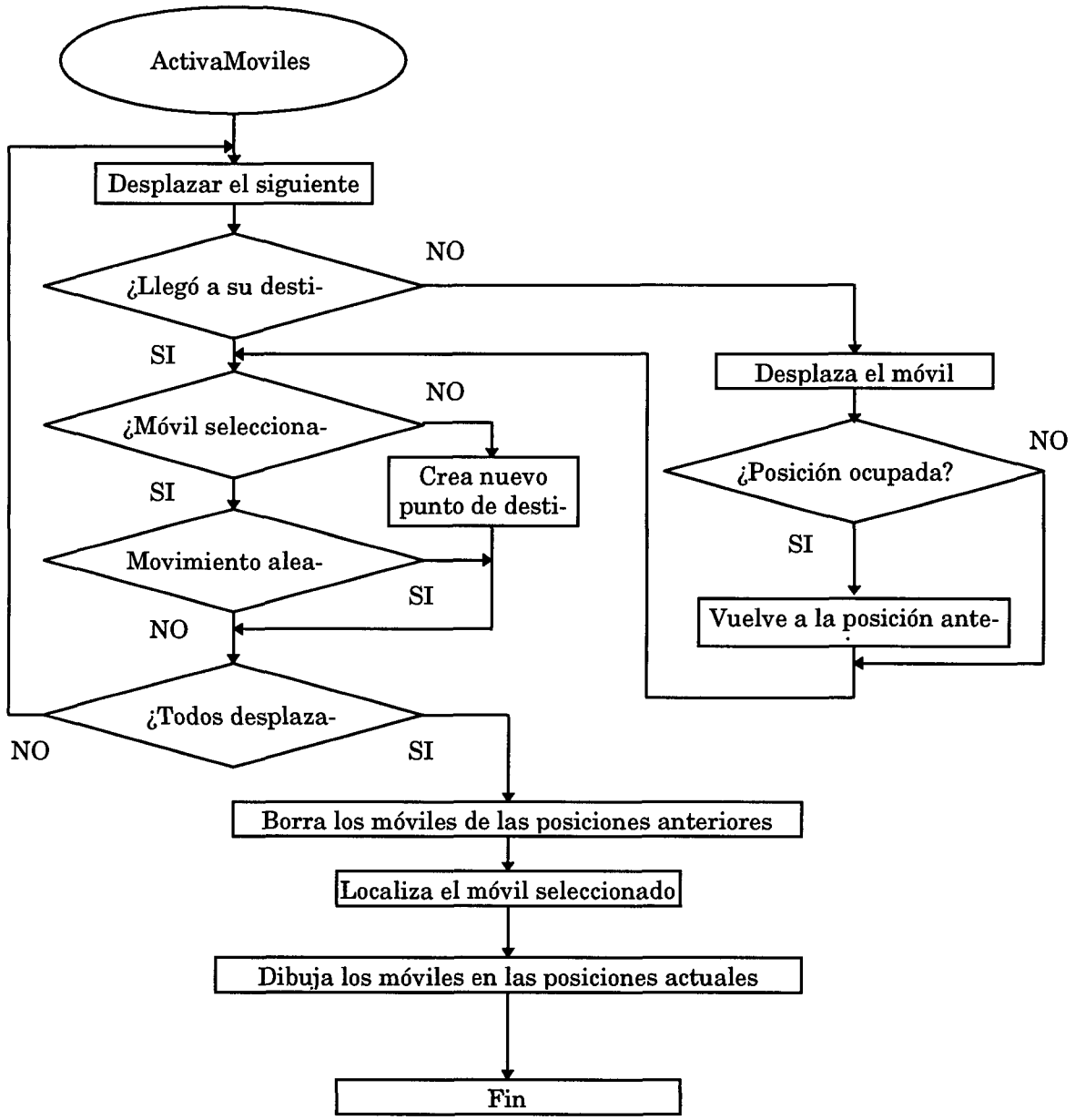
byMovil: Indica cuál es el móvil que vamos a desplazar.




```
BOOL TMainWindow :: PosicionOcupada (POINT ptMovil, BYTE byMovil)
{
        RECT rect =
        {
                ptMovil.x-xBmp, ptMovil.y-yBmp,
                ptMovil.x+(2*xBmp), ptMovil.y+(2*yBmp)
        };
        BOOL bDentroRect = FALSE;
        BYTE by=0;
        do
        {
                if (byMovil != by)
                        bDentroRect = PtInRect(&rect, ptActual[by]);
                by++;
        }
        while ((by < NumMoviles) && (!bDentroRect));
                return bDentroRect;
}
```

void ActivaMoviles ()

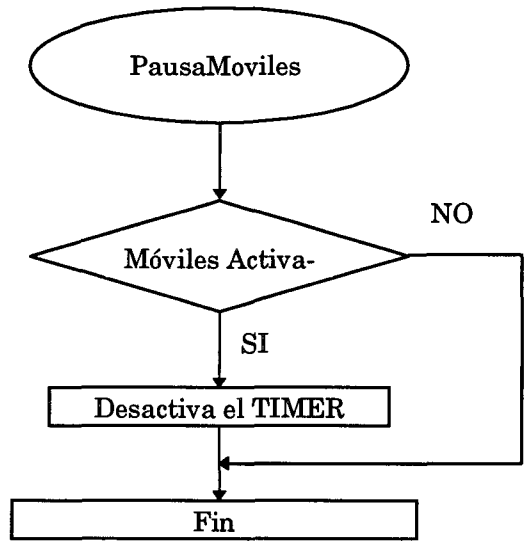
Esta función se encarga de gestionar el movimiento de los móviles, ya sea cuando se desplazan o cuando han alcanzado su posición final. No necesita parámetros.



```
void TMainWindow :: ActivaMoviles ()
{
        for (BYTE by = 0; by < NumMoviles; by++)
        {
                ptAnterior[by] = ptActual[by];
                if ((ptActual[by].x == ptFinal[by].x) && (ptActual[by].y == ptFinal[by].y))
                {
                        if (by == byMovilSeleccionado )
                        {
                                if (MovilBuffer.wMovimientoAleatorio == BF_CHECKED)
                                        CreaPuntoAleatorio(ptFinal[by]);
                        }
                        else
                                CreaPuntoAleatorio(ptFinal[by]);
                }
                else
                {
                        DesplazaMovil(ptActual[by], ptFinal[by]);
                        if (PosicionOcupada(ptActual[by], by))
                        {
                                ptActual[by] = ptAnterior[by];
                                if (by == byMovilSeleccionado )
                                {
                                        if (MovilBuffer.wMovimientoAleatorio ==
BF_CHECKED)
                                                CreaPuntoAleatorio(ptFinal[by]);
                                }
                                else
                                        CreaPuntoAleatorio(ptFinal[by]);
                        }
                }
        }
        for ( by = 0; by < NumMoviles; by++)
                BorraMovil(ptAnterior[by], by);
        LocalizaMovil(ptActual[byMovilSeleccionado]);
        for ( by = 0; by < NumMoviles; by++)
                DibujaMovil(ptActual[by], by);
}
```

void PausaMoviles ()

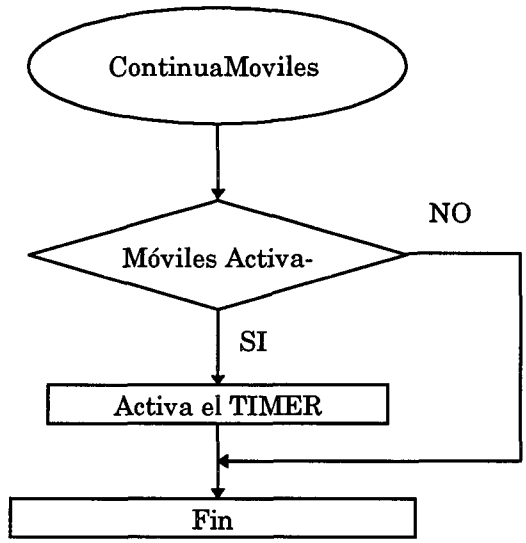
Esta función desactiva el timer que permite el desplazamiento de los móviles en pantalla. Se consigue retener a los móviles en situaciones en las que es conveniente para evitar errores cuando estamos trabajando con otras ventanas distintas a la principal.




```
void TMainWindow :: PausaMoviles ()
{
        if (bMovilesActivados)
                KillTimer(HWindow, TIMER_MOVIL);
}
```

void ContinuaMoviles ()

Esta función se utiliza para activar el timer después de haber utilizado la función PausaMoviles(). Se suele encontrar al final de las funciones que abren un cuadro de diálogo, mientras que la función PausaMoviles() la encontraremos al principio.



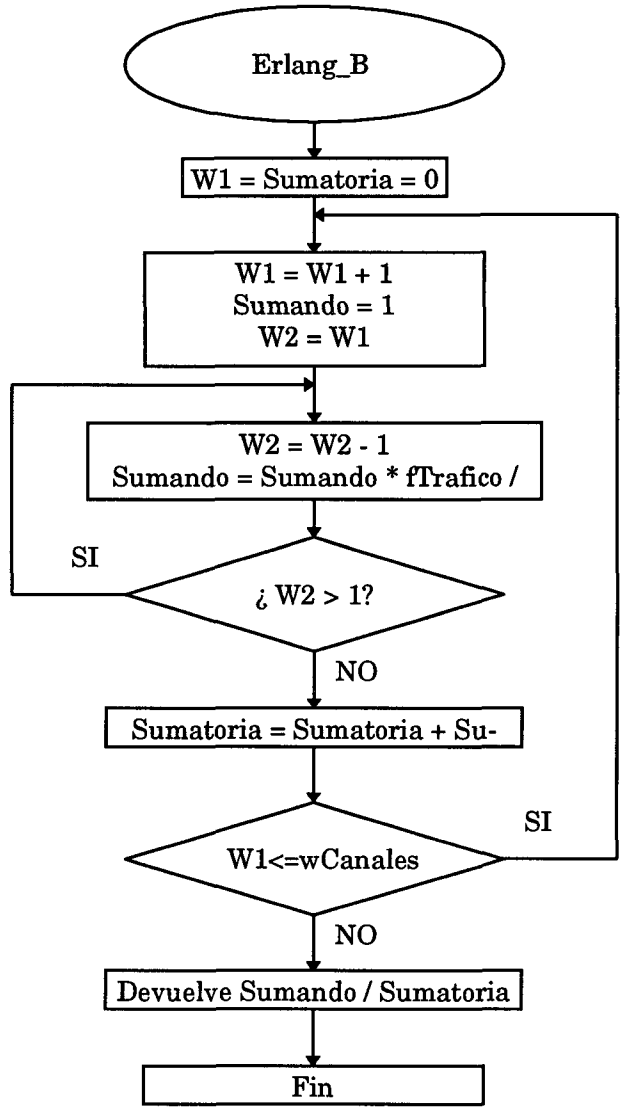

```
void TMainWindow :: ContinuaMoviles ()
{
        if (bMovilesActivados)
                SetTimer(HWindow, TIMER_MOVIL, wTiempo, NULL);
}
```

float Erlang_B (float fTrafico, WORD wCanales)

Esta función es la encargada de calcular la probabilidad de bloqueo de una llamada utilizando la fórmula B de Erlang. Necesita dos parámetros por valor y devuelve un número en coma flotante que contendrá el resultado.

fTrafico: Tráfico ofrecido en una zona.

wCanales: Número de canales disponibles en dicha zona.

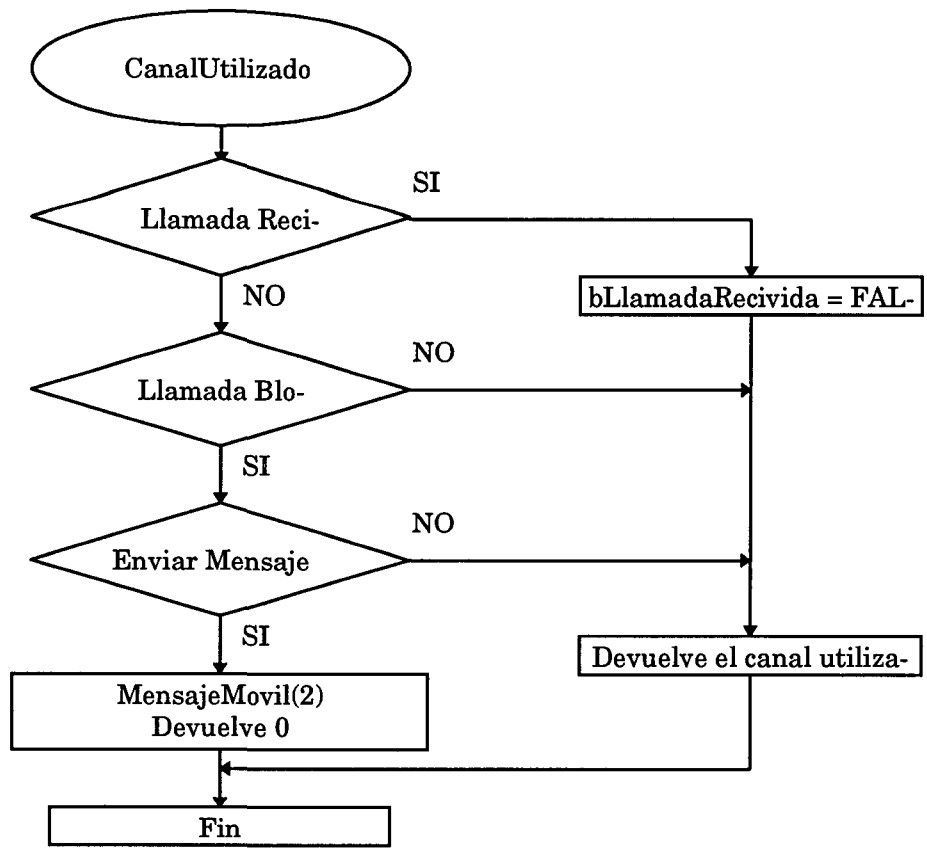



```
float TMainWindow :: Erlang_B (float fTrafico, WORD wCanales)
{
        WORD w1,w2;
        long double ldSumando, ldSumatoria = 0;
        for (w1 = 0; w1 <= wCanales; w1++)
        {
                ldSumando = 1;
                for (w2 = w1; w2 > 1; w2--)
                        ldSumando *= fTrafico / w2;
                ldSumatoria += ldSumando;
        }
        return ldSumando / ldSumatoria;
}
```

WORD CanalUtilizado (BYTE byCelulaActual)

Esta función calcula de forma aleatoria el canal utilizado por el móvil seleccionado en cada una de las células a las que se desplace, teniendo en cuenta el número de célula que ocupa dentro del grupo al que pertenece evitando así que se puedan repetir los canales en células pertenecientes a un mismo grupo. Devuelve un entero sin signo donde se encuentra el canal utilizado. Utiliza un parámetro por valor:

byCelulaActual: Contiene el número de célula dentro de su grupo.



```
WORD TMainWindow :: CanalUtilizado (BYTE byCelulaActual)
{
        if (bLlamadaRecibida)
                bLlamadaRecibida = FALSE;
        else
        {
                BYTE byBloqueo = random(101);
                if (byBloqueo < 100*fBloqueo)
                {
                        if (MovilBuffer.wMsgCanalBloqueado == BF_CHECKED)
                                MensajeMovil(2);
                                return 0;
                }
        }
        return random(wCanalesCelula-byCelulasGrupo+1)+byCelulaActual;
}
```

4.3 La clase MovilDialog

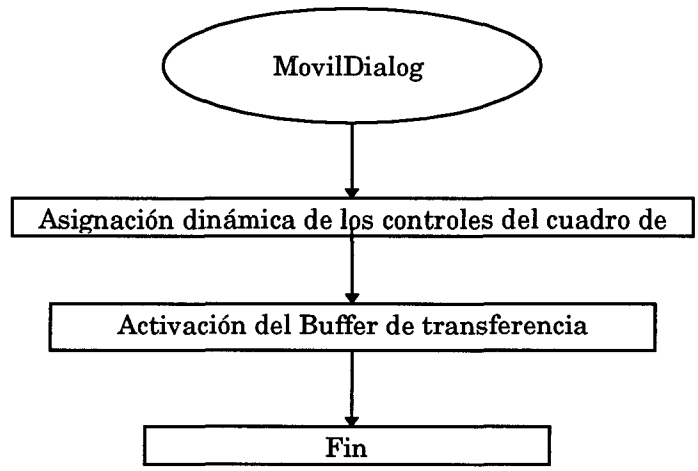
Esta clase añade el cuadro de diálogo **Seleccionar Móvil...** a nuestra aplicación. Desciende de la clase TDialog y se encarga de definir las funciones miembro que procesan los mensajes del cuadro de diálogo.

```
class MovilDialog : public TDialog
{
protected:
        //Punteros a los controles del cuadro de diálogo.
        PTGroupBox pMovimiento;
        PTGroupBox pMovilSeleccionado;
        PTGroupBox pVelocidad;

public:
        //Constructor del cuadro de diálogo.
        MovilDialog (PTWindowsObject AParent, LPSTR DName);
};
```

4.3.1 Las funciones de MovilDialog

MovilDialog (PTWindowsObject AParent, LPSTR DName)

Este es el constructor de la clase MovilDialog, en él se creará el buffer de transferencia de datos entre la ventana principal y el cuadro de diálogo **Seleccionar Móviles...**




```
MovilDialog :: MovilDialog (PTWindowsObject AParent, LPSTR DName) : TDialog(AParent,
DName)
{
        pMovimiento = new TGroupBox(this, ID_MOVIMIENTO);
        new TRadioButton(this, ID_MOVIMIENTOALEATORIO, pMovimiento);
        new TRadioButton(this, ID_MOVIMIENTORATON, pMovimiento);
        new TCheckBox(this, ID_MSGCANALBLOQUEADO, NULL);
        new TCheckBox(this, ID_MSGSALIDAZONA, NULL);
        pMovilSeleccionado = new TGroupBox(this, ID_MOVILSELECCIONADO);
        new TRadioButton(this, ID_MOVILAMARILLO, pMovilSeleccionado);
        new TRadioButton(this, ID_MOVILAZUL, pMovilSeleccionado);
        new TRadioButton(this, ID_MOVILNEGRO, pMovilSeleccionado);
        new TRadioButton(this, ID_MOVILROJO, pMovilSeleccionado);
        new TRadioButton(this, ID_MOVILVERDE, pMovilSeleccionado);
        pVelocidad = new TGroupBox(this, ID_VELOCIDAD);
        new TRadioButton(this, ID_VELOCIDADMAXIMA, pVelocidad);
        new TRadioButton(this, ID_VELOCIDADMEDIA, pVelocidad);
        new TRadioButton(this, ID_VELOCIDADMINIMA, pVelocidad);
        new TRadioButton(this, ID_RECIBIRLLAMADAS, NULL);
        TransferBuffer = (void far*)&(((TMainWindow *)Parent)->MovilBuffer);
}
```

4.4 La clase GeomeDialog

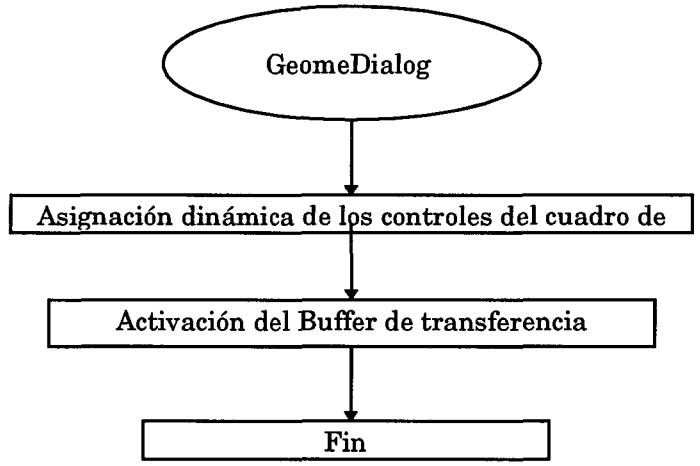
Esta clase añade el cuadro de diálogo **Magnitudes Geométricas...** a nuestra aplicación. Desciende de la clase TDialog y se encarga de definir las funciones miembro que procesan los mensajes del cuadro de diálogo.

```
class GeomeDialog : public TDialog
{
protected:
        //Punteros a los controles del cuadro de diálogo.
        PTEdit pRadioCelula;
        PTEdit pAreaTotal;
        PTGroupBox pNumCelulas;

public:
        //Constructor del cuadro de diálogo.
        GeomeDialog (PTWindowsObject AParent, LPSTR DName);

        //Función de control del cuadro de edición "Radio de cobertura de las células".
        virtual void IDRadioCelula (RTMessage Msg) = [ID_FIRST + ID_RADIOCELULA];

        //Función de control del cuadro de edición "Area total de la zona a cubrir".
        virtual void IDAreaTotal (RTMessage Msg) = [ID_FIRST + ID_AREATOTAL];

private:
        //Función amiga de la función que limita la introducción de datos en los cuadros de
edición.
        friend void CompruebaDato (PTEdit p, WORD wMensaje, float fLimiteInferior,
                float fLimiteSuperior);
};
```

4.4.1 Las funciones de GeomeDialog

GeomeDialog (PTWindowsObject AParent, LPSTR DName)

Este es el constructor de la clase MovilDialog, en él se creará el buffer de transferencia de datos entre la ventana principal y el cuadro de diálogo **Magnitudes Geométricas...**




```
GeomeDialog :: GeomeDialog (PTWindowsObject AParent, LPSTR DName) : TDia-
log(AParent, DName)
{
        pRadioCelula = new TEdit(this, ID_RADIOCELULA, NumChar);
        pAreaTotal = new TEdit(this, ID_AREATOTAL, NumChar);
        pNumCelulas = new TGroupBox(this, ID_GRUPO);
        new TRadioButton(this, ID_GRUPO3CELULAS, pNumCelulas);
        new TRadioButton(this, ID_GRUPO4CELULAS, pNumCelulas);
        new TRadioButton(this, ID_GRUPO7CELULAS, pNumCelulas);
        new TRadioButton(this, ID_GRUPO12CELULAS, pNumCelulas);
        TransferBuffer = (void far*)&(((TMainWindow *)Parent)->GeomeBuffer);
}
```

void IDRadioCelula (RTMessage Msg)

void IDAreaTotal (RTMessage Msg)

void IDAnchoBanda (RTMessage Msg)

Estas funciones gestionan sus respectivos cuadros de edición realizando una llamada a la función amiga CompruebaDato() pasándole los parámetros adecuados.

```
void GeomeDialog :: IDRadioCelula (RTMessage Msg)
{
        CompruebaDato(pRadioCelula, Msg.LP.Hi);
}

void GeomeDialog :: IDAreaTotal (RTMessage Msg)
{
        CompruebaDato(pAreaTotal, Msg.LP.Hi);
}

void RadioDialog :: IDAnchoBanda (RTMessage Msg)
{
        CompruebaDato(pAnchoBanda, Msg.LP.Hi,);
}
```

4.5 La clase RadioDialog

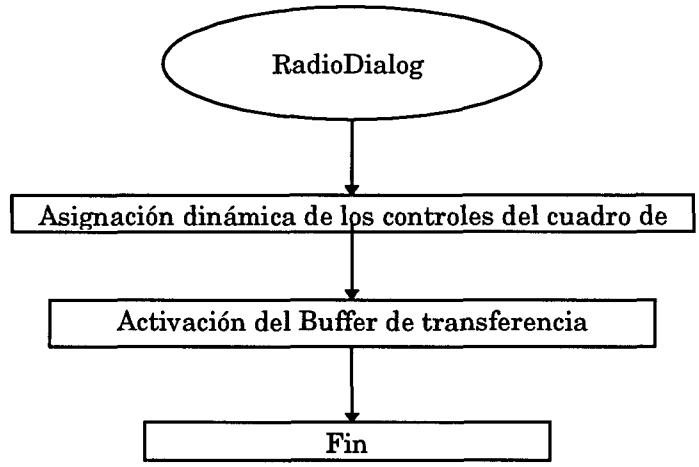
Esta clase añade el cuadro de diálogo **Magnitudes Radioeléctricas...** a nuestra aplicación. Desciende de la clase TDialog y se encarga de definir las funciones miembro que procesan los mensajes del cuadro de diálogo.

```
class RadioDialog : public TDialog
{
protected:
        //Punteros a los controles del cuadro de diálogo.
        PTEdit pAnchoBanda;
        PTEdit pSeparacionCanales;
        PTEdit pRelacionProteccion;
        PTEdit pLeyPropagacion;
        PTGroupBox pAntena;

public:
        //Constructor del cuadro de diálogo.
        RadioDialog (PTWindowsObject AParent, LPSTR DName);

        //Función de control del cuadro de edición "Ancho de banda disponible".
        virtual void IDAnchoBanda (RTMessage Msg) = [ID_FIRST + ID_ANCHOBANDA];
        //Función de control del cuadro de edición "Separación entre canales".
        virtual void IDSeparacionCanales (RTMessage Msg) = [ID_FIRST +
                ID_SEPARACIONCANALES];

        //Función de control del cuadro de edición "Relación de protección de RF".
        virtual void IDRelacionProteccion (RTMessage Msg) = [ID_FIRST +
                ID_RELACIONPROTECCION];

        //Función de control del cuadro de edición "Ley de propagación de la señal".
        virtual void IDLeyPropagacion (RTMessage Msg) = [ID_FIRST +
ID_LEYPROPAGACION];

private:
        //Función amiga de la función que limita la introducción de datos en los cuadros de
edición.
        friend void CompruebaDato (PTEdit p, WORD wMensaje, float fLimiteInferior,
                float fLimiteSuperior);
```

4.5.1 Las funciones de RadioDialog

RadioDialog (PTWindowsObject AParent, LPSTR DName)

Este es el constructor de la clase MovilDialog, en él se creará el buffer de transferencia de datos entre la ventana principal y el cuadro de diálogo **Magnitudes Radioeléctricas...**




```
RadioDialog :: RadioDialog (PTWindowsObject AParent, LPSTR DName) : TDialog(AParent,
DName)
{
        pAnchoBanda = new TEdit(this, ID_ANCHOBANDA, NumChar);
        pSeparacionCanales = new TEdit(this, ID_SEPARACIONCANALES, NumChar);
        pRelacionProteccion = new TEdit(this, ID_RELACIONPROTECCION, NumChar);
        pLeyPropagacion = new TEdit(this, ID_LEYPROPAGACION, NumChar);
        pAntena = new TGroupBox(this, ID_ANTENAS);
        new TRadioButton(this, ID_ANTENAOMNIDIRECCIONAL, pAntena);
        new TRadioButton(this, ID_ANTENASECTORIAL120, pAntena);
        new TRadioButton(this, ID_ANTENASECTORIAL60, pAntena);
        TransferBuffer = (void far*)&(((TMainWindow *)Parent)->RadioBuffer);
}
```

void IDSeparacionCanales (RTMessage Msg)

void IDRelacionProteccion (RTMessage Msg)

void IDLeyPropagacion (RTMessage Msg)

Estas funciones gestionan sus respectivos cuadros de edición realizando una llamada a la función amiga CompruebaDato() pasándole los parámetros adecuados.

```
void RadioDialog :: IDSeparacionCanales (RTMessage Msg)
{
        CompruebaDato(pSeparacionCanales, Msg.LP.Hi);
}

void RadioDialog :: IDRelacionProteccion (RTMessage Msg)
{
        CompruebaDato(pRelacionProteccion, Msg.LP.Hi);
}

void RadioDialog :: IDLeyPropagacion (RTMessage Msg)
{
        CompruebaDato(pLeyPropagacion, Msg.LP.Hi);
}
```

4.6 La clase TraficoDialog

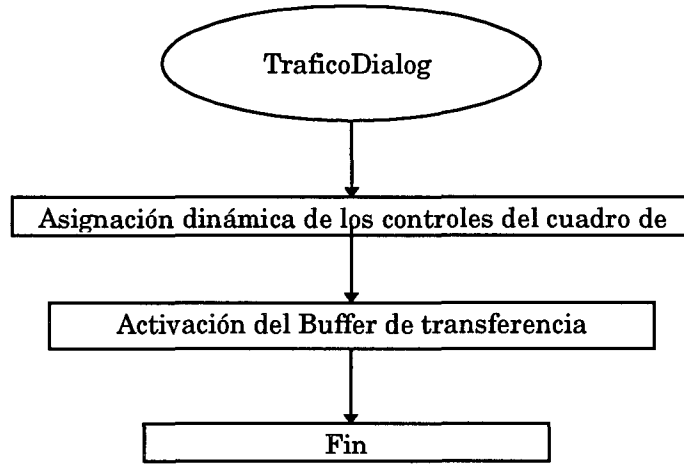
Esta clase añade el cuadro de diálogo **Magnitudes de Tráfico...** a nuestra aplicación. Desciende de la clase TDialog y se encarga de definir las funciones miembro que procesan los mensajes del cuadro de diálogo.

```
class TraficoDialog : public TDialog
{
protected:
        //Puntero a los controles del cuadro de diálogo.
        PTEdit pDensidadMoviles;

public:
        //Constructor del cuadro de diálogo.
        TraficoDialog (PTWindowsObject AParent, LPSTR DName);

        //Función de control del cuadro de edición "Densidad de móviles requerida".
        virtual void IDDensidadMoviles (RTMessage Msg) = [ID_FIRST +
ID_DENSIDADMOVILES];

private:
        //Función amiga de la función que limita la introducción de datos en los cuadros de
edición.
        friend void CompruebaDato (PTEdit p, WORD wMensaje);
```

4.6.1 Las funciones de TraficoDialog

TraficoDialog (PTWindowsObject AParent, LPSTR DName)

Este es el constructor de la clase MovilDialog, en él se creará el buffer de transferencia de datos entre la ventana principal y el cuadro de diálogo **Magnitudes de Tráfico...**




```
TraficoDialog :: TraficoDialog (PTWindowsObject AParent, LPSTR DName) : TDia-
log(AParent, DName)
{
        pDensidadMoviles = new TEdit(this, ID_DENSIDADMOVILES, NumChar);
        TransferBuffer = (void far*)&(((TMainWindow *)Parent)->TraficoBuffer);
}
```

void IDDensidadMoviles (RTMessage Msg)

Esta función gestiona su cuadro de edición, realizando una llamada a la función amiga CompruebaDato(), pasándole los parámetros adecuados.

```
void TraficoDialog :: IDDensidadMoviles (RTMessage Msg)
{
        CompruebaDato(pDensidadMoviles, Msg.LP.Hi);
}
```

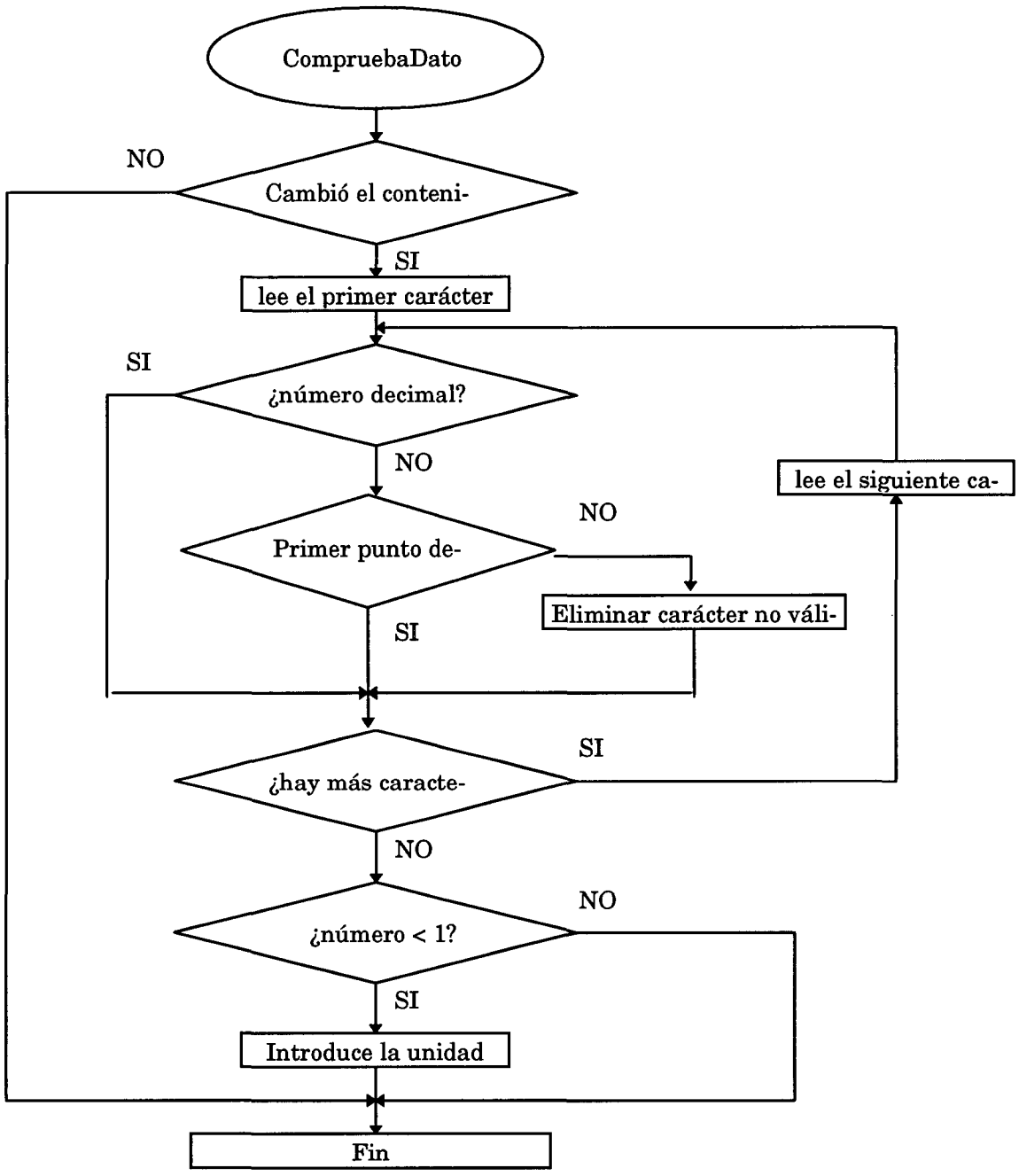
4.7 funciones especiales de TMA

4.7.1 void CompruebaDato (PTEdit p, WORD wMensaje)

Esta función controla la inserción de datos por parte del usuario en los cuadros de edición de la cajas de diálogo. Se declara como friend de las clases que lo utilizan para evitar repetir la definición de dicha función en cada una de ellas. Utiliza dos parámetros por valor:

p: Es un puntero a una caja de edición, que se utiliza para referenciar a la caja correspondiente.

wMensaje: Contiene el mensaje de notificación asociado a la caja de edición. Indica lo que ha ocurrido en la caja de edición.



```
void CompruebaDato (PTEdit p, WORD wMensaje)
{
        if (wMensaje == EN_CHANGE)
        {
                // el contenido del cuadro de edición ha cambiado.
                BYTE by, byLongitud = p->GetLineLength(0);
                BOOL bPuntoDecimal = FALSE;
                char sValor[NumChar];
                p->GetText(sValor, byLongitud+1);
                for (by = 0; by < byLongitud; by++)
                {
                        // comprueba el contenido carácter a carácter.
                        if ((isalpha(sValor[by])) || (!isalnum(sValor[by])))
                        {
                                // el carácter no es un número entero.
                                if ((!bPuntoDecimal) && (sValor[by] == '.'))
                                        bPuntoDecimal = TRUE;
                                else
                                {
                                        // el carácter no es válido.
                                        sValor[by]='0';
                                        p->SetText(sValor);
                                        p->SetSelection(by, by+1);
                                }
                        }
                }
                float fValor = atof(sValor);
                if (fValor < 1)
                {
                        // el valor es menor que la unidad.
                        gcvt(fLimiteInferior, NumChar-1, sValor);
                        p->SetText("1");
                        p->SetSelection(0, 1);
                }
        }
}
```

4.7.2 int WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR lpCmdLine, int nCmdShow)

Esta es la función de entrada de la aplicación, es la encargada de ejecutarla y de llamar a las funciones de inicialización, InitApplication() e InitInstance(). Por último llama a la función MessageLoop(), encargada de gestionar el bucle de mensajes.

```
int PASCAL WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance, LPSTR
lpCmdLine,
                    int nCmdShow)
{
        TApp App ("Planificación de un Sistema de Telefonía Móvil Celular",hInstance, hPre-
vInstance,
                    lpCmdLine, nCmdShow);
        App.Run();
        return App.Status;
```

ANEXO 1

LOS ARCHIVOS DE TMA.EXE

Para poder ejecutar esta aplicación, es necesario cargar en un mismo directorio los ficheros TMA.EXE y TMA.HLP. En este anexo encontraremos todo el código de los archivos que han sido utilizados para poder crear el ejecutable TMA.EXE. Dichos archivos se muestran a continuación:

* **TMA.ICO**
* **TMA.CUR**
* **TMA.DEF**
* **TMA.H**
* **TMA.RC**
* **TMA.CPP**

A1.1 TMA.ICO

Este fichero contiene el icono utilizado por el ejecutable de la aplicación TMA. Se utiliza para identificar y ejecutar el programa dentro del entorno de Microsoft Windows.

A1.2 TMA.CUR

Este archivo contiene la definición de un cursor que será utilizado por la aplicación. El cursor simboliza un teléfono móvil que estará presente en la aplicación cuando haya una llamada en curso.

A1.3 TMA.DEF

Este es el archivo de definición de módulo. Todas las aplicaciones Windows necesitan definir un archivo de definición de módulo donde se definen ciertos aspectos de la aplicación:

```
; Nombre del programa ejecutable "tma.exe".
NAME            TMA

; Información adicional para el archivo ejecutable "tma.exe".
DESCRIPTION "Planificador de TMA por Julio César Caballero Ortiz, 1995"

; Tipo de archivo ejecutable.
EXETYPE         WINDOWS

; Definición del segmento de código.
CODE            PRELOAD MOVEABLE DISCARDABLE

; Definición del segmento de datos.
DATA            PRELOAD MOVEABLE MULTIPLE

; Memeoria reservada del segmento de datos.
HEAPSIZE        5120

; Memoria reservada de la pila.
STACKSIZE       1024
```

A1.4 TMA.H

Este es el archivo de cabecera de la aplicación, donde definimos todos las constantes identificadoras utilizadas por la aplicación:

```
// Constante identificadora del timer utilizado
#define TIMER_MOVIL                    1

// Constantes identificaadoras de las opciones del menú principal
#define CM_VISUALIZARMOVILES           100
#define CM_BORRARMOVILES               101
#define CM_EFECTUARLLAMADA             102
#define CM_CORTARCOMUNICACION          103
#define CM_SELECCIONARMOVIL            104

#define CM_DIMENSIONES                 120
#define CM_RADIOELECTRICOS             121
#define CM_TRAFICO                     122
#define CM_MOVILES                     123

#define CM_MAGGEOMETRICAS              140
#define CM_MAGRADIOELECTRICAS          141
#define CM_MAGTRAFICO                  142

#define CM_SALIR                       160

#define CM_INDICE                      180
#define CM_AYUDAAREACLIENTE            181
#define CM_AYUDAMENU                   182
#define CM_AYUDARATON                  183
#define CM_AYUDASISTEMAS               184
#define CM_AYUDAMENSAJES               185
#define CM_BUSCARAYUDA                 186
#define CM_USOAYUDA                    187
#define CM_ACERCADE                    188
```

```
// Constantes identificadoras de los cuadros de diálogo
#define ID_MOVIMIENTO                   200
#define ID_MOVIMIENTOALEATORIO          201
#define ID_MOVIMIENTORATON              202
#define ID_MSGCANALBLOQUEADO            203
#define ID_MSGSALIDAZONA                204
#define ID_MOVILSELECCIONADO            205
#define ID_MOVILAMARILLO                206
#define ID_MOVILAZUL                    207
#define ID_MOVILNEGRO                   208
#define ID_MOVILROJO                    209
#define ID_MOVILVERDE                   210
#define ID_VELOCIDAD                    211
#define ID_VELOCIDADMAXIMA              212
#define ID_VELOCIDADMEDIA               213
#define ID_VELOCIDADMINIMA              214
#define ID_RECIBIRLLAMADAS              215

#define ID_RADIOCELULA                  220
#define ID_AREATOTAL                    221
#define ID_GRUPO                        222
#define ID_GRUPO3CELULAS                223
#define ID_GRUPO4CELULAS                224
#define ID_GRUPO7CELULAS                225
#define ID_GRUPO12CELULAS               226

#define ID_ANCHOBANDA                   240
#define ID_SEPARACIONCANALES            241
#define ID_RELACIONPROTECCION           242
#define ID_LEYPROPAGACION               243
#define ID_ANTENAS                      244
#define ID_ANTENAOMNIDIRECCIONAL        245
#define ID_ANTENASECTORIAL120           246
#define ID_ANTENASECTORIAL60            247

#define ID_DENSIDADMOVILES              260
```

A1.5 TMA.RC

Este es el archivo de definición de recursos de la aplicación. En él se encuentran declarados el cursor , el icono y los mapas de bits (bitmaps) de la aplicación, y definidos el menú y los cuadros de diálogo:

```
// Incluye las constantes identificadoras de la aplicación.
#include        "tma.h"

// Declaración del icono de la aplicación.
tmaIco          ICON            tma.ico

// Declaración del cursor definido por el programador.
tmaCur          CURSOR          tma.cur

// Declaración de los mapas de bits utilizados.
amarilloBmp     BITMAP          amarillo.bmp
azulBmp         BITMAP          azul.bmp
negroBmp        BITMAP          negro.bmp
rojoBmp         BITMAP          rojo.bmp
verdeBmp        BITMAP          verde.bmp

// Definición del menú de la aplicación.
tmaMenu MENU
BEGIN
        POPUP "&Móviles"
        BEGIN
                MENUITEM "&Visualizar Móviles", CM_VISUALIZARMOVILES
                MENUITEM "&Borrar Móviles", CM_BORRARMOVILES, GRAYED
                MENUITEM SEPARATOR
                MENUITEM "&Efectuar llamada telefónica", CM_EFECTUARLLAMADA, GRAYED
                MENUITEM "&Cortar la comunicación", CM_CORTARCOMUNICACION, GRAYED,
                        CHECKED
                MENUITEM SEPARATOR
                MENUITEM "&Seleccionar Móvil...", CM_SELECCIONARMOVIL
        END

        POPUP "&Información"
        BEGIN
                MENUITEM "Cálculo de &Dimensiones", CM_DIMENSIONES
                MENUITEM "Cálculos &Radioeléctricos", CM_RADIOELECTRICOS
                MENUITEM "Cálculos de &Trafico", CM_TRAFICO
                MENUITEM "Cálculos sobre los &Móviles", CM_MOVILES
        END
```

```
        POPUP "&Planificación Celular"
        BEGIN
                MENUITEM "Magnitudes &Geométricas...", CM_MAGGEOMETRICAS
                MENUITEM "Magnitudes &Radioeléctricas...", CM_MAGRADIOELECTRICAS
                MENUITEM "Magnitudes de &Tráfico...", CM_MAGTRAFICO
        END

        MENUITEM "&Salir", CM_SALIR
        POPUP "&Ayuda", HELP
        BEGIN
                MENUITEM "&Indice\tF1", CM_INDICE
                MENUITEM SEPARATOR
                MENUITEM "&Area de Cliente", CM_AYUDAAREACLIENTE
                MENUITEM "&Menú principal", CM_AYUDAMENU
                MENUITEM "&Ratón", CM_AYUDARATON
                MENUITEM "M&ensajes de información", CM_AYUDAMENSAJES
                MENUITEM "&Sistemas Celulares", CM_AYUDASISTEMAS
                MENUITEM SEPARATOR
                MENUITEM "&Buscar ayuda sobre...", CM_BUSCARAYUDA
                MENUITEM "&Uso de la ayuda", CM_USOAYUDA
                MENUITEM SEPARATOR
                MENUITEM "Acerca de &TMA...", CM_ACERCADE
        END

END

// Definición de los aceleradores de Menú o teclas rápidas.
tmaMenu ACCELERATORS
{
VK_F1, CM_INDICE, VIRTKEY
}



// Definición del cuadro de diálogo "Acerca de TMA...".
acercadeDlg DIALOG 70, 45, 180, 148
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Acerca de TMA"
FONT 8, "System"
BEGIN
        CTEXT "Planificación de Telefonía Móvil Automática ", -1, 19, 7, 142, 8
        CTEXT "EUITT de la Universidad de Las Palmas de GranCanaria, año 1995 ", -1, 22, 55, 135, 19
        PUSHBUTTON "&Aceptar", IDOK, 67, 80, 45, 14
        ICON "tmaIco", -1, 82, 21, 16, 16, WS_CHILD | WS_VISIBLE
        CONTROL "Tutora:      Itziar Goretti Alonso González", -1, "STATIC", SS_LEFT | WS_CHILD |
                WS_VISIBLE | WS_GROUP, 22, 127, 135, 8
        LTEXT "Autor:       Julio César Caballero Ortiz", -1, 22, 112, 135, 8
        LTEXT "  ----------Proyecto Fin de Carrera ----------", -1, 7, 42, 165, 8
        CONTROL "", 101, "button", BS_GROUPBOX | WS_CHILD | WS_VISIBLE, 7, 100, 165, 42
END
```

```
// Definición del cuadro de diálogo "Seleccionar Móvil...".
movilDlg DIALOG 60, 25, 201, 175
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Seleccionar movil"
FONT 8, "System"
BEGIN
        CONTROL "Movimiento: ", ID_MOVIMIENTO, "button", BS_GROUPBOX | WS_CHILD |
                WS_VISIBLE | WS_GROUP, 6, 4, 90, 50
        CONTROL "Aleatorio", ID_MOVIMIENTOALEATORIO, "BUTTON", BS_AUTORADIOBUTTON
                | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 14, 19, 45, 12
        CONTROL "Controlado por ratón", ID_MOVIMIENTORATON, "BUTTON",
                BS_AUTORADIOBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 14, 34, 75, 12
        CONTROL "Mensajes en pantalla: ", -1, "button", BS_GROUPBOX | WS_CHILD | WS_VISIBLE |
                WS_GROUP, 104, 4, 90, 50
        CONTROL "Canal Bloqueado", ID_MSGCANALBLOQUEADO, "BUTTON",
                BS_AUTOCHECKBOX | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 112, 19, 75, 12
        CONTROL "Salida de la zona", ID_MSGSALIDAZONA, "BUTTON", BS_AUTOCHECKBOX |
                WS_CHILD | WS_VISIBLE | WS_TABSTOP, 112, 34, 75, 12
        CONTROL "Móvil seleccionado: ", ID_MOVILSELECCIONADO, "button", BS_GROUPBOX |
                WS_CHILD | WS_VISIBLE | WS_GROUP, 6, 58, 90, 95
        CONTROL "Amarillo", ID_MOVILAMARILLO, "BUTTON", BS_AUTORADIOBUTTON |
                WS_CHILD | WS_VISIBLE | WS_TABSTOP, 14, 73, 35, 12
        CONTROL "Azul", ID_MOVILAZUL, "BUTTON", BS_AUTORADIOBUTTON | WS_CHILD |
                WS_VISIBLE | WS_TABSTOP, 14, 88, 35, 12
        CONTROL "Negro", ID_MOVILNEGRO, "BUTTON", BS_AUTORADIOBUTTON | WS_CHILD |
                WS_VISIBLE | WS_TABSTOP, 14, 103, 35, 12
        CONTROL "Rojo", ID_MOVILROJO, "BUTTON", BS_AUTORADIOBUTTON | WS_CHILD |
                WS_VISIBLE | WS_TABSTOP, 14, 118, 35, 12
        CONTROL "Verde", ID_MOVILVERDE, "BUTTON", BS_AUTORADIOBUTTON | WS_CHILD |
                WS_VISIBLE | WS_TABSTOP, 14, 133, 35, 12
        CONTROL "Velocidad: ", ID_VELOCIDAD, "button", BS_GROUPBOX | WS_CHILD |
                WS_VISIBLE | WS_GROUP, 104, 58, 90, 65
        CONTROL "Máxima", ID_VELOCIDADMAXIMA, "BUTTON", BS_AUTORADIOBUTTON |
                WS_CHILD | WS_VISIBLE | WS_TABSTOP, 112, 73, 35, 12
        CONTROL "Media", ID_VELOCIDADMEDIA, "BUTTON", BS_AUTORADIOBUTTON |
                WS_CHILD | WS_VISIBLE | WS_TABSTOP, 112, 88, 35, 12
        CONTROL "Mínima", ID_VELOCIDADMINIMA, "BUTTON", BS_AUTORADIOBUTTON |
                WS_CHILD | WS_VISIBLE | WS_TABSTOP, 112, 103, 35, 12
        CONTROL "Recibir Llamadas", ID_RECIBIRLLAMADAS, "BUTTON", BS_AUTOCHECKBOX |
                WS_CHILD | WS_VISIBLE | WS_TABSTOP, 14, 157, 75, 12
        DEFPUSHBUTTON "&Aceptar", IDOK, 122, 132, 60, 14, WS_CHILD | WS_VISIBLE |
                WS_TABSTOP
        PUSHBUTTON "&Cancelar", IDCANCEL, 122, 154, 60, 14, WS_CHILD | WS_VISIBLE |
                WS_TABSTOP
END
```

```
// Definición del cuadro de diálogo "Magnitudes Geométricas...".
geomeDlg DIALOG 60, 60, 196, 107
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Magnitudes Geométricas"
FONT 8, "System"
BEGIN
        LTEXT "Radio de cobertura de las células :", -1, 12, 12, 115, 12
        LTEXT "Area  total de la zona a cubrir :", -1, 12, 32, 115, 12
        EDITTEXT ID_RADIOCELULA, 127, 12, 30, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
                WS_BORDER | WS_TABSTOP
        EDITTEXT ID_AREATOTAL, 127, 32, 30, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
                WS_BORDER | WS_TABSTOP
        LTEXT "Km", -1, 173, 12, 16, 12
        LTEXT "Km\262", -1, 173, 32, 16, 12
        CONTROL "Células por grupo :", ID_GRUPO, "button", BS_GROUPBOX | WS_CHILD |
                WS_VISIBLE | WS_GROUP, 4, 50, 116, 50
        CONTROL "3 Células", ID_GRUPO3CELULAS, "BUTTON", BS_AUTORADIOBUTTON |
                WS_CHILD | WS_VISIBLE | WS_TABSTOP, 12, 65, 45, 12
        CONTROL "4 Células", ID_GRUPO4CELULAS, "BUTTON", BS_AUTORADIOBUTTON |
                WS_CHILD | WS_VISIBLE | WS_TABSTOP, 12, 80, 45, 12
        CONTROL "7 Células", ID_GRUPO7CELULAS, "BUTTON", BS_AUTORADIOBUTTON |
                WS_CHILD | WS_VISIBLE | WS_TABSTOP, 61, 65, 45, 12
        CONTROL "12 Células", ID_GRUPO12CELULAS, "BUTTON", BS_AUTORADIOBUTTON |
                WS_CHILD | WS_VISIBLE | WS_TABSTOP, 61, 80, 45, 12
        DEFPUSHBUTTON "&Aceptar", IDOK, 135, 53, 45, 18
        PUSHBUTTON "&Cancelar", IDCANCEL, 135, 81, 45, 17
END



// Definición del cuadro de diálogo "Magnitudes de Tráfico...".
traficoDlg DIALOG 55, 70, 211, 96
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Magnitudes de tráfico"
FONT 8, "System"
BEGIN
        LTEXT "Densidad de móviles requerida :", -1, 4, 14, 106, 12
        EDITTEXT ID_DENSIDADMOVILES, 118, 14, 30, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
                WS_BORDER | WS_TABSTOP
        LTEXT "moviles / km\262", -1, 158, 14, 55, 8
        DEFPUSHBUTTON "&Aceptar", IDOK, 30, 65, 60, 15, WS_CHILD | WS_VISIBLE |
                WS_TABSTOP
        PUSHBUTTON "&Cancelar", IDCANCEL, 120, 65, 60, 15, WS_CHILD | WS_VISIBLE |
                WS_TABSTOP
END
```

```
// Definición del cuadro de diálogo "Magnitudes Radioeléctricas...".
RadioDlg DIALOG 60, 40, 207, 158
STYLE DS_MODALFRAME | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Magnitudes Radioeléctricas"
FONT 8, "System"
BEGIN
        LTEXT "Ancho de banda disponible :", -1, 15, 4, 92, 12
        LTEXT "Separación entre canales :", -1, 15, 24, 92, 12
        LTEXT "Relación de protección de RF : ", -1, 15, 44, 120, 11
        LTEXT "Ley de propagación de la señal : ", -1, 15, 64, 120, 12
        EDITTEXT ID_ANCHOBANDA, 139, 4, 30, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
                WS_BORDER | WS_TABSTOP
        EDITTEXT ID_SEPARACIONCANALES, 139, 24, 30, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
                WS_BORDER | WS_TABSTOP
        EDITTEXT ID_RELACIONPROTECCION, 139, 44, 30, 12, ES_LEFT | WS_CHILD | WS_VISIBLE
                | WS_BORDER | WS_TABSTOP
        EDITTEXT ID_LEYPROPAGACION, 139, 64, 30, 12, ES_LEFT | WS_CHILD | WS_VISIBLE |
                WS_BORDER | WS_TABSTOP
        LTEXT "Mhz", -1, 177, 4, 16, 12
        LTEXT "Khz", -1, 177, 24, 16, 12
        LTEXT "dB", -1, 177, 44, 16, 12
        CONTROL "Antenas de estación de Base: ", ID_ANTENAS, "button", BS_GROUPBOX |
                WS_CHILD | WS_VISIBLE | WS_GROUP, 7, 80, 115, 73
        CONTROL "Omnidireccionales 360\272", ID_ANTENAOMNIDIRECCIONAL, "BUTTON",
                BS_AUTORADIOBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 15, 95, 90, 12
        CONTROL "Sectoriales de 120\272", ID_ANTENASECTORIAL120, "BUTTON",
                BS_AUTORADIOBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 15, 110, 90, 12
        CONTROL "Sectoriales de 60\272", ID_ANTENASECTORIAL60, "BUTTON",
                BS_AUTORADIOBUTTON | WS_CHILD | WS_VISIBLE | WS_TABSTOP, 15, 125, 90, 12
        PUSHBUTTON "&Aceptar", IDOK, 138, 93, 60, 19
        PUSHBUTTON "&Cancelar", IDCANCEL, 138, 125, 60, 18
END
```

A1.6 TMA.CPP

Este es el archivo fuente de la aplicación, donde se encuentra todo el código necesario para realizar sus funciones específicas:

```
/************************
Definición de constantes.
************************/

#define        NumChar          7
#define        NumMoviles       5
#define        xBmp             15
#define        yBmp             8
#define        TraficoMovil     0.03
#define        WIN31

/***********************************************
Ficheros de la librería de BORLAND C++ utilizados.
***********************************************/

#include       <button.h>
#include       <checkbox.h>
#include       <ctype.h>
#include       <edit.h>
#include       <dialog.h>
#include       <groupbox.h>
#include       <math.h>
#include       <owl.h>
#include       <radiobut.h>
#include       <static.h>
#include       <stdio.h>
#include       <stdlib.h>
#include       <string.h>
#include       <time.h>
#include       "tma.h"

/***********************************************
Referencia a las clases de ventana de la aplicación.
***********************************************/

_CLASSDEF (TMainWindow)
_CLASSDEF (MovilDialog)
_CLASSDEF (GeomeDialog)
_CLASSDEF (RadioDialog)
_CLASSDEF (TraficoDialog)
```

```
/*************************************************************************
Buffer de transferencia de datos del cuadro de diálogo "Seleccionar móvil...".
*************************************************************************/

struct MovilTransferBuffer
{
        WORD  wMovimientoAleatorio;
        WORD  wMovimientoRaton;
        WORD  wMsgCanalBloqueado;
        WORD  wMsgSalidaZona;
        WORD  wMovilAmarillo;
        WORD  wMovilAzul;
        WORD  wMovilNegro;
        WORD  wMovilRojo;
        WORD  wMovilVerde;
        WORD  wVelocidadMaxima;
        WORD  wVelocidadMedia;
        WORD  wVelocidadMinima;
        WORD  wRecibirLlamadas;
};

/*************************************************************************
Buffer de transferencia de datos del cuadro de diálogo "Magnitudes Geométricas...".
*************************************************************************/

struct GeomeTransferBuffer
{
        char    sRadioCelula[NumChar];
        char    sAreaTotal[NumChar];
        WORD  wGrupo3Celulas;
        WORD  wGrupo4Celulas;
        WORD  wGrupo7Celulas;
        WORD  wGrupo12Celulas;
};

/*************************************************************************
Buffer de transferencia de datos del cuadro de diálogo "Magnitudes Radioeléctricas...".
*************************************************************************/

struct RadioTransferBuffer
{
        char    sAnchoBanda[NumChar];
        char    sSeparacionCanales[NumChar];
        char    sRelacionProteccion[NumChar];
        char    sLeyPropagacion[NumChar];
        WORD  wAntenaOmnidireccional;
        WORD  wAntenaSectorial120;
        WORD  wAntenaSectorial60;
};
```

```
/*****************************************************************************
Buffer de transferencia de datos del cuadro de diálogo "Magnitudes de tráfico...".
*****************************************************************************/

struct TraficoTransferBuffer
{
        char    sDensidadMoviles[NumChar];
};

/*********************************************************************
Función que controla la introducción de datos en los cuadros de edición.
        p -> Puntero a un cuadro de edición.
        wMensaje -> Recibe los mensajes enviados al cuadro de edición.
*********************************************************************/

void CompruebaDato (PTEdit p, WORD wMensaje);

/**************************
Definición de la aplicación.
**************************/

class TApp : public TApplication
{
public:

/*****************************************************
Declaración y definición del constructor de la aplicación.
*****************************************************/

        TApp (LPSTR AName, HINSTANCE hInstance, HINSTANCE hPrevInstance,
                LPSTR lpCmdLine, int nCmdShow) : TApplication(AName, hInstance,
                hPrevInstance, lpCmdLine, nCmdShow) {};

protected:

/*****************************************************
Crea e inicializa una instancia de la ventana principal.
*****************************************************/

        virtual void InitMainWindow ();
};
```

```
/************************************************
Definición del cuadro de diálogo "Seleccionar movil".
************************************************/

class MovilDialog : public TDialog
{
protected:

/*****************************************
Punteros a los controles del cuadro de diálogo.
*****************************************/

        PTGroupBox pMovimiento;
        PTGroupBox pMovilSeleccionado;
        PTGroupBox pVelocidad;

public:

/*********************************
Constructor del cuadro de diálogo.
*********************************/

        MovilDialog (PTWindowsObject AParent, LPSTR DName);
};

/************************************************************
Definición del cuadro de diálogo "Magnitudes Geométricas...".
************************************************************/

class GeomeDialog : public TDialog
{
protected:

/*********************************************
Punteros a los controles del cuadro de diálogo.
*********************************************/

        PTEdit pRadioCelula;
        PTEdit pAreaTotal;
        PTGroupBox pNumCelulas;

public:

/*********************************
Constructor del cuadro de diálogo.
*********************************/

        GeomeDialog (PTWindowsObject AParent, LPSTR DName);

/*************************************************************************
Función de control del cuadro de edición "Radio de cobertura de las células".
*************************************************************************/

        virtual void IDRadioCelula (RTMessage Msg) = [ID_FIRST + ID_RADIOCELULA];
```

```
/************************************************************************
Función de control del cuadro de edición "Area total de la zona a cubrir".
************************************************************************/

        virtual void IDAreaTotal (RTMessage Msg) = [ID_FIRST + ID_AREATOTAL];

private:

/*****************************************************************************************
Función amiga de la función que limita la introducción de datos en los cuadros de edición.
*****************************************************************************************/

        friend void CompruebaDato (PTEdit p, WORD wMensaje);
};

/*****************************************************************
Definición del cuadro de diálogo "Magnitudes Radioeléctricas...".
*****************************************************************/

class RadioDialog : public TDialog
{
protected:

/*********************************************
Punteros a los controles del cuadro de diálogo.
*********************************************/

        PTEdit pAnchoBanda;
        PTEdit pSeparacionCanales;
        PTEdit pRelacionProteccion;
        PTEdit pLeyPropagacion;
        PTGroupBox pAntena;

public:

/*********************************
Constructor del cuadro de diálogo.
*********************************/

        RadioDialog (PTWindowsObject AParent, LPSTR DName);

/*****************************************************************
Función de control del cuadro de edición "Ancho de banda disponible".
*****************************************************************/

        virtual void IDAnchoBanda (RTMessage Msg) = [ID_FIRST + ID_ANCHOBANDA];

/*****************************************************************
Función de control del cuadro de edición "Separación entre canales".
*****************************************************************/

        virtual void IDSeparacionCanales (RTMessage Msg) = [ID_FIRST + ID_SEPARACIONCANALES];
```

```
/*************************************************************************
Función de control del cuadro de edición "Relación de protección de RF".
*************************************************************************/

        virtual void IDRelacionProteccion (RTMessage Msg) = [ID_FIRST +
ID_RELACIONPROTECCION];

/*************************************************************************
Función de control del cuadro de edición "Ley de propagación de la señal".
*************************************************************************/

        virtual void IDLeyPropagacion (RTMessage Msg) = [ID_FIRST + ID_LEYPROPAGACION];

private:

/*****************************************************************************************
Función amiga de la función que limita la introducción de datos en los cuadros de edición.
*****************************************************************************************/

        friend void CompruebaDato (PTEdit p, WORD wMensaje);
};

/*********************************************************
Definición del cuadro de diálogo "Magnitudes de tráfico...".
*********************************************************/

class TraficoDialog : public TDialog
{
protected:

/***********************************************
Punteros a los controles del cuadro de diálogo.
***********************************************/

        PTEdit pDensidadMoviles;

public:

/*********************************
Constructor del cuadro de diálogo.
*********************************/

        TraficoDialog (PTWindowsObject AParent, LPSTR DName);

/*************************************************************************
Función de control del cuadro de edición "Densidad de móviles requerida".
*************************************************************************/

        virtual void IDDensidadMoviles (RTMessage Msg) = [ID_FIRST + ID_DENSIDADMOVILES];

private:
```

```
/*****************************************************************************
Función amiga de la función que limita la introducción de datos en los cuadros de edición.
*****************************************************************************/

        friend void CompruebaDato (PTEdit p, WORD wMensaje);
};

/************************************************
Definición de la ventana principal de la aplicación.
************************************************/

class TMainWindow : public TWindow
{
public:

/*************************************************
Variables los buffers de transferencia de datos.
*************************************************/

        MovilTransferBuffer     MovilBuffer;
        GeomeTransferBuffer     GeomeBuffer;
        RadioTransferBuffer     RadioBuffer;
        TraficoTransferBuffer   TraficoBuffer;


/*********************************
Constructor de la ventana principal.
*********************************/

        TMainWindow (PTWindowsObject AParent, LPSTR ATitle);

/*************************************
Inicialización de los menús desplegables.
*************************************/

        virtual void WMInitMenuPopup (RTMessage Msg) = [WM_FIRST + WM_INITMENUPOPUP];

/*********************************************
Gestión de las selecciones realizadas en un menú.
*********************************************/

        virtual void WMMenuSelect (RTMessage Msg) = [WM_FIRST + WM_MENUSELECT];

/*********************************************
Gestión de la apertura de la ventana principal.
*********************************************/

        virtual void WMCreate (RTMessage Msg) = [WM_FIRST + WM_CREATE];

/*****************************************************
Gestión de los cambios de dimensión de la ventana principal.
*****************************************************/

        virtual void WMSize (RTMessage Msg) = [WM_FIRST + WM_SIZE];
```

```
/*******************************
Gestión del movimiento del ratón.
*******************************/

        virtual void WMMouseMove (RTMessage Msg) = [WM_FIRST + WM_MOUSEMOVE];

/*****************************************************
Gestión de las pulsaciones del botón izquierdo del ratón.
*****************************************************/

        virtual void WMLButtonDown (RTMessage Msg) = [WM_FIRST + WM_LBUTTONDOWN];

/*****************************************************
Gestión de las pulsaciones del botón derecho del ratón.
*****************************************************/

        virtual void WMRButtonDown (RTMessage Msg) = [WM_FIRST + WM_RBUTTONDOWN];

/*********************************
Gestión de la activación del timer.
*********************************/

        virtual void WMTimer (RTMessage Msg) = [WM_FIRST + WM_TIMER];


/***********************************************
Gestión de la opción de menú "Visualizar móviles".
***********************************************/

        virtual void CMVisualizarMoviles (RTMessage Msg) = [CM_FIRST +
CM_VISUALIZARMOVILES];

/*******************************************
Gestión de la opción de menú "Borrar móviles".
*******************************************/

        virtual void CMBorrarMoviles (RTMessage Msg) = [CM_FIRST + CM_BORRARMOVILES];

/*********************************************************
Gestión de la opción de menú "Efectuar llamada telefónica".
*********************************************************/

        virtual void CMEfectuarLlamada (RTMessage Msg) = [CM_FIRST + CM_EFECTUARLLAMADA];

/*************************************************
Gestión de la opción de menú "Cortar la comunicación".
*************************************************/

        virtual void CMCortarComunicacion (RTMessage Msg) = [CM_FIRST +
CM_CORTARCOMUNICACION];
```

```
/***********************************************
Gestión de la opción de menú "Seleccionar móvil...".
***********************************************/

        virtual void CMSeleccionarMovil (RTMessage Msg) = [CM_FIRST + CM_SELECCIONARMOVIL];

/***********************************************
Gestión de la opción de menú "Cálculo de Dimensiones".
***********************************************/

        virtual void CMDimensiones (RTMessage Msg) = [CM_FIRST + CM_DIMENSIONES];

/***********************************************
Gestión de la opción de menú "Cálculos Radioeléctricos".
***********************************************/

        virtual void CMRadioelectricos (RTMessage Msg) = [CM_FIRST + CM_RADIOELECTRICOS];

/***********************************************
Gestión de la opción de menú "Cálculos de tráfico".
***********************************************/

        virtual void CMTrafico (RTMessage Msg) = [CM_FIRST + CM_TRAFICO];

/***********************************************
Gestión de la opción de menú "Cálculos sobre los móviles".
***********************************************/

        virtual void CMMoviles (RTMessage Msg) = [CM_FIRST + CM_MOVILES];
/***********************************************
Gestión de la opción de menú "Magnitudes geométricas...".
***********************************************/

        virtual void CMMagGeometricas (RTMessage Msg) = [CM_FIRST + CM_MAGGEOMETRICAS];

/***********************************************
Gestión de la opción de menú "Magnitudes radioeléctricas...".
***********************************************/

        virtual void CMMagRadioelectricas (RTMessage Msg) = [CM_FIRST +
CM_MAGRADIOELECTRICAS];

/***********************************************
Gestión de la opción de menú "Magnitudes de tráfico...".
***********************************************/

        virtual void CMMagTrafico (RTMessage Msg) = [CM_FIRST + CM_MAGTRAFICO];

/***********************************
Gestión de la opción de menú "Salir".
***********************************/

        virtual void CMSalir (RTMessage Msg) = [CM_FIRST + CM_SALIR];
```

```
/***********************************
Gestión de la opción de menú "Indice".
***********************************/

        virtual void CMIndice (RTMessage Msg) = [CM_FIRST + CM_INDICE];

/*****************************************
Gestión de la opción de menú "Area de Cliente".
*****************************************/

        virtual void CMAyudaAreaCliente (RTMessage Msg) = [CM_FIRST +
CM_AYUDAAREACLIENTE];

/*****************************************
Gestión de la opción de menú "Menú principal".
*****************************************/

        virtual void CMAyudaMenu (RTMessage Msg) = [CM_FIRST + CM_AYUDAMENU];

/***********************************
Gestión de la opción de menú "Ratón".
***********************************/

        virtual void CMAyudaRaton (RTMessage Msg) = [CM_FIRST + CM_AYUDARATON];

/*************************************************
Gestión de la opción de menú "Sistemas Celulares".
*************************************************/

        virtual void CMAyudaSistemas (RTMessage Msg) = [CM_FIRST + CM_AYUDASISTEMAS];
/*****************************************************
Gestión de la opción de menú "Mensajes de información".
*****************************************************/

        virtual void CMAyudaMensajes (RTMessage Msg) = [CM_FIRST + CM_AYUDAMENSAJES];

/*************************************************
Gestión de la opción de menú "Buscar Ayuda sobre...".
*************************************************/

        virtual void CMBuscarAyuda (RTMessage Msg) = [CM_FIRST + CM_BUSCARAYUDA];

/*****************************************
Gestión de la opción de menú "Uso de la ayuda".
*****************************************/

        virtual void CMUsoAyuda (RTMessage Msg) = [CM_FIRST + CM_USOAYUDA];

/*****************************************
Gestión de la opción de menú "Acerca de TMA...".
*****************************************/

        virtual void CMAcercade (RTMessage Msg) = [CM_FIRST + CM_ACERCADE];
```

```
/************************************
Protección de la salida de la aplicación.
************************************/

        virtual BOOL CanClose ();

protected:

/************************************************************
Definición de ciertas propiedades y recursos de la aplicación".
************************************************************/

        virtual void GetWindowClass (WNDCLASS & WndClass);

/*****************************************************
Gestión de los mensajes de dibujo en la ventana principal".
*****************************************************/

        virtual void Paint (HDC hdc, PAINTSTRUCT &PS);

private:

        BYTE    byBaseAnterior,// Indica la base a la que estaba enganchado el móvil seleccionado.
                byCelulasGrupo,// Indica cuántas células conforman un grupo.
                byGrupoAnterior,// Indica el grupo al que estaba enganchado el móvil seleccionado.
                byMovilSeleccionado;// Indica cuál es el móvil seleccionado.

        BOOL    bCortaComunicacion,// Indica si la llamada telefónica se ha cortado.
                bCursorDentro,// Indica si el cursor se encuentra dentro de la ventana superior izquierda.
                bIniciaLlamada,// Indica si se acaba de iniciar la llamada.
                bLlamadaEnCurso,// Indica si la llamada está en curso.
                bLlamadaPerdida,// Indica si se ha perdido la llamada.
                bLlamadaRecibida,// Indica si se ha recibido una llamada.
                bMovilesActivados,// Indica si los móviles están visualizados en pantalla.
                bYaEnganchado;//Si el móvil seleccionado se encontraba dentro del área de cobertura.

        char    sCanal[5],// Contiene el el canal utilizado por el móvil seleccionado.
                sCelula[3],// Contiene la célula en la que se encuentra el móvil seleccionado.
                sControl[11],// Forma en la que se controla el movimiento del móvil seleccionado.
                sGrupo[2],// Contiene el grupo en el que se encuentra el móvil seleccionado.
                sLlamada[11],// Contiene el estado de la llamada telefónica del móvil seleccionado.
                sMovil[9];// Contiene el móvil seleccionado.

        DWORD           dwColor[NumMoviles][xBmp][yBmp];
                        // Almacena el color de los pixels que van a ser redibujados por los móviles.

        float   fAreaCelula,// Valor del área ocupada por cada célula.
                fBloqueo,// Valor de la probabilidad de bloqueo de una llamada.
                fMaximaRSI//Valor máximo de la relación Señal/Interferencia.
                fMinimaRSI,// Valor mínimo de la relación Señal/Interferencia.
                fTraficoCelula;//Valor del tráfico ofrecido por cada célula.
```

```
        POINT  *pOrigenCelulas,// Apuntará a un vector de las coordenas de los centros de las células.
               ptActual[NumMoviles],// Almacena las coordenadas actuales de los móviles.
               ptAnterior[NumMoviles],// Almacena las coordenadas anteriores de los móviles.
               ptFinal[NumMoviles],// Almacena las coordenadas hacia donde se desplazarán los móviles
               ptVentana;// Almacena las coordenadas de la ventana principal.

        WORD   wCanalesCelula,// Indica el número de canales que contiene cada célula.
               wGruposZona,// Indica el número de grupos que conforman el sistema.
               wInformacion,// Indica las magnitudes a ver en la ventana derecha.
               wRadio,// Indica el radio de las células.
               wTiempo;// Indica el tiempo en milisegundos que tardan en moverse los móviles.

/*************************************************************
Crea las coordenadas a ocupar por las células de un grupo dado.
        ptOrigen -> coordenadas del origen del grupo.
        byGrupo -> identificador del grupo.
*************************************************************/

        void OrigenCelulas (POINT ptOrigen, BYTE byGrupo);

/*************************************************************
Crea las coordenadas a ocupar por los grupos del sistema celular.
*************************************************************/

        void OrigenGrupos ();

/*************************************************************
Dibuja la estación de base de una célula.
        ptOrigen -> coordenadas de la estación de base
        bActiva -> indica si hay que dibujarla de color blanco (TRUE),
                o de color negro (FALSE).
*************************************************************/
        void DibujaBase (POINT ptOrigen, BOOL bActiva);

/************************************************
Dibuja una célula.
        ptOrigen -> coordenadas del centro de la célula.
        dwColorBorde -> color de los bordes de la célula.
        dwColorFondo -> color de fondo de la célula.
************************************************/

        void DibujaCelula (POINT ptOrigen, DWORD dwColorBorde, DWORD dwColorFondo);

/*************************************************************
Dibuja un grupo de células.
        byGrupo -> identificador del grupo a dibujar.
        dwColorBorde -> color de los bordes de las células del grupo.
*************************************************************/

        void DibujaGrupo (BYTE byGrupo, DWORD dwColorBorde);
```

```
/*************************************************************************
Crea de forma aleatoria las coordenadas del móvil identificado por ptMovil.
*************************************************************************/

        void CreaPuntoAleatorio (POINT &ptMovil);

/*****************************************************
Dibuja un móvil.
        ptMovil -> coordenadas sobre las que dibujar el móvil.
        byMovil -> identificador del móvil.
*****************************************************/

        void DibujaMovil (POINT ptMovil, BYTE byMovil);

/*****************************************************
Borra un móvil.
        ptMovil -> coordenadas sobre las que dibujar el móvil.
        byMovil -> identificador del móvil.
*****************************************************/

        void BorraMovil (POINT ptMovil, BYTE byMovil);

/*************************************************************************
Envía cuadros de mensaje sobre el móvil seleccionado a la ventana principal.
        byMensaje -> identificador del mensaje a enviar.
*************************************************************************/

        void MensajeMovil (BYTE byMensaje);

/*************************************************************
Visualiza los datos referentes al móvil seleccionado.
        bEnganchado -> el móvil puede estar dentro del área de cobertura
                                (TRUE) o fuera (FALSE).
        bMismaBase -> el móvil ya se encontraba en la misma célula
                                (TRUE) o no (FALSE).
        byBase -> identificador de la célula en la que se encuentra.
        byGrupo -> identificador del grupo en el que se encuentra.
*************************************************************/
        void CaracteristicasMovil (BOOL bEnganchado, BOOL bMismaBase, BYTE byBase, BYTE byGru-
po);

/*************************************************************
Localiza al móvil cuyas coordenadas se encuentran en ptMovil
  y llama a las funciones necesarias para indicarselo al usuario.
*************************************************************/

        void LocalizaMovil (POINT ptMovil);
```

```
/*****************************************************************
Calcula si la posición en la que va a dibujarse un móvil está ocupada.
        Devuelve TRUE en caso afirmativo y FALSE en caso contrario.
        ptMovil -> coordenadas sobre las que dibujar el móvil.
        byMovil -> identificador del móvil.
*****************************************************************/

        BOOL PosicionOcupada (POINT ptMovil, BYTE byMovil);

/*********************************************
 Desplaza un móvil.
        ptActual -> posición actuál del móvil.
        ptFinal -> posición a la que debe desplazarse.
*********************************************/

        void DesplazaMovil (POINT &ptActual, POINT ptFinal);

/*****************************************************************
Calcula la posición donde debe desplarse un móvil para acercarse
a la posición final deseada.
*****************************************************************/

        void ActivaMoviles ();

/*****************************************************************
 Detiene a todos los móviles en la posición en la que se encuentran.
*****************************************************************/

        void PausaMoviles ();

/*****************************************************************
 Permite que se desplacen los móviles si antes estaban detenidos.
*****************************************************************/

        void ContinuaMoviles ();

/*********************************************************
Probabilidad de bloqueo de una llamada, fórmula B de Erlang.
        Devuelve la probabilidad en tantos por cien.
        fTrafico -> Tráfico ofrecido
        wCanales -> Número de radiocanales.
*********************************************************/

        float Erlang_B (float fTrafico, WORD wCanales);

/*************************************************
Calcula el canal utilizado por el móvil seleccionado.
        byCelulaActual -> Célula en la que se encuentra.
*************************************************/

        WORD CanalUtilizado (BYTE byCelulaActual);
};
```

```
/*********************************************************
 Definición de las funciones creadas para esta aplicación.
*********************************************************/

void CompruebaDato (PTEdit p, WORD wMensaje)
{
        if (wMensaje == EN_CHANGE))
        {
        // el contenido del cuadro de edición ha cambiado.
                BYTE by, byLongitud = p->GetLineLength(0);
                BOOL bPuntoDecimal = FALSE;
                char sValor[NumChar];
                p->GetText(sValor, byLongitud+1);
                for (by = 0; by < byLongitud; by++)
                {
                // comprueba el contenido carácter a carácter.
                        if ((isalpha(sValor[by])) || (!isalnum(sValor[by])))
                        {
                        // el carácter no es un número entero.
                                if ((!bPuntoDecimal) && (sValor[by] == '.'))
                                        bPuntoDecimal = TRUE;
                                else
                                {
                                // el carácter no es válido.
                                        sValor[by]='0';
                                        p->SetText(sValor);
                                        p->SetSelection(by, by+1);
                                }
                        }
                }
                float fValor = atof(sValor);
                if (fValor < 1)
                {
                // el valor es menor que la unidad.
                        p->SetText("1");
                        p->SetSelection(0, 1);
                }
        }
}

void TApp :: InitMainWindow ()
{
        randomize();
        MainWindow = new TMainWindow(NULL, Name);
        HAccTable = LoadAccelerators(hInstance, "tmaMenu");
}
```

```
MovilDialog :: MovilDialog (PTWindowsObject AParent, LPSTR DName) : TDialog(AParent, DName)
{
/* Crea el buffer de transferencia de datos entre la ventana principal
y el cuadro de diálogo "Seleccionar Móviles...". */

        pMovimiento = new TGroupBox(this, ID_MOVIMIENTO);
        new TRadioButton(this, ID_MOVIMIENTOALEATORIO, pMovimiento);
        new TRadioButton(this, ID_MOVIMIENTORATON, pMovimiento);
        new TCheckBox(this, ID_MSGCANALBLOQUEADO, NULL);
        new TCheckBox(this, ID_MSGSALIDAZONA, NULL);
        pMovilSeleccionado = new TGroupBox(this, ID_MOVILSELECCIONADO);
        new TRadioButton(this, ID_MOVILAMARILLO, pMovilSeleccionado);
        new TRadioButton(this, ID_MOVILAZUL, pMovilSeleccionado);
        new TRadioButton(this, ID_MOVILNEGRO, pMovilSeleccionado);
        new TRadioButton(this, ID_MOVILROJO, pMovilSeleccionado);
        new TRadioButton(this, ID_MOVILVERDE, pMovilSeleccionado);
        pVelocidad = new TGroupBox(this, ID_VELOCIDAD);
        new TRadioButton(this, ID_VELOCIDADMAXIMA, pVelocidad);
        new TRadioButton(this, ID_VELOCIDADMEDIA, pVelocidad);
        new TRadioButton(this, ID_VELOCIDADMINIMA, pVelocidad);
        new TRadioButton(this, ID_RECIBIRLLAMADAS, NULL);
        TransferBuffer = (void far*)&(((TMainWindow *)Parent)->MovilBuffer);
}

GeomeDialog :: GeomeDialog (PTWindowsObject AParent, LPSTR DName) : TDialog(AParent, DName)
{
/* Crea el buffer de transferencia de datos entre la ventana principal
y el cuadro de diálogo "Magnitudes Geométricas...". */

        pRadioCelula = new TEdit(this, ID_RADIOCELULA, NumChar);
        pAreaTotal = new TEdit(this, ID_AREATOTAL, NumChar);
        pNumCelulas = new TGroupBox(this, ID_GRUPO);
        new TRadioButton(this, ID_GRUPO3CELULAS, pNumCelulas);
        new TRadioButton(this, ID_GRUPO4CELULAS, pNumCelulas);
        new TRadioButton(this, ID_GRUPO7CELULAS, pNumCelulas);
        new TRadioButton(this, ID_GRUPO12CELULAS, pNumCelulas);
        TransferBuffer = (void far*)&(((TMainWindow *)Parent)->GeomeBuffer);
}

RadioDialog :: RadioDialog (PTWindowsObject AParent, LPSTR DName) : TDialog(AParent, DName)
{
/* Crea el buffer de transferencia de datos entre la ventana principal
y el cuadro de diálogo "Magnitudes Radioeléctricas...". */
        pAnchoBanda = new TEdit(this, ID_ANCHOBANDA, NumChar);
        pSeparacionCanales = new TEdit(this, ID_SEPARACIONCANALES, NumChar);
        pRelacionProteccion = new TEdit(this, ID_RELACIONPROTECCION, NumChar);
        pLeyPropagacion = new TEdit(this, ID_LEYPROPAGACION, NumChar);
        pAntena = new TGroupBox(this, ID_ANTENAS);
        new TRadioButton(this, ID_ANTENAOMNIDIRECCIONAL, pAntena);
        new TRadioButton(this, ID_ANTENASECTORIAL120, pAntena);
        new TRadioButton(this, ID_ANTENASECTORIAL60, pAntena);
        TransferBuffer = (void far*)&(((TMainWindow *)Parent)->RadioBuffer);
}
```

```
TraficoDialog :: TraficoDialog (PTWindowsObject AParent, LPSTR DName) : TDialog(AParent, DName)
{
/* Crea el buffer de transferencia de datos entre la ventana principal
y el cuadro de diálogo "Magnitudes de Tráfico...". */

        pDensidadMoviles = new TEdit(this, ID_DENSIDADMOVILES, NumChar);
        TransferBuffer = (void far*)&(((TMainWindow *)Parent)->TraficoBuffer);
}

void GeomeDialog :: IDRadioCelula (RTMessage Msg)
{
        CompruebaDato(pRadioCelula, Msg.LP.Hi);
}

void GeomeDialog :: IDAreaTotal (RTMessage Msg)
{
        CompruebaDato(pAreaTotal, Msg.LP.Hi);
}

void RadioDialog :: IDAnchoBanda (RTMessage Msg)
{
        CompruebaDato(pAnchoBanda, Msg.LP.Hi);
}

void RadioDialog :: IDSeparacionCanales (RTMessage Msg)
{
        CompruebaDato(pSeparacionCanales, Msg.LP.Hi);
}

void RadioDialog :: IDRelacionProteccion (RTMessage Msg)
{
        CompruebaDato(pRelacionProteccion, Msg.LP.Hi);
}

void RadioDialog :: IDLeyPropagacion (RTMessage Msg)
{
        CompruebaDato(pLeyPropagacion, Msg.LP.Hi);
}

void TraficoDialog :: IDDensidadMoviles (RTMessage Msg)
{
        CompruebaDato(pDensidadMoviles, Msg.LP.Hi);
}
```

```
TMainWindow :: TMainWindow (PTWindowsObject AParent, LPSTR ATitle) : TWindow (AParent, ATitle)
{
/* Inicializa la ventana principal, sus variables y el contenido de
los buffers de transferencia de datos. */

        AssignMenu("tmaMenu");

        bCortaComunicacion = FALSE;
        bCursorDentro = FALSE;
        bIniciaLlamada = FALSE;
        bLlamadaEnCurso = FALSE;
        bLlamadaPerdida = FALSE;
        bLlamadaRecibida = FALSE;
        bMovilesActivados = FALSE;
        bYaEnganchado = FALSE;

        byCelulasGrupo = 4;
        byMovilSeleccionado = 2;

        fAreaCelula = 260;
        fBloqueo = 0;
        fMaximaRSI = 13.80;
        fMinimaRSI = 11.35;
        fTraficoCelula = 78;

        pOrigenCelulas = new POINT[byCelulasGrupo*7];
        if (pOrigenCelulas == NULL)
        {
                MessageBox(HWindow, "No existe suficiente memoria para ejecutar "
                                                "esta aplicación. Cierre otras aplicaciones "
                                                "y vuelva a ejecutar esta.",
                        "Mensaje de error", MB_OK | MB_ICONQUESTION);
                exit(1);
        }

        strcpy(sCanal, "0");
        strcpy(sCelula, "0");
        strcpy(sControl, "Automático");
        strcpy(sGrupo, "0");
        strcpy(sLlamada, "Finalizada");
        strcpy(sMovil, "Negro");

        wCanalesCelula = 250;
        wGruposZona = 10;
        wInformacion = CM_DIMENSIONES;
        wTiempo = 1000;
```

```
// Primero se pone el buffer a cero y luego se introducen los datos.
        memset(&MovilBuffer, 0x0, sizeof (MovilBuffer));
        MovilBuffer.wMovimientoAleatorio = BF_CHECKED;
        MovilBuffer.wMovilNegro = BF_CHECKED;
        MovilBuffer.wMsgCanalBloqueado = BF_CHECKED;
        MovilBuffer.wMsgSalidaZona = BF_CHECKED;
        MovilBuffer.wVelocidadMedia = BF_CHECKED;
        MovilBuffer.wRecibirLlamadas = BF_CHECKED;

        memset(&GeomeBuffer, 0x0, sizeof (GeomeBuffer));
        strcpy(GeomeBuffer.sRadioCelula, "10");
        strcpy(GeomeBuffer.sAreaTotal, "10000");
        GeomeBuffer.wGrupo4Celulas = BF_CHECKED;
        memset(&RadioBuffer, 0x0, sizeof (RadioBuffer));
        strcpy(RadioBuffer.sAnchoBanda, "50");
        strcpy(RadioBuffer.sSeparacionCanales, "25");
        strcpy(RadioBuffer.sRelacionProteccion, "13");
        strcpy(RadioBuffer.sLeyPropagacion, "4");
        RadioBuffer.wAntenaSectorial120 = BF_CHECKED;

        memset(&TraficoBuffer, 0x0, sizeof (TraficoBuffer));
        strcpy(TraficoBuffer.sDensidadMoviles, "10");
}

void TMainWindow :: WMInitMenuPopup (RTMessage Msg)
{
        PausaMoviles();
}

void TMainWindow :: WMMenuSelect (RTMessage Msg)
{
        if (Msg.LP.Lo == 65535)
        // Se ha cerrado el menú desplegable.
                ContinuaMoviles();
}

void TMainWindow :: WMCreate (RTMessage Msg)
{
        ShowWindow(HWindow, SW_MAXIMIZE);
}

void TMainWindow :: WMSize (RTMessage Msg)
{
        ptVentana = MAKEPOINT(Msg.LParam);
        OrigenGrupos();
}
```

```
void TMainWindow :: WMMouseMove (RTMessage Msg)
{
        if (bMovilesActivados)
        {
                if ((Msg.LP.Lo >= ptVentana.x*0.6-10)||(Msg.LP.Hi >= ptVentana.y*0.8-10)
                        ||(Msg.LP.Lo < 10)||(Msg.LP.Hi < 10))
                        // el cursor se encuentra fuera de la ventana superior izquierda.
                        bCursorDentro = FALSE;
                else
                        // el cursor se encuentra dentro de la ventana superior izquierda.
                        bCursorDentro = TRUE;
        }
}

void TMainWindow :: WMLButtonDown (RTMessage Msg)
{
        if (bMovilesActivados)
        {
                if (bCursorDentro)
                {
                        if (MovilBuffer.wMovimientoRaton == BF_CHECKED)
                        // móvil controlado por ratón.
                                ptFinal[byMovilSeleccionado] = MAKEPOINT(Msg.LParam);
                        else
                        // móvil controlado por la aplicación.
                                MensajeMovil(0);
                }
                else
                        MensajeMovil(1);
        }
}

void TMainWindow :: WMRButtonDown (RTMessage Msg)
{
        if (wInformacion == CM_MOVILES)
        // la ventana derecha contiene "Cálculos sobre los móviles".
                SendMessage(HWindow, WM_COMMAND, CM_DIMENSIONES, 0);
        else
                SendMessage(HWindow, WM_COMMAND, wInformacion+1, 0);
}

void TMainWindow :: WMTimer (RTMessage Msg)
{
        ActivaMoviles();
}
```

```
void TMainWindow :: CMVisualizarMoviles (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        EnableMenuItem(hmenu, CM_VISUALIZARMOVILES, MF_GRAYED);
        EnableMenuItem(hmenu, CM_BORRARMOVILES, MF_ENABLED);
        EnableMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_ENABLED);
        EnableMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_GRAYED);
        CheckMenuItem(hmenu, CM_BORRARMOVILES, MF_UNCHECKED);
        // Crea las posiciones iniciales y finales de los móviles.
        for (BYTE by = 0; by < NumMoviles; by++)
        {
                CreaPuntoAleatorio(ptActual[by]);
                CreaPuntoAleatorio(ptFinal[by]);
        }
        // Localiza el móvil seleccionado, los dibuja a todos y actica el timer.
        LocalizaMovil(ptActual[byMovilSeleccionado]);
        for (by = 0; by < NumMoviles; by++)
                DibujaMovil(ptActual[by], by);
        bMovilesActivados = TRUE;
        SetTimer(HWindow, TIMER_MOVIL, wTiempo, NULL);
}

void TMainWindow :: CMBorrarMoviles (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        EnableMenuItem(hmenu, CM_VISUALIZARMOVILES, MF_ENABLED);
        EnableMenuItem(hmenu, CM_BORRARMOVILES, MF_GRAYED);
        EnableMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_GRAYED);
        EnableMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_GRAYED);
        EnableMenuItem(hmenu, CM_SELECCIONARMOVIL, MF_ENABLED);
        CheckMenuItem(hmenu, CM_VISUALIZARMOVILES, MF_UNCHECKED);
        CheckMenuItem(hmenu, CM_BORRARMOVILES, MF_CHECKED);
        CheckMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_UNCHECKED);
        CheckMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_CHECKED);
        // Actualiza el cursor y las ventanas de la izquierda.
        SetClassWord(HWindow, GCW_HCURSOR, LoadCursor(NULL, IDC_ARROW));
        for (BYTE by = 0; by < NumMoviles; by++)
                BorraMovil(ptActual[by], by);
        DibujaGrupo(byGrupoAnterior, RGB(0,0,0));
        bYaEnganchado = FALSE;
        bMovilesActivados = FALSE;
        KillTimer(HWindow, TIMER_MOVIL);
        bLlamadaEnCurso = FALSE;
        bLlamadaPerdida = FALSE;
        bCortaComunicacion = TRUE;
        strcpy(sLlamada, "Finalizada");
        strcpy(sCelula, "0");
        strcpy(sGrupo, "0");
        strcpy(sCanal, "0");
        RECT rect = {10, ptVentana.y*0.8+5, ptVentana.x*0.6-10, ptVentana.y-10};
        InvalidateRect(HWindow, &rect, TRUE);
}
```

```
void TMainWindow :: CMEfectuarLlamada (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        EnableMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_GRAYED);
        EnableMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_ENABLED);
        EnableMenuItem(hmenu, CM_SELECCIONARMOVIL, MF_GRAYED);
        CheckMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_CHECKED);
        CheckMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_UNCHECKED);
        bLlamadaEnCurso = TRUE;
        bIniciaLlamada = TRUE;
}

void TMainWindow :: CMCortarComunicacion (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        EnableMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_ENABLED);
        EnableMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_GRAYED);
        EnableMenuItem(hmenu, CM_SELECCIONARMOVIL, MF_ENABLED);
        CheckMenuItem(hmenu, CM_EFECTUARLLAMADA, MF_UNCHECKED);
        CheckMenuItem(hmenu, CM_CORTARCOMUNICACION, MF_CHECKED);
        bLlamadaEnCurso = FALSE;
        bLlamadaPerdida = FALSE;
        bCortaComunicacion = TRUE;
}

void TMainWindow :: CMSeleccionarMovil (RTMessage Msg)
{
        PausaMoviles();
        GetModule()->ExecDialog(new MovilDialog(this, "movilDlg"));
        // Identifica al móvil seleccionado.
        if (MovilBuffer.wMovilAmarillo == BF_CHECKED)
        {
                byMovilSeleccionado = 0;
                strcpy(sMovil, "Amarillo");
        }
        else
        {
                if (MovilBuffer.wMovilAzul == BF_CHECKED)
                {
                        byMovilSeleccionado = 1;
                        strcpy(sMovil, "Azul");
                }
                else
                {
                        if (MovilBuffer.wMovilNegro == BF_CHECKED)
                        {
                                byMovilSeleccionado = 2;
                                strcpy(sMovil, "Negro");
                        }
                        else
                        {
```

```
                        if (MovilBuffer.wMovilRojo == BF_CHECKED)
                        {
                                byMovilSeleccionado = 3;
                                strcpy(sMovil, "Rojo");
                        }
                        else
                        {
                                byMovilSeleccionado = 4;
                                strcpy(sMovil, "Verde");
                        }
                }
        }
    }
    // Determina la velocidad de los móviles.
    if (MovilBuffer.wVelocidadMaxima == BF_CHECKED)
        wTiempo = 500;
    else
    {
        if (MovilBuffer.wVelocidadMedia == BF_CHECKED)
            wTiempo = 1000;
        else
            wTiempo = 2000;
    }
    // Determina el movimiento del móvil seleccionado.
    if ((MovilBuffer.wMovimientoRaton == BF_CHECKED))
        strcpy(sControl, "Ratón");
    else
        strcpy(sControl, "Automático");
    // Visualiza las características del móvil seleccionado.
    BYTE byEspacioFilas = (ptVentana.y*0.2-20)/3;
    BYTE byEspacioColum = (ptVentana.x*0.6-30)/4;
    HDC hdc = GetDC(HWindow);
    HBRUSH hbrush = SelectObject(hdc, GetStockObject(GRAY_BRUSH));
    HPEN hpen = SelectObject(hdc, CreatePen(PS_SOLID, 1, RGB(128,128,128)));
    Rectangle(hdc, 20+byEspacioColum, ptVentana.y*0.8+10,19+byEspacioColum*2, ptVenta-
na.y*0.8+10+byEspacioFilas);
    Rectangle(hdc, 20+byEspacioColum, ptVentana.y*0.8+10+byEspacioFilas, 19+byEspacioColum*2,
ptVentana.y*0.8+10+byEspacioFilas*2);
    DeleteObject(SelectObject(hdc, hbrush));
    DeleteObject(SelectObject(hdc, hpen));
    DWORD dwMovil[NumMoviles] =
    { RGB(255,255,0), RGB(0,0,255), RGB(0,0,0), RGB(191,0,0), RGB(0,255,0) };
    SetTextColor(hdc, dwMovil[byMovilSeleccionado]);
    SetBkMode(hdc, TRANSPARENT);
    TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10, sMovil, strlen(sMovil));
    TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10+byEspacioFilas, sControl, strlen(sControl));
    TextOut(hdc,20+byEspacioColum, ptVentana.y*0.8+10+byEspacioFilas*2, sLlamada,
strlen(sLlamada));
    TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10, sCelula, strlen(sCelula));
    TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10+byEspacioFilas, sGrupo,strlen(sGrupo));
    TextOut(hdc, 20+byEspacioColum*3.5,ptVentana.y*0.8+10+byEspacioFilas*2,sCanal,strlen(sCanal));
    ReleaseDC(HWindow, hdc);
    ContinuaMoviles();
}
```

```
void TMainWindow :: CMDimensiones (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        CheckMenuItem(hmenu, wInformacion, MF_UNCHECKED);
        wInformacion = Msg.WParam;
        CheckMenuItem(hmenu, wInformacion, MF_CHECKED);
        // Visualiza la ventana derecha.
        HDC hdc = GetDC(HWindow);
        char sTexto[50];
        char sValor[15];
        HBRUSH hbrush = SelectObject(hdc, CreateSolidBrush(RGB(128,128,0)));
        Rectangle(hdc, ptVentana.x*0.6, 5, ptVentana.x-5, ptVentana.y-5);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, (ptVentana.x*0.6)+5, 10, ptVentana.x-10, ptVentana.y-10);
        DeleteObject(SelectObject(hdc, hbrush));
        // Visualiza los cálculos de dimensiones.
        SetBkMode( hdc, TRANSPARENT);
        BYTE byEspacioFilas = (ptVentana.y-20)/21;
        for (BYTE by = 0; by < 20; by++)
        {
                switch (by)
                {
                case 0:
                        strcpy(sTexto, "    CALCULOS DE DIMENSIONES");
                        SetTextColor(hdc, RGB(128,128,0));
                        break;
                case 1:
                        strcpy(sTexto, "EN CADA CELULA:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 2:
                        strcpy(sTexto, "Area de cobertura:");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 3:
                        strcpy(sTexto, "Sc = ");
                        gcvt(fAreaCelula, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " Km²");
                        SetTextColor(hdc,RGB(0,128,128));
                        break;
                case 4:
                        strcpy(sTexto, "Radio de cobertura:");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 5:
                        strcpy(sTexto, "R = ");
                        strcat(sTexto, GeomeBuffer.sRadioCelula);
                        strcat(sTexto, " Km");
                        SetTextColor(hdc,RGB(0,128,128));
                        break;
```

```
case 6:
        strcpy(sTexto, "Altura de la estación de base:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 7:
        strcpy(sTexto, "hb = ");
        gcvt(pow(atof(GeomeBuffer.sRadioCelula)/4.1, 2), 4, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " m");
        SetTextColor(hdc,RGB(0,128,128));
        break;
case 8:
        strcpy(sTexto, "EN CADA GRUPO:");
        SetTextColor(hdc, RGB(0,0,255));
        break;
case 9:
        strcpy(sTexto, "Area de cobertura:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 10:
        strcpy(sTexto, "Sg = ");
        gcvt(fAreaCelula*byCelulasGrupo, NumChar, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " Km²");
        SetTextColor(hdc,RGB(0,128,128));
        break;
case 11:
        strcpy(sTexto, "Número de células:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 12:
        strcpy(sTexto, "j = ");
        gcvt(byCelulasGrupo, NumChar, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " células");
        SetTextColor(hdc, RGB(0,128,128));
        break;
case 13:
        strcpy(sTexto, "EN EL SISTEMA CELULAR:");
        SetTextColor(hdc, RGB(0,0,255));
        break;
case 14:
        strcpy(sTexto, "Area de cobertura:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 15:
        strcpy(sTexto, "Ss = ");
        gcvt(fAreaCelula*byCelulasGrupo*wGruposZona, NumChar, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " km²");
        SetTextColor(hdc, RGB(0,128,128));
        break;
```

```
                case 16:
                        strcpy(sTexto, "Número de grupos:");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 17:
                        strcpy(sTexto, "Q = ");
                        gcvt(wGruposZona, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " grupos");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 18:
                        strcpy(sTexto, "Número de células:");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 19:
                        strcpy(sTexto, "N = ");
                        gcvt(wGruposZona*byCelulasGrupo, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " células");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                }
                TextOut(hdc, (ptVentana.x*0.6)+15, byEspacioFilas*(by+1), sTexto, strlen(sTexto));
        }
        ReleaseDC(HWindow, hdc);
        ContinuaMoviles();
}

void TMainWindow :: CMRadioelectricos (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        CheckMenuItem(hmenu, wInformacion, MF_UNCHECKED);
        wInformacion = Msg.WParam;
        CheckMenuItem(hmenu, wInformacion, MF_CHECKED);
        // Visualiza la ventana derecha.
        HDC hdc = GetDC(HWindow);
        char sTexto[50];
        char sValor[15];
        float fRCocanal = sqrt(3*byCelulasGrupo);
        HBRUSH hbrush = SelectObject(hdc, CreateSolidBrush(RGB(0,128,0)));
        Rectangle(hdc, ptVentana.x*0.6, 5, ptVentana.x-5, ptVentana.y-5);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, (ptVentana.x*0.6)+5, 10, ptVentana.x-10, ptVentana.y-10);
        DeleteObject(SelectObject(hdc, hbrush));
        // Visualiza los cálculos radioeléctricos.
        SetBkMode( hdc, TRANSPARENT);
        BYTE byEspacioFilas = (ptVentana.y-20)/16;
        for (BYTE by = 0; by < 15; by++)
        {
                switch (by)
                {
```

```
case 0:
        strcpy(sTexto, "   CALCULOS RADIOELECTRICOS");
        SetTextColor(hdc, RGB(0,128,0));
        break;
case 1:
        strcpy(sTexto, "Reutilización Cocanal:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 2:
        strcpy(sTexto, "q = ");
        gcvt(fRCocanal, 4, sValor);
        strcat(sTexto, sValor);
        SetTextColor(hdc, RGB(0,128,128));
        break;
case 3:
        strcpy(sTexto, "Distancia de Reutilización:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 4:
        strcpy(sTexto, "D = ");
        gcvt(atof(GeomeBuffer.sRadioCelula)*fRCocanal, 5, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " km");
        SetTextColor(hdc, RGB(0,128,128));
        break;
case 5:
        strcpy(sTexto, "Relación S / I mínima:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 6:
        strcpy(sTexto, "S / I mín = ");
        gcvt(fMinimaRSI, 5, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " dB");
        SetTextColor(hdc, RGB(0,128,128));
        break;
case 7:
        strcpy(sTexto, "Relación S / I máxima:");
        SetTextColor(hdc, RGB(0,0,0));
        break;
case 8:
        strcpy(sTexto, "S / I máx = ");
        gcvt(fMaximaRSI, 5, sValor);
        strcat(sTexto, sValor);
        strcat(sTexto, " dB");
        SetTextColor(hdc, RGB(0,128,128));
        break;
case 9:
        strcpy(sTexto, "EN CADA CELULA:");
        SetTextColor(hdc, RGB(0,0,255));
        break;
```

```
                case 10:
                        strcpy(sTexto, "Cc = ");
                        gcvt(wCanalesCelula, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " canales");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 11:
                        strcpy(sTexto, "EN CADA GRUPO:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 12:
                        strcpy(sTexto, "Cg = ");
                        gcvt(wCanalesCelula*wGruposZona, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " canales");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 13:
                        strcpy(sTexto, "EN EL SISTEMA CELULAR:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 14:
                        strcpy(sTexto, "Cs = ");
                        gcvt(wCanalesCelula*wGruposZona*byCelulasGrupo, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " canales");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                }
                TextOut(hdc, (ptVentana.x*0.6)+15, byEspacioFilas*(by+1), sTexto, strlen(sTexto));
        }
        ReleaseDC(HWindow, hdc);
        ContinuaMoviles();
}

void TMainWindow :: CMTrafico (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        CheckMenuItem(hmenu, wInformacion, MF_UNCHECKED);
        wInformacion = Msg.WParam;
        CheckMenuItem(hmenu, wInformacion, MF_CHECKED);
        // Visualiza la ventana derecha.
        HDC hdc = GetDC(HWindow);
        char sTexto[50];
        char sValor[15];
        HBRUSH hbrush = SelectObject(hdc, CreateSolidBrush(RGB(255,0,0)));
        Rectangle(hdc, ptVentana.x*0.6, 5, ptVentana.x-5, ptVentana.y-5);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, (ptVentana.x*0.6)+5, 10, ptVentana.x-10, ptVentana.y-10);
        DeleteObject(SelectObject(hdc, hbrush));
        // Visualiza los cálculos de tráfico.
        SetBkMode( hdc, TRANSPARENT);
```

```
BYTE byEspacioFilas = (ptVentana.y-20)/18;
for (BYTE by = 0; by < 17; by++)
{
        switch (by)
        {
        case 0:
                strcpy(sTexto, "      CALCULOS DE TRAFICO");
                SetTextColor(hdc, RGB(255,0,0));
                break;
        case 1:
                strcpy(sTexto, "Densidad de Trafico Ofrecido: ");
                SetTextColor(hdc, RGB(0,0,0));
                break;
        case 2:
                strcpy(sTexto, "DT = ");
                gcvt(fTraficoCelula/fAreaCelula, 4, sValor);
                strcat(sTexto, sValor);
                strcat(sTexto, " E/km²");
                SetTextColor(hdc, RGB(0,128,128));
                break;
        case 3:
                strcpy(sTexto, "Densidad de Trafico Cursado: ");
                SetTextColor(hdc, RGB(0,0,0));
                break;
        case 4:
                strcpy(sTexto, "DT' = ");
                gcvt((fTraficoCelula/fAreaCelula)*(1-fBloqueo), 4, sValor);
                strcat(sTexto, sValor);
                strcat(sTexto, " E/km²");
                SetTextColor(hdc, RGB(0,128,128));
                break;
        case 5:
                strcpy(sTexto, "EN CADA CANAL:");
                SetTextColor(hdc, RGB(0,0,255));
                break;
        case 6:
                strcpy(sTexto, "Trafico Ofrecido, A = ");
                gcvt(fTraficoCelula/wCanalesCelula, 4, sValor);
                strcat(sTexto, sValor);
                strcat(sTexto, " E");
                SetTextColor(hdc, RGB(0,0,0));
                break;
        case 7:
                strcpy(sTexto, "Trafico Cursado, A' = ");
                gcvt((fTraficoCelula/wCanalesCelula)*(1-fBloqueo), 4, sValor);
                strcat(sTexto, sValor);
                strcat(sTexto, " E");
                SetTextColor(hdc, RGB(0,0,0));
                break;
        case 8:
                strcpy(sTexto, "EN CADA CELULA:");
                SetTextColor(hdc, RGB(0,0,255));
                break;
```

```
            case 9:
                    strcpy(sTexto, "Trafico Ofrecido, Ac = ");
                    gcvt(fTraficoCelula, 5, sValor);
                    strcat(sTexto, sValor);
                    strcat(sTexto, " E");
                    SetTextColor(hdc, RGB(0,0,0));
                    break;
            case 10:
                    strcpy(sTexto, "Trafico Cursado, Ac' = ");
                    gcvt((fTraficoCelula)*(1-fBloqueo), 5, sValor);
                    strcat(sTexto, sValor);
                    strcat(sTexto, " E");
                    SetTextColor(hdc, RGB(0,0,0));
                    break;
            case 11:
                    strcpy(sTexto, "EN CADA GRUPO:");
                    SetTextColor(hdc, RGB(0,0,255));
                    break;
            case 12:
                    strcpy(sTexto, "Trafico Ofrecido, Ag = ");
                    gcvt(fTraficoCelula*byCelulasGrupo, 5, sValor);
                    strcat(sTexto, sValor);
                    strcat(sTexto, " E");
                    SetTextColor(hdc, RGB(0,0,0));
                    break;
            case 13:
                    strcpy(sTexto, "Trafico Cursado, Ag' = ");
                    gcvt((fTraficoCelula*byCelulasGrupo)*(1-fBloqueo), 5, sValor);
                    strcat(sTexto, sValor);
                    strcat(sTexto, " E");
                    SetTextColor(hdc, RGB(0,0,0));
                    break;
            case 14:
                    strcpy(sTexto, "EN EL SISTEMA CELULAR:");
                    SetTextColor(hdc, RGB(0,0,255));
                    break;
            case 15:
                    strcpy(sTexto, "Trafico Ofrecido, As = ");
                    gcvt(fTraficoCelula*byCelulasGrupo*wGruposZona, 5, sValor);
                    strcat(sTexto, sValor);
                    strcat(sTexto, " E");
                    SetTextColor(hdc, RGB(0,0,0));
                    break;
            case 16:
                    strcpy(sTexto, "Trafico Cursado, As' = ");
                    gcvt((fTraficoCelula*byCelulasGrupo*wGruposZona)*(1-fBloqueo), 5, sValor);
                    strcat(sTexto, sValor);
                    strcat(sTexto, " E");
                    SetTextColor(hdc, RGB(0,0,0));
                    break;
            }
            TextOut(hdc, (ptVentana.x*0.6)+15, byEspacioFilas*(by+1), sTexto, strlen(sTexto));
    }
    ReleaseDC(HWindow, hdc);
```

```
        ContinuaMoviles();
}

void TMainWindow :: CMMoviles (RTMessage Msg)
{
        // Actualiza las opciones del menú desplegable.
        HMENU hmenu = GetMenu (HWindow);
        CheckMenuItem(hmenu, wInformacion, MF_UNCHECKED);
        wInformacion = Msg.WParam;
        CheckMenuItem(hmenu, wInformacion, MF_CHECKED);
        // Visualiza la ventana deecha.
        HDC hdc = GetDC(HWindow);
        char sTexto[50];
        char sValor[15];
        WORD wMovilesCelula = atof(TraficoBuffer.sDensidadMoviles)*fAreaCelula;
        HBRUSH hbrush = SelectObject(hdc, CreateSolidBrush(RGB(0,0,255)));
        Rectangle(hdc, ptVentana.x*0.6, 5, ptVentana.x-5, ptVentana.y-5);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, (ptVentana.x*0.6)+5, 10, ptVentana.x-10, ptVentana.y-10);
        DeleteObject(SelectObject(hdc, hbrush));
        // Visualiza los cálculos sobre los móviles.
        SetBkMode( hdc, TRANSPARENT);
        BYTE byEspacioFilas = (ptVentana.y-20)/14;
        for (BYTE by = 0; by < 13; by++)
        {
                switch (by)
                {
                case 0:
                        strcpy(sTexto, "  CALCULOS SOBRE LOS MOVILES");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 1:
                        strcpy(sTexto, "Probabilidad de Bloqueo:");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 2:
                        strcpy(sTexto, "p = ");
                        gcvt(100*fBloqueo, 4, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " %");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 3:
                        strcpy(sTexto, "Tiempo medio de llamada:");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 4:
                        strcpy(sTexto, "T = ");
                        gcvt((fTraficoCelula*(1-fBloqueo)*60)/wMovilesCelula, 4, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " minutos");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
```

```
                case 5:
                        strcpy(sTexto, "Usuarios por cada canal:");
                        SetTextColor(hdc, RGB(0,0,0));
                        break;
                case 6:
                        strcpy(sTexto, "M = ");
                        gcvt(wMovilesCelula/wCanalesCelula, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " moviles/canal");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 7:
                        strcpy(sTexto, "EN CADA CELULA:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 8:
                        strcpy(sTexto, "Mc = ");
                        gcvt(wMovilesCelula, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " moviles");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 9:
                        strcpy(sTexto, "EN CADA GRUPO:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 10:
                        strcpy(sTexto, "Mg = ");
                        gcvt(wMovilesCelula*byCelulasGrupo, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " moviles");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                case 11:
                        strcpy(sTexto, "EN EL SISTEMA CELULAR:");
                        SetTextColor(hdc, RGB(0,0,255));
                        break;
                case 12:
                        strcpy(sTexto, "Ms = ");
                        gcvt(wMovilesCelula*byCelulasGrupo*wGruposZona, NumChar, sValor);
                        strcat(sTexto, sValor);
                        strcat(sTexto, " moviles");
                        SetTextColor(hdc, RGB(0,128,128));
                        break;
                }
                TextOut(hdc, (ptVentana.x*0.6)+15, byEspacioFilas*(by+1), sTexto, strlen(sTexto));
        }
        ReleaseDC(HWindow, hdc);
        ContinuaMoviles();
}
```

```
void TMainWindow :: CMMagGeometricas (RTMessage Msg)
{
        PausaMoviles();
        BYTE byCGAnterior = byCelulasGrupo;
        WORD wGZAnterior = wGruposZona;
        GetModule()->ExecDialog(new GeomeDialog(this, "geomeDlg"));
        // Identifica el número de células por grupo.
        if (GeomeBuffer.wGrupo3Celulas == BF_CHECKED)
                byCelulasGrupo = 3;
        else
        {
                if (GeomeBuffer.wGrupo4Celulas == BF_CHECKED)
                        byCelulasGrupo = 4;
                else
                {
                        if (GeomeBuffer.wGrupo7Celulas == BF_CHECKED)
                                byCelulasGrupo = 7;
                        else
                                byCelulasGrupo = 12;
                }
        }
        // Realiza los cálculos necesarios.
        wCanalesCelula =
500*atof(RadioBuffer.sAnchoBanda)/(atof(RadioBuffer.sSeparacionCanales)*byCelulasGrupo);
        fAreaCelula = 2.6*pow(atof(GeomeBuffer.sRadioCelula),2);
        fTraficoCelula = atof(TraficoBuffer.sDensidadMoviles)*TraficoMovil*fAreaCelula;
        fBloqueo = Erlang_B(fTraficoCelula, wCanalesCelula);
        float fGruposZona = atof(GeomeBuffer.sAreaTotal)/(byCelulasGrupo*fAreaCelula);
        wGruposZona = ceil(fGruposZona);
        // Comprueba que los datos no se contradigan.
        if (fGruposZona < 1)
        {
                char sMsg[150] = {"El área ocupada por un grupo es mayor "
                                  "que el área total a cubrir.  SOLUCION:\n"
                                 };
                if (atof(GeomeBuffer.sRadioCelula) > 1)
                        strcat(sMsg, "\nDisminuir el radio de las células.");
                else
                        strcat(sMsg, "\nDisminuir el número de células por grupo.");
                if (MessageBox(HWindow, sMsg, "Mensaje de advertencia", MB_YESNO |
MB_ICONEXCLAMATION) == IDYES)
                        SendMessage(HWindow, WM_COMMAND, CM_MAGGEOMETRICAS, 0);
        }
        float fRCocanal = sqrt(3*byCelulasGrupo);
        if (RadioBuffer.wAntenaOmnidireccional == BF_CHECKED)
        {
                fMinimaRSI = 10*log10(1/((2/pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)))+
                        (2/pow(fRCocanal-1, atof(RadioBuffer.sLeyPropagacion)))+
                        (2/pow(fRCocanal+1, atof(RadioBuffer.sLeyPropagacion)))));
                fMaximaRSI = 10*log10(pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion))/6);
                if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                        if (MessageBox(HWindow, "La relación Señal / Interferencia del sistema "
                                                "es menor que la especificada. SOLUCION:\n"
                                                "\nUtilizar antenas sectoriales de 120º.",
```

```
                                        "Mensaje de advertencia", MB_YESNO |
                        MB_ICONEXCLAMATION) == IDYES)
                                SendMessage(HWindow, WM_COMMAND,
                                                CM_MAGRADIOELECTRICAS, 0);
        }
        else
        {
                if (RadioBuffer.wAntenaSectorial120 == BF_CHECKED)
                {
                        fMinimaRSI = 10*log10(pow(fRCocanal,
                                atof(RadioBuffer.sLeyPropagacion))/2);
                        fMaximaRSI = 10*log10(1/((1/pow(fRCocanal+0.7,
                                atof(RadioBuffer.sLeyPropagacion)))+
                                 (1/pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)))));
                        if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                                if (MessageBox(HWindow, "La relación Señal / Interferencia del
                                                sistema "
                                                "es menor que la especificada.  SOLUCION:\n"
                                                "\nUtilizar antenas sectoriales de 60°.",
                                        "Mensaje de advertencia", MB_YESNO |
                                        MB_ICONEXCLAMATION) == IDYES)
                                        SendMessage(HWindow, WM_COMMAND,
                                                CM_MAGRADIOELECTRICAS, 0);
                }
                else
                {
                        fMinimaRSI = 10*log10(pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)));
                        fMaximaRSI = 10*log10(pow(fRCocanal+0.7,
                                atof(RadioBuffer.sLeyPropagacion)));
                        if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                                if (MessageBox(HWindow, "La relación Señal / Interferencia del
                                                sistema "
                                                "es menor que la especificada.  SOLUCION:\n"
                                                "\nAumentar el número de células por grupo.",
                                                  "Mensaje de advertencia", MB_YESNO |
                                        MB_ICONEXCLAMATION) == IDYES)
                                        SendMessage(HWindow, WM_COMMAND,
                                                CM_MAGGEOMETRICAS, 0);
                }
        }
        // Actualiza la ventana superior izquierda si es necesario.
        if ((byCGAnterior != byCelulasGrupo)||(wGZAnterior != wGruposZona))
        {
                delete pOrigenCelulas;
                BYTE byGrupos = (wGruposZona > 7)? 7: wGruposZona;
                pOrigenCelulas = new POINT[byCelulasGrupo*byGrupos];
                if (pOrigenCelulas == NULL)
                {
                        MessageBox(HWindow, "No existe suficiente memoria para ejecutar "
                                                        "esta aplicación. Cierre otras
                                                        aplicaciones "
                                                        "y vuelva a ejecutar esta.",
                                        "Mensaje de error", MB_OK | MB_ICONQUESTION);
                        exit(1);
```

```
                }
                OrigenGrupos();
                InvalidateRect(HWindow, NULL, TRUE);
        }
        SendMessage(HWindow, WM_COMMAND, wInformacion, 0);
        ContinuaMoviles();
}

void TMainWindow :: CMMagRadioelectricas (RTMessage Msg)
{
        PausaMoviles();
        GetModule()->ExecDialog(new RadioDialog(this, "radioDlg"));
        // Realiza los cálculos necesarios.
        wCanalesCelula =
500*atof(RadioBuffer.sAnchoBanda)/(atof(RadioBuffer.sSeparacionCanales)*byCelulasGrupo);
        fBloqueo = Erlang_B(fTraficoCelula, wCanalesCelula);
        float fRCocanal = sqrt(3*byCelulasGrupo);
        // Comprueba que los datos no se contradigan.
        if (RadioBuffer.wAntenaOmnidireccional == BF_CHECKED)
        {
                fMinimaRSI = 10*log10(1/((2/pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)))+
                                                        (2/pow(fRCocanal-1,
                                atof(RadioBuffer.sLeyPropagacion)))+
                                 (2/pow(fRCocanal+1, atof(RadioBuffer.sLeyPropagacion)))));
                fMaximaRSI = 10*log10(pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion))/6);
                if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                        if (MessageBox(HWindow, "La relación Señal / Interferencia del sistema "
                                                "es menor que la
                                                especificada. SOLUCION:\n"
                                                "\nUtilizar antenas sectoriales de 120º.",
                                                  "Mensaje de advertencia", MB_YESNO |
                                                MB_ICONEXCLAMATION) == IDYES)
                                SendMessage(HWindow, WM_COMMAND,
                                        CM_MAGRADIOELECTRICAS, 0);
        }
        else
        {
                if (RadioBuffer.wAntenaSectorial120 == BF_CHECKED)
                {
                        fMinimaRSI = 10*log10(pow(fRCocanal,
                                atof(RadioBuffer.sLeyPropagacion))/2);
                        fMaximaRSI = 10*log10(1/((1/pow(fRCocanal+0.7,
                                        atof(RadioBuffer.sLeyPropagacion)))+
                                 (1/pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)))));
                        if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                                if (MessageBox(HWindow, "La relación Señal / Interferencia del
                                                sistema "
                                                "es menor que la especificada.  SOLUCION:\n"
                                                "\nUtilizar antenas sectoriales de 60º.",
                                                  "Mensaje de advertencia", MB_YESNO |
                                                MB_ICONEXCLAMATION) == IDYES)
                                        SendMessage(HWindow, WM_COMMAND,
                                                CM_MAGRADIOELECTRICAS, 0);
                }
```

```
                else
                {
                        fMinimaRSI = 10*log10(pow(fRCocanal, atof(RadioBuffer.sLeyPropagacion)));
                        fMaximaRSI = 10*log10(pow(fRCocanal+0.7,
                                atof(RadioBuffer.sLeyPropagacion)));
                        if (atof(RadioBuffer.sRelacionProteccion) > (fMinimaRSI))
                                if (MessageBox(HWindow, "La relación Señal / Interferencia del
                                                sistema "
                                                "es menor que la especificada. SOLUCION:\n"
                                                "\nAumentar el número de células por grupo.",
                                                  "Mensaje de advertencia", MB_YESNO |
                                                MB_ICONEXCLAMATION) == IDYES)
                                        SendMessage(HWindow, WM_COMMAND,
                                                CM_MAGGEOMETRICAS, 0);
                }
        }
        SendMessage(HWindow, WM_COMMAND, wInformacion, 0);
        ContinuaMoviles();
}

void TMainWindow :: CMMagTrafico (RTMessage Msg)
{
        PausaMoviles();
        GetModule()->ExecDialog(new TraficoDialog(this, "traficoDlg"));
        // Realiza los cálculos necesarios.
        fTraficoCelula = atof(TraficoBuffer.sDensidadMoviles)*TraficoMovil*fAreaCelula;
        fBloqueo = Erlang_B(fTraficoCelula, wCanalesCelula);
        SendMessage(HWindow, WM_COMMAND, wInformacion, 0);
        ContinuaMoviles();
}

void TMainWindow :: CMSalir (RTMessage Msg)
{
        SendMessage(HWindow, WM_CLOSE, 0, 0);
}

void TMainWindow :: CMIndice (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_INDEX, 0);
}

void TMainWindow :: CMAyudaAreaCliente (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_KEY, (LONG)(LPSTR) "Area de Cliente");
}
```

```
void TMainWindow :: CMAyudaMenu (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_KEY, (LONG)(LPSTR) "Menú");
}

void TMainWindow :: CMAyudaRaton (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_KEY, (LONG)(LPSTR) "Ratón");
}

void TMainWindow :: CMAyudaMensajes (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_KEY, (LONG)(LPSTR) "Mensajes de información");
}

void TMainWindow :: CMAyudaSistemas (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_KEY, (LONG)(LPSTR) "TMA");
}

void TMainWindow :: CMBuscarAyuda (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_PARTIALKEY, 0);
}

void TMainWindow :: CMUsoAyuda (RTMessage Msg)
{
        // Desactiva los móviles si están activados.
         if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        WinHelp (HWindow, "tma.hlp", HELP_HELPONHELP, 0);
}

void TMainWindow :: CMAcercade (RTMessage Msg)
{
        PausaMoviles();
        GetModule()->ExecDialog(new TDialog(this, "AcercadeDlg"));
        ContinuaMoviles();
}
```

```
BOOL TMainWindow :: CanClose ()
{
        PausaMoviles();
        MessageBeep(0);
        // Pregunta al usuario si desea abandonar la aplicación.
        if (MessageBox(HWindow, "¿Desea abandonar esta aplicación?",
                                "Salida de la aplicación",
                                MB_YESNO | MB_ICONQUESTION) == IDYES)
        {
                delete pOrigenCelulas;
                return TRUE;
        }
        else
        {
                ContinuaMoviles();
                return FALSE;
        }
}

void TMainWindow :: GetWindowClass (WNDCLASS & WndClass)
{
        TWindow :: GetWindowClass(WndClass);
        WndClass.hIcon = LoadIcon(WndClass.hInstance, "tmaIco");
        WndClass.hbrBackground = GetStockObject(WHITE_BRUSH);
}

void TMainWindow :: Paint (HDC hdc, PAINTSTRUCT &PS)
{
        // Desactiva los móviles si están activados.
        if (bMovilesActivados)
                SendMessage(HWindow, WM_COMMAND, CM_BORRARMOVILES, 0);
        // Dibuja la ventana superior izquierda.
        BYTE by, byGrupos = (wGruposZona > 7)? 7: wGruposZona;
        HBRUSH hbrush = SelectObject(hdc, GetStockObject(GRAY_BRUSH));
        Rectangle(hdc, 5, 5, ptVentana.x*0.6-5, ptVentana.y*0.8-5);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, 10, 10, ptVentana.x*0.6-10, ptVentana.y*0.8-10);
        // Dibuja el sistema celular.
        for (by = 0; by < byGrupos; by++)
                DibujaGrupo(by, RGB(0,0,0));
        // Dibuja la ventana inferior izquierda.
        SendMessage(HWindow, WM_COMMAND, wInformacion, 0);
        SelectObject(hdc, GetStockObject(LTGRAY_BRUSH));
        Rectangle(hdc, 5, ptVentana.y*0.8, ptVentana.x*0.6-5, ptVentana.y-5);
        SelectObject(hdc, GetStockObject(GRAY_BRUSH));
        Rectangle(hdc, 10, ptVentana.y*0.8+5, ptVentana.x*0.6-10, ptVentana.y-10);
        DeleteObject(SelectObject(hdc, hbrush));
        // Presenta las características del móvil seleccionado.
        SetBkMode(hdc, TRANSPARENT);
        BYTE byEspacioFilas = (ptVentana.y*0.2-20)/3;
        BYTE byEspacioColum = (ptVentana.x*0.6-30)/4;
        DWORD dwMovil[NumMoviles] =
        { RGB(255,255,0), RGB(0,0,255), RGB(0,0,0), RGB(191,0,0), RGB(0,255,0) };
        SetTextColor(hdc, RGB(255,255,255));
```

```
        TextOut(hdc, 20, ptVentana.y*0.8+10, "Móvil:", 6);
        TextOut(hdc, 20, ptVentana.y*0.8+10+byEspacioFilas, "Control:", 8);
        TextOut(hdc, 20, ptVentana.y*0.8+10+byEspacioFilas*2, "Llamada:", 8);
        TextOut(hdc, 20+byEspacioColum*2, ptVentana.y*0.8+10, "Célula Actual:", 14);
        TextOut(hdc, 20+byEspacioColum*2, ptVentana.y*0.8+10+byEspacioFilas, "Grupo Actual:", 13);
        TextOut(hdc, 20+byEspacioColum*2, ptVentana.y*0.8+10+byEspacioFilas*2, "Canal Actual:", 13);
        SetTextColor(hdc, dwMovil[byMovilSeleccionado]);
        TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10, sMovil, strlen(sMovil));
        TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10+byEspacioFilas, sControl, strlen(sControl));
        TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10+byEspacioFilas*2, sLlamada,
strlen(sLlamada));
        TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10, sCelula, strlen(sCelula));
        TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10+byEspacioFilas, sGrupo,
strlen(sGrupo));
        TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10+byEspacioFilas*2, sCanal,
strlen(sCanal));
}

void TMainWindow :: OrigenCelulas (POINT ptOrigen, BYTE byGrupo)
{
        BYTE by;
        switch (byCelulasGrupo)
        {
        case 3:
                POINT pt3Celulas[] =
                {
                        ptOrigen.x-(wRadio*0.75), ptOrigen.y,
                        ptOrigen.x+(wRadio*0.75), ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x+(wRadio*0.75), ptOrigen.y+(wRadio*0.866),
                };
                for (by = 0; by < byCelulasGrupo; by++)
                        pOrigenCelulas[by+(byGrupo*byCelulasGrupo)] = pt3Celulas[by];
                break;
        case 4:
                POINT pt4Celulas[] =
                {
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y,
                        ptOrigen.x,                          ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x,                          ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y
                };
                for (by = 0; by < byCelulasGrupo; by++)
                        pOrigenCelulas[by+(byGrupo*byCelulasGrupo)] = pt4Celulas[by];
                break;
        case 7:
                POINT pt7Celulas[] =
                {
                        ptOrigen.x,                  ptOrigen.y,
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x,          ptOrigen.y+(wRadio*1.732),
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x,                  ptOrigen.y-(wRadio*1.732),
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y-(wRadio*0.866),
```

```
                };
                for (by = 0; by < byCelulasGrupo; by++)
                        pOrigenCelulas[by+(byGrupo*byCelulasGrupo)] = pt7Celulas[by];
                break;
        case 12:
                POINT pt12Celulas[] =
                {
                        ptOrigen.x-(wRadio*0.25), ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x+(wRadio*1.25), ptOrigen.y,
                        ptOrigen.x-(wRadio*0.25), ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x-(wRadio*1.75), ptOrigen.y,
                        ptOrigen.x-(wRadio*1.75), ptOrigen.y-(wRadio*1.732),
                        ptOrigen.x-(wRadio*0.25), ptOrigen.y-(wRadio*2.598),
                        ptOrigen.x+(wRadio*1.25), ptOrigen.y-(wRadio*1.732),
                        ptOrigen.x+(wRadio*2.75), ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x+(wRadio*2.75), ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x+(wRadio*1.25), ptOrigen.y+(wRadio*1.732),
                        ptOrigen.x-(wRadio*0.25), ptOrigen.y+(wRadio*2.598),
                        ptOrigen.x-(wRadio*1.75), ptOrigen.y+(wRadio*1.732),
                };
                for (by = 0; by < byCelulasGrupo; by++)
                        pOrigenCelulas[by+(byGrupo*byCelulasGrupo)] = pt12Celulas[by];
                break;
        }
}

void TMainWindow :: OrigenGrupos ()
{
        POINT ptOrigen;
        ptOrigen.x = (ptVentana.x*0.6-20)/2+10;
        ptOrigen.y = (ptVentana.y*0.8-20)/2+10;
        WORD wMenor = (ptOrigen.x < ptOrigen.y)? ptOrigen.x: ptOrigen.y;
        BYTE by, byGrupos = (wGruposZona > 7)? 7: wGruposZona;
        switch (byCelulasGrupo)
        {
        case 3:
                wRadio = wMenor/5;
                POINT pt7Grupos3Celulas[] =
                {
                        ptOrigen.x,                             ptOrigen.y,
                        ptOrigen.x+(wRadio*3),   ptOrigen.y,
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y+(wRadio*2.598),
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y+(wRadio*2.598),
                        ptOrigen.x-(wRadio*3),   ptOrigen.y,
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y-(wRadio*2.598),
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y-(wRadio*2.598)
                };
                for (by = 0; by < byGrupos; by++)
                        OrigenCelulas(pt7Grupos3Celulas[by], by);
                break;
```

```
        case 4:
                wRadio = wMenor/6;
                POINT pt7Grupos4Celulas[] =
                {
                        ptOrigen.x,                             ptOrigen.y,
                        ptOrigen.x+(wRadio*3), ptOrigen.y+(wRadio*1.732),
                        ptOrigen.x,                     ptOrigen.y+(wRadio*3.464),
                        ptOrigen.x-(wRadio*3), ptOrigen.y+(wRadio*1.732),
                        ptOrigen.x-(wRadio*3), ptOrigen.y-(wRadio*1.732),
                        ptOrigen.x,         ptOrigen.y-(wRadio*3.464),
                        ptOrigen.x+(wRadio*3), ptOrigen.y-(wRadio*1.732)
                };
                for (by = 0; by < byGrupos; by++)
                        OrigenCelulas(pt7Grupos4Celulas[by], by);
                break;
        case 7:
                wRadio = wMenor/7.5;
                POINT pt7Grupos7Celulas[] =
                {
                        ptOrigen.x,         ptOrigen.y,
                        ptOrigen.x+(wRadio*4.5), ptOrigen.y+(wRadio*0.866),
                        ptOrigen.x+(wRadio*1.5), ptOrigen.y+(wRadio*4.33),
                        ptOrigen.x-(wRadio*3),   ptOrigen.y+(wRadio*3.464),
                        ptOrigen.x-(wRadio*4.5), ptOrigen.y-(wRadio*0.866),
                        ptOrigen.x-(wRadio*1.5), ptOrigen.y-(wRadio*4.33),
                        ptOrigen.x+(wRadio*3),   ptOrigen.y-(wRadio*3.464)
                };
                for (by = 0; by < byGrupos; by++)
                        OrigenCelulas(pt7Grupos7Celulas[by], by);
                break;
        case 12:
                wRadio = wMenor/10;
                POINT pt7Grupos12Celulas[] =
                {
                        ptOrigen.x,                             ptOrigen.y,
                        ptOrigen.x+(wRadio*6), ptOrigen.y,
                        ptOrigen.x+(wRadio*3), ptOrigen.y+(wRadio*5.196),
                        ptOrigen.x-(wRadio*3), ptOrigen.y+(wRadio*5.196),
                        ptOrigen.x-(wRadio*6), ptOrigen.y,
                        ptOrigen.x-(wRadio*3), ptOrigen.y-(wRadio*5.196),
                        ptOrigen.x+(wRadio*3), ptOrigen.y-(wRadio*5.196)
                };
                for (by = 0; by < byGrupos; by++)
                        OrigenCelulas(pt7Grupos12Celulas[by], by);
                break;
        }
}
```

```
void TMainWindow :: DibujaBase (POINT ptOrigen, BOOL bActiva)
{
        HDC hdc = GetDC(HWindow);
        BYTE byRadioBase = ceil(wRadio/10);
        if (bActiva)
        {
                if (byRadioBase < 2)
                        SetPixel(hdc, ptOrigen.x, ptOrigen.y, RGB(255, 255, 255));
        else
                        Ellipse(hdc, ptOrigen.x-byRadioBase, ptOrigen.y-byRadioBase,
                                        ptOrigen.x+byRadioBase, ptOrigen.y+byRadioBase);
        }
        else
        {
                if (byRadioBase < 2)
                        SetPixel(hdc, ptOrigen.x, ptOrigen.y, RGB(0, 0, 0));
                else
                {
                        HBRUSH hbrush = SelectObject(hdc, GetStockObject(BLACK_BRUSH));
                        Ellipse(hdc, ptOrigen.x-byRadioBase, ptOrigen.y-byRadioBase,
                                ptOrigen.x+byRadioBase, ptOrigen.y+byRadioBase);
                        DeleteObject(SelectObject(hdc, hbrush));
                }
        }
        ReleaseDC(HWindow, hdc);
}

void TMainWindow :: DibujaCelula (POINT ptOrigen, DWORD dwColorBorde,
                                DWORD dwColorFondo)
{
        POINT  ptCelula[] =
        {
                ptOrigen.x+wRadio,    ptOrigen.y,
                ptOrigen.x+(wRadio/2), ptOrigen.y+(wRadio*0.866),
                ptOrigen.x-(wRadio/2), ptOrigen.y+(wRadio*0.866),
                ptOrigen.x-wRadio,    ptOrigen.y,
                ptOrigen.x-(wRadio/2), ptOrigen.y-(wRadio*0.866),
                ptOrigen.x+(wRadio/2), ptOrigen.y-(wRadio*0.866),
        };
        HDC hdc = GetDC(HWindow);
        HPEN hpen = SelectObject(hdc, CreatePen(PS_SOLID, 2, dwColorBorde));
        HBRUSH hbrush = SelectObject(hdc, CreateSolidBrush(dwColorFondo));
        Polygon(hdc, ptCelula, 6);
        DeleteObject(SelectObject(hdc, hpen));
        DeleteObject(SelectObject(hdc, hbrush));
        ReleaseDC (HWindow, hdc);
        DibujaBase(ptOrigen, FALSE);
}
```

```
void TMainWindow :: DibujaGrupo (BYTE byGrupo, DWORD dwColorBorde)
{
        DWORD dwColorFondo[] =
        {
                RGB(255,255,0), RGB(0,255,128), RGB(0,255,255), RGB(0,128,128),
                RGB(0,0,255), RGB(255,0,128), RGB(128,0,128), RGB(128,128,0),
                RGB(255,0,0), RGB(128,128,128), RGB(255,0,255), RGB(128,255,0)
        };
        BYTE by, byColor = 0, byPrimeraCelula = byGrupo*byCelulasGrupo,
                byUltimaCelula = byPrimeraCelula+byCelulasGrupo;
        for (by = byPrimeraCelula; by < byUltimaCelula; by++)
        {
                DibujaCelula(pOrigenCelulas[by], dwColorBorde, dwColorFondo[byColor]);
                byColor++;
        }
}

void TMainWindow :: CreaPuntoAleatorio (POINT &ptMovil)
{
        ptMovil.x = 10+random(ptVentana.x*0.6-20);
        ptMovil.y = 10+random(ptVentana.y*0.8-20);
}

void TMainWindow :: DibujaMovil (POINT ptMovil, BYTE byMovil)
{
        HDC hdc = GetDC(HWindow), hdcMem = CreateCompatibleDC(hdc);
        for (WORD x = 0; x < xBmp; x++)
                for (WORD y = 0; y < yBmp; y++)
                        dwColor[byMovil][x][y] = GetPixel(hdc, ptMovil.x+x, ptMovil.y+y);
        HBITMAP hbitmap;
        switch (byMovil)
        {
        case 0:
                hbitmap = LoadBitmap(GetModule()->hInstance,"amarilloBmp");
                break;
        case 1:
                hbitmap = LoadBitmap(GetModule()->hInstance,"azulBmp");
                break;
        case 2:
                hbitmap = LoadBitmap(GetModule()->hInstance,"negroBmp");
                break;
        case 3:
                hbitmap = LoadBitmap(GetModule()->hInstance,"rojoBmp");
                break;
        case 4:
                hbitmap = LoadBitmap(GetModule()->hInstance,"verdeBmp");
                break;
        }
        SelectObject(hdcMem, hbitmap);
        BitBlt(hdc, ptMovil.x, ptMovil.y, xBmp, yBmp, hdcMem, 0, 0, SRCCOPY);
        DeleteDC(hdcMem);
        DeleteObject(hbitmap);
        ReleaseDC(HWindow, hdc);
}
```

```
void TMainWindow :: BorraMovil (POINT ptMovil, BYTE byMovil)
{
        HDC hdc = GetDC(HWindow);
        for (WORD x = 0; x < xBmp; x++)
                for (WORD y = 0; y < yBmp; y++)
                        SetPixel(hdc, ptMovil.x+x, ptMovil.y+y, dwColor[byMovil][x][y]);
        ReleaseDC(HWindow, hdc);
}

void TMainWindow :: DesplazaMovil (POINT &ptActual, POINT ptFinal)
{
        POINT ptDistancia;
        ptDistancia.x = (ptActual.x > ptFinal.x)?
                                        ptActual.x-ptFinal.x: ptFinal.x-ptActual.x;
        if (ptDistancia.x <= yBmp-1)
                ptActual.x = ptFinal.x;
        else
        {
                if (ptActual.x > ptFinal.x)
                        ptActual.x -= yBmp;
                else
                        if (ptActual.x < ptFinal.x)
                                ptActual.x += yBmp;
        }
        ptDistancia.y = (ptActual.y > ptFinal.y)?
                                        ptActual.y-ptFinal.y: ptFinal.y-ptActual.y;
        if (ptDistancia.y <= yBmp-1)
                ptActual.y = ptFinal.y;
        else
        {
                if (ptActual.y > ptFinal.y)
                        ptActual.y -= yBmp;
                else
                        if (ptActual.y < ptFinal.y)
                                ptActual.y += yBmp;
        }
}

void TMainWindow :: MensajeMovil (BYTE byMensaje)
{
        MessageBeep(0);
        PausaMoviles();
        char sMovil[5][9] =
        {"amarillo", "azul", "negro", "rojo", "verde"};
        char sMsg[200];
        if (byMensaje == 4)
        {
                strcpy(sMsg, "El móvil ");
                strcat(sMsg, sMovil[byMovilSeleccionado]);
                strcat(sMsg, " ha recibido una llamada telefónica.\n"
                                        "¿Desea atender la llamada?");
                if (MessageBox(HWindow, sMsg, "Aviso de llamada", MB_YESNO |
MB_ICONINFORMATION) == IDYES)
                {
```

```
                        SendMessage(HWindow, WM_COMMAND, CM_EFECTUARLLAMADA, 0);
                        bLlamadaRecibida = TRUE;
                }
        }
        else
        {
                switch (byMensaje)
                {
                case 0:
                        strcpy(sMsg, "Para poder desplazar el móvil ");
                        strcat(sMsg, sMovil[byMovilSeleccionado]);
                        strcat(sMsg, " a la posición actual del cursor, tiene que estar habilitada la opción"
                        " <<Movimiento por ratón>> del cuadro de diálogo <<Seleccionar Móvil...>>");
                        break;
                case 1:
                        strcpy(sMsg, "No se puede desplazar el móvil ");
                        strcat(sMsg, sMovil[byMovilSeleccionado]);
                        strcat(sMsg, " fuera del rectángulo que recubre el sistema celular.");
                        break;
                case 2:
                        strcpy(sMsg, "El móvil ");
                        strcat(sMsg, sMovil[byMovilSeleccionado]);
                        strcat(sMsg, " ha perdido la comunicación por no encontrase ningún "
                                        "canal disponible en la célula en la que se encuentra.");
                        break;
                case 3:
                        strcpy(sMsg, "La LLamada se ha perdido por encontrase el móvil "
                                        "fuera del área de cobertura del sistema celular.");
                        break;
                }
                MessageBox(HWindow, sMsg, "Localización del móvil", MB_OK |
MB_ICONINFORMATION);
        }
        ContinuaMoviles();
}
void TMainWindow :: CaracteristicasMovil (BOOL bEnganchado, BOOL bMismaBase, BYTE byBase, BYTE
byGrupo)
{
        HDC hdc = GetDC(HWindow);
        BYTE byEspacioFilas = (ptVentana.y*0.2-20)/3;
        BYTE byEspacioColum = (ptVentana.x*0.6-30)/4;
        WORD wCanal;
        HBRUSH hbrush = SelectObject(hdc, GetStockObject(GRAY_BRUSH));
        HPEN hpen = SelectObject(hdc, CreatePen(PS_SOLID, 1, RGB(128,128,128)));
        DWORD dwMovil[NumMoviles] =
        { RGB(255,255,0), RGB(0,0,255), RGB(0,0,0), RGB(191,0,0), RGB(0,255,0) };
        SetTextColor(hdc, dwMovil[byMovilSeleccionado]);
        SetBkMode(hdc, TRANSPARENT);
        if (bCortaComunicacion)
        {
                bCortaComunicacion = FALSE;
                SetClassWord(HWindow, GCW_HCURSOR, LoadCursor(NULL, IDC_ARROW));
                Rectangle(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+byEspacioFilas*2+10,19+byEspacioColum*4, ptVentana.y*0.8+10+byEspacioFilas*3);
```

```
                strcpy(sCanal, "0");
                TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10+byEspacioFilas*2, sCanal,
strlen(sCanal));
                        Rectangle(hdc, 20+byEspacioColum, ptVenta-
na.y*0.8+10+byEspacioFilas*2,19+byEspacioColum*2, ptVentana.y*0.8+10+byEspacioFilas*3);
                strcpy(sLlamada, "Finalizada");
                TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10+byEspacioFilas*2, sLlamada,
strlen(sLlamada));
                }
        else
        {
                if (bEnganchado)
                {
                        if (bMismaBase)
                        {
                                if (bLlamadaEnCurso)
                                {
                                        if (bIniciaLlamada)
                                        {
                                                bIniciaLlamada = FALSE;
                                                wCanal = CanalUtilizado(byBase);
                                                if (wCanal == 0)
                                                {
                                                        strcpy(sLlamada, "Perdida");
                                                        bLlamadaPerdida = TRUE;
                                                        SetClassWord(HWindow, GCW_HCURSOR,
LoadCursor(NULL, IDC_CROSS));
                                                }
                                                else
                                                {
                                                        strcpy(sLlamada, "En Curso");
                                                        SetClassWord(HWindow, GCW_HCURSOR,
LoadCursor(GetModule()->hInstance, "tmaCur"));
                                                }
                                                Rectangle(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+byEspacioFilas*2+10,19+byEspacioColum*4, ptVentana.y*0.8+10+byEspacioFilas*3);
                                                itoa(wCanal, sCanal, 10);
                                                TextOut(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+10+byEspacioFilas*2, sCanal, strlen(sCanal));
                                                Rectangle(hdc, 20+byEspacioColum, ptVenta-
na.y*0.8+10+byEspacioFilas*2,19+byEspacioColum*2, ptVentana.y*0.8+10+byEspacioFilas*3);
                                                TextOut(hdc, 20+byEspacioColum, ptVenta-
na.y*0.8+10+byEspacioFilas*2, sLlamada, strlen(sLlamada));
                                        }
                                }
                                else
                                        if (MovilBuffer.wRecibirLlamadas == BF_CHECKED)
                                                if (random(21) == 0)
                                                        MensajeMovil(4);
                        }
                        else
                        {
                                Rectangle(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+10,19+byEspacioColum*4, ptVentana.y*0.8+10+byEspacioFilas);
```

```
                                itoa(byBase-(byGrupo*byCelulasGrupo)+1, sCelula, 10);
                                TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10, sCelula,
strlen(sCelula));
                                Rectangle(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+10+byEspacioFilas,19+byEspacioColum*4, ptVentana.y*0.8+10+byEspacioFilas*2);
                                itoa(byGrupo+1, sGrupo, 10);
                                TextOut(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+10+byEspacioFilas, sGrupo, strlen(sGrupo));
                                if (bLlamadaEnCurso)
                                {
                                        if (!bLlamadaPerdida)
                                        {
                                                wCanal = CanalUtilizado(byBase);
                                                if (wCanal == 0)
                                                {
                                                        strcpy(sLlamada, "Perdida");
                                                        bLlamadaPerdida = TRUE;
                                                        SetClassWord(HWindow, GCW_HCURSOR,
LoadCursor(NULL, IDC_CROSS));
                                                }
                                                else
                                                {
                                                        strcpy(sLlamada, "En Curso");
                                                        SetClassWord(HWindow, GCW_HCURSOR,
LoadCursor(GetModule()->hInstance, "tmaCur"));
                                                }
                                                Rectangle(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+byEspacioFilas*2+10,19+byEspacioColum*4, ptVentana.y*0.8+10+byEspacioFilas*3);
                                                itoa(wCanal, sCanal, 10);
                                                TextOut(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+10+byEspacioFilas*2, sCanal, strlen(sCanal));
                                                Rectangle(hdc, 20+byEspacioColum, ptVenta-
na.y*0.8+10+byEspacioFilas*2,19+byEspacioColum*2, ptVentana.y*0.8+10+byEspacioFilas*3);
                                                TextOut(hdc, 20+byEspacioColum, ptVenta-
na.y*0.8+10+byEspacioFilas*2, sLlamada, strlen(sLlamada));
                                        }
                                }
                                else
                                        if (MovilBuffer.wRecibirLlamadas == BF_CHECKED)
                                                if (random(21) == 0)
                                                        MensajeMovil(4);
                        }
                }
                else
                {
                        if (bYaEnganchado)
                        {
                                bYaEnganchado = FALSE;
                                Rectangle(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+10,19+byEspacioColum*4, ptVentana.y*0.8+10+byEspacioFilas);
                                strcpy(sCelula, "0");
                                TextOut(hdc, 20+byEspacioColum*3.5, ptVentana.y*0.8+10, sCelula,
strlen(sCelula));
```

```
                                    Rectangle(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+10+byEspacioFilas,19+byEspacioColum*4, ptVentana.y*0.8+10+byEspacioFilas*2);
                                    strcpy(sGrupo, "0");
                                    TextOut(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+10+byEspacioFilas, sGrupo, strlen(sGrupo));
                            }
                            if ((bLlamadaEnCurso)&&(!bLlamadaPerdida))
                            {
                                    bLlamadaPerdida = TRUE;
                                    bIniciaLlamada = FALSE;
                                    SetClassWord(HWindow, GCW_HCURSOR, LoadCursor(NULL,
IDC_CROSS));
                                    Rectangle(hdc, 20+byEspacioColum, ptVenta-
na.y*0.8+10+byEspacioFilas*2,19+byEspacioColum*2, ptVentana.y*0.8+10+byEspacioFilas*3);
                                    strcpy(sLlamada, "Perdida");
                                    TextOut(hdc, 20+byEspacioColum, ptVentana.y*0.8+10+byEspacioFilas*2,
sLlamada, strlen(sLlamada));
                                    Rectangle(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+10+byEspacioFilas*2,19+byEspacioColum*4, ptVentana.y*0.8+10+byEspacioFilas*3);
                                    strcpy(sCanal, "0");
                                    TextOut(hdc, 20+byEspacioColum*3.5, ptVenta-
na.y*0.8+10+byEspacioFilas*2, sCanal, strlen(sCanal));
                                    if (MovilBuffer.wMsgSalidaZona == BF_CHECKED)
                                            MensajeMovil(3);
                            }
                    }
            }
            DeleteObject(SelectObject(hdc, hpen));
            DeleteObject(SelectObject(hdc, hbrush));
            ReleaseDC(HWindow, hdc);
}

void TMainWindow :: LocalizaMovil (POINT ptMovil)
{
            WORD wMenorDistancia, wDistancia, wDistanciax, wDistanciay;
            BYTE by, byGrupos, byBaseEnganchado = 0, byGrupoEnganchado;
            BOOL bEnganchado;
            wDistanciax = (pOrigenCelulas[0].x > ptMovil.x)? pOrigenCelulas[0].x-ptMovil.x: ptMovil.x-
pOrigenCelulas[0].x;
            wDistanciay = (pOrigenCelulas[0].y > ptMovil.y)? pOrigenCelulas[0].y-ptMovil.y: ptMovil.y-
pOrigenCelulas[0].y;
            wMenorDistancia = sqrt(pow((wDistanciax),2)+pow((wDistanciay),2));
            byGrupos = (wGruposZona > 7)? 7: wGruposZona;
            for (by = 1; by < byCelulasGrupo*byGrupos; by++)
            {
                    wDistanciax = (pOrigenCelulas[by].x > ptMovil.x)? pOrigenCelulas[by].x-ptMovil.x: ptMo-
vil.x-pOrigenCelulas[by].x;
                    wDistanciay = (pOrigenCelulas[by].y > ptMovil.y)? pOrigenCelulas[by].y-ptMovil.y: ptMo-
vil.y-pOrigenCelulas[by].y;
                    if ((wMenorDistancia) >= (wDistancia = sqrt(pow((wDistanciax),2)+pow((wDistanciay),2))))
                    {
                            byBaseEnganchado = by;
                            wMenorDistancia = wDistancia;
                    }
```

```
        }
        bEnganchado = (wMenorDistancia > wRadio)? FALSE: TRUE;
        if (bEnganchado)
        {
                byGrupoEnganchado = byBaseEnganchado/byCelulasGrupo;
                if (bYaEnganchado)
                {
                        if (byGrupoEnganchado == byGrupoAnterior)
                        {
                                if (byBaseEnganchado != byBaseAnterior)
                                {
                                        DibujaBase(pOrigenCelulas[byBaseAnterior], FALSE);
                                        DibujaBase(pOrigenCelulas[byBaseEnganchado], TRUE);
                                        byBaseAnterior = byBaseEnganchado;
                                        CaracteristicasMovil(bEnganchado, FALSE, byBaseEnganchado,
byGrupoEnganchado);
                                }
                                else
                                        CaracteristicasMovil(bEnganchado, TRUE, byBaseEnganchado,
byGrupoEnganchado);
                        }
                        else
                        {
                                DibujaGrupo(byGrupoAnterior, RGB(0,0,0));
                                DibujaGrupo(byGrupoEnganchado, RGB(255,255,255));
                                DibujaBase(pOrigenCelulas[byBaseEnganchado], TRUE);
                                byGrupoAnterior = byGrupoEnganchado;
                                byBaseAnterior = byBaseEnganchado;
                                CaracteristicasMovil(bEnganchado, FALSE, byBaseEnganchado, byGru-
poEnganchado);
                        }
                }
                else
                {
                        DibujaGrupo(byGrupoEnganchado, RGB(255,255,255));
                        DibujaBase(pOrigenCelulas[byBaseEnganchado], TRUE);
                        byGrupoAnterior = byGrupoEnganchado;
                        byBaseAnterior = byBaseEnganchado;
                        bYaEnganchado = TRUE;
                        CaracteristicasMovil(bEnganchado, FALSE, byBaseEnganchado, byGrupoEngan-
chado);
                }
        }
        else
        {
                CaracteristicasMovil(bEnganchado, FALSE, byBaseEnganchado, byGrupoEnganchado);
                if (bYaEnganchado)
                {
                        DibujaGrupo(byGrupoAnterior, RGB(0,0,0));
                        bYaEnganchado = FALSE;
                }
        }
}
```

```
BOOL TMainWindow :: PosicionOcupada (POINT ptMovil, BYTE byMovil)
{
        RECT rect =
        {
                ptMovil.x-xBmp, ptMovil.y-yBmp,
                ptMovil.x+(2*xBmp), ptMovil.y+(2*yBmp)
        };
        BOOL bDentroRect = FALSE;
        BYTE by=0;
        do
        {
                if (byMovil != by)
                        bDentroRect = PtInRect(&rect, ptActual[by]);
                by++;
        }
        while ((by < NumMoviles) && (!bDentroRect));
                return bDentroRect;
}

void TMainWindow :: ActivaMoviles ()
{
        for (BYTE by = 0; by < NumMoviles; by++)
        {
                ptAnterior[by] = ptActual[by];
                if ((ptActual[by].x == ptFinal[by].x) && (ptActual[by].y == ptFinal[by].y))
                {
                        if (by == byMovilSeleccionado )
                        {
                                if (MovilBuffer.wMovimientoAleatorio == BF_CHECKED)
                                        CreaPuntoAleatorio(ptFinal[by]);
                        }
                        else
                                CreaPuntoAleatorio(ptFinal[by]);
                }
                else
                {
                        DesplazaMovil(ptActual[by], ptFinal[by]);
                        if (PosicionOcupada(ptActual[by], by))
                        {
                                ptActual[by] = ptAnterior[by];
                                if (by == byMovilSeleccionado )
                                {
                                        if (MovilBuffer.wMovimientoAleatorio == BF_CHECKED)
                                                CreaPuntoAleatorio(ptFinal[by]);
                                }
                                else
                                        CreaPuntoAleatorio(ptFinal[by]);
                        }
                }
        }
        for ( by = 0; by < NumMoviles; by++)
                BorraMovil(ptAnterior[by], by);
        LocalizaMovil(ptActual[byMovilSeleccionado]);
        for ( by = 0; by < NumMoviles; by++)
```

```
                DibujaMovil(ptActual[by], by);
}

void TMainWindow :: PausaMoviles ()
{
        if (bMovilesActivados)
                KillTimer(HWindow, TIMER_MOVIL);
}

void TMainWindow :: ContinuaMoviles ()
{
        if (bMovilesActivados)
                SetTimer(HWindow, TIMER_MOVIL, wTiempo, NULL);
}
float TMainWindow :: Erlang_B (float fTrafico, WORD wCanales)
{
        WORD w1,w2;
        long double ldSumando, ldSumatoria = 0;
        for (w1 = 0; w1 <= wCanales; w1++)
        {
                ldSumando = 1;
                for (w2 = w1; w2 > 1; w2--)
                        ldSumando *= fTrafico / w2;
                ldSumatoria += ldSumando;
        }
        return ldSumando / ldSumatoria;
}

WORD TMainWindow :: CanalUtilizado (BYTE byCelulaActual)
{
        if (bLlamadaRecibida)
                bLlamadaRecibida = FALSE;
        else
        {
                BYTE byBloqueo = random(101);
                if (byBloqueo < 100*fBloqueo)
                {
                        if (MovilBuffer.wMsgCanalBloqueado == BF_CHECKED)
                                MensajeMovil(2);
                                return 0;
                }
        }
        return random(wCanalesCelula-byCelulasGrupo+1)+byCelulaActual;
}

// Punto de entrada de la aplicación.
int PASCAL WinMain (HINSTANCE hInstance, HINSTANCE hPrevInstance,
                LPSTR lpCmdLine, int nCmdShow)
{
        TApp App ("Planificación de un Sistema de Telefonía Móvil Celular",
                        hInstance, hPrevInstance, lpCmdLine, nCmdShow);
        App.Run();
        return App.Status;
}
```

ANEXO 2

EL SISTEMA DE AYUDA TMA.HLP

En el anexo 2 vamos a explicar los pasos necesarios que hay que seguir para poder desarrollar un sistema de ayuda para el entorno de Microsoft Windows.[1],[8]

Una vez comentados todos los archivos que son necesarios para crear un sistema de ayuda, se presentarán los archivos que forman la ayuda TMA.HLP.

A2.1 La ayuda de Windows

La mayoría de las aplicaciones Windows vienen provistas de un sistema de ayuda con información especializada. La ventana en la que se muestran estos textos de ayuda la pone a disposición el sistema de Windows y tiene por nombre WINHELP.EXE. Esto requiere una sofisticada capacidad de ayuda que incluye hiperenlaces, ayuda sensible al contexto, índices, capacidad de hojear y gráficos integrados en caso necesario.

Borland ha simplificado el proceso de generación de esta sofisticada ayuda incluyendo junto al compilador de Borland C++ para Windows un compilador de archivos de ayuda. Dicho compilador necesita archivos de entrada con formato de texto enriquecido (RTF), un formato soportado por los buenos procesadores de texto.

A2.1.1 Proyectar una función de ayuda

A continuación veremos cuáles son los pasos a seguir para proyectar una función de ayuda:

- Recolectar informaciones y planear la estructura del sistema de ayuda.

- Colocar textos y códigos de control en el archivo RTF.

- Editar el archivo de proyecto de ayuda (HPJ).

- Crear el archivo con el texto de la ayuda (HLP) mediante el compilador de Borland HC(Help Compiler).

- Comprobar el sistema de ayuda activándolo en nuestra aplicación.

A2.1.2 Archivos de texto enriquecido (RTF)

Para crear nuestro sistema de ayuda, debemos editar o copiar el texto insertándolo en el formato .RTF, y por otro lado deben incluirse los códigos de control que decidirán el modo en el que el usuario podrá desplazarse dentro del sistema de ayuda.

Para poder crear archivos de texto enriquecido se requiere un editor de texto capaz de guardar archivos en formato RTF, que sea capaz de trabajar con notas a pie de página y con el que se pueda tachar y subrayar palabras.

A2.1.3 Códigos de control

A continuación mostraremos cuales son los códigos de control que reconoce el compilador de ayuda en nuestros ficheros RTF. Dichos códigos se engloban en dos grupos, y serán comentados con detalle en los siguientes apartados de este anexo.

Caracteres de pie de página:

#	->	Context_string.
$	->	Título del tema.
*****	->	Nombre simbólico.
k	->	Palabra clave.
+	->	Número de la secuencia de búsqueda.

Identificadores de texto:

Subrayado o doble		
subrayado	->	Salto a otro tema.
Subrayado	->	Existe definición adicional.
Oculto	->	Referencia cruzada.

A2.1.3.1 Código de control para un context_string

Cada tema del sistema de ayuda exige para su identificación un context_string propio e individual con un máximo de 255 caracteres, sin hacer distinción entre minúsculas y mayúsculas.

Para asignar un context_string a un tema de ayuda, debe marcarse el principio del tema con el carácter "#" como marca de referencia al pie de página. A continuación se introduce el context_string, pudiendo haber un espacio en blanco entre ellos, como muestra la figura A2-1.

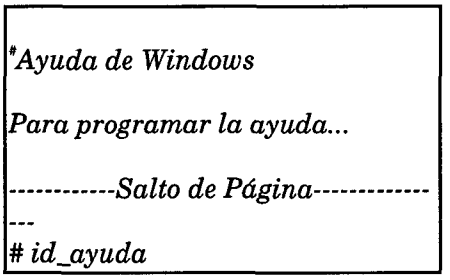


Figura A2-1

A2.1.3.2 Código de control para un título

La mayoría de los temas de ayuda están provistos de un título válido internamente para el sistema de ayuda. Este título se ofrece al usuario, por ejemplo, como cadena de caracteres por defecto cuando se trata de definir una marca en la función de ayuda a través de la opción *Marca Texto*. Además aparece en la lista inferior del cuadro de diálogo buscar, una vez que se ha hallado un término clave en uno o varios temas.

Para definir un título, debe introducirse al principio del tema en cuestión el carácter "$" como marca de referencia al pie de página, y a continuación se escribe el título, que no puede formatearse, pudiendo constar de hasta 127 caracteres. En la figura A2-2 podemos ver un ejemplo:

```
#$Ayuda de Windows

Para programar la ayuda...

----------Salto de Página----------
---
# id_ayuda
$ Ayuda de Windows
```

Figura A2-2

A2.1.3.3 Código de control para un término clave

Los términos claves (keyword) pueden compararse con el índice de un libro. Pueden introducirse tantos como se desee. Aparecerán en un cuadro de lista cuando se active el cuadro de diálogo del botón buscar. No se distingue entre minúsculas y mayúsculas.

La marca de referencia de nota al pie de página para los términos clave es la letra "K", minúscula o mayúscula. Tras ella deben introducirse como texto de nota al pie, todos los términos clave, separados los unos de los otros mediante un guión (;). El texto puede constar de varias líneas.

En la figura A2-3 vemos como a nuestro archivo RTF le hemos añadido dos términos clave:

```
#$kAyuda de Windows

Para programar la ayuda...

----------Salto de Página----------
---
# id_ayuda
$ Ayuda de Windows
k Ayuda; Windows
```

Figura A2-3

A2.1.3.4 Código de control para un número de secuencia de búsqueda

El orden en el que se presentan los temas se denomina "Browse Sequence", que se establece mediante números de secuencia que deben ser fijados por el autor del sistema de ayuda.

El carácter "+" es la marca de referencia de nota al pie para las secuencias de búsqueda y como texto de nota al pie posee un número de secuencia de búsqueda. Este número de secuencia consta normalmente de un nombre de lista de secuencia y un número separados por el carácter ":". Todos los números de secuencia que tienen el mismo nombre de la lista de secuencia, pertenecen a la misma secuencia de búsqueda. Los números han de tener la misma cantidad de dígitos.

En la figura A2-4 hemos añadido un número de secuencia de búsqueda a nuestra página de ayuda:

```
#$K+Ayuda de Windows

Para programar la ayuda...

----------Salto de Página----------
---
# id_ayuda
$ Ayuda de Windows
k Ayuda; Windows
+ ayuda:05
```

Figura A2-4

A2.1.3.5 Código de control para una referencia cruzada

Las referencias cruzadas existentes hacen posible pasar del nivel más alto de la jerarquía al siguiente nivel hasta alcanzar el nivel inferior. Para hacer posibles estos saltos, se necesitan los context_strings de los diversos temas.

En primer lugar debe determinarse el pasaje de texto desde el que se efectuará el salto. A este pasaje debe asignársele el formato de texto tachado o doble subrayado. A continuación se introduce con el texto oculto, el context_string del tema al que se pasará. El texto oculto debe añadirse al texto tachado o doble subrayado, como se muestra en la figura A2-5.

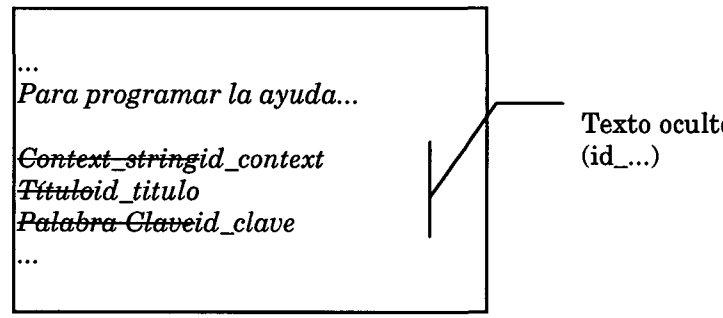


Figura A2-5

A2.1.3.6 Código de control para definiciones de conceptos

Las expresiones que ofrecen esta opción están subrayadas por una línea discontinua. Cuando el usuario hace un clic con el ratón sobre una palabra señalizada de este modo, aparece una ventana de detalle con informaciones adicionales.

Las explicaciones adicionales son desde el punto de vista del compilador de ayuda, temas independientes, que requieren un context_string.

La expresión que se desee explicar con más exactitud, debe tener el formato de carácter subrayado. A continuación debe escribirse en formato de texto oculto el context_string del tema que suministra la explicación.

Un ejemplo de esto lo podemos ver en la figura A2-6:

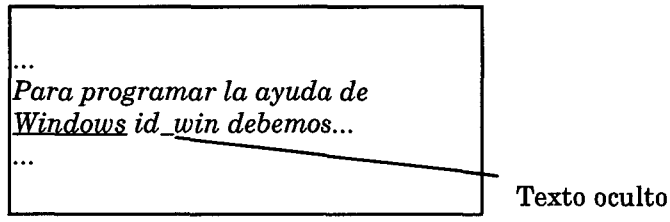


Figura A2-6

A2.1.3.7 Código de control para la inserción de bitmaps

Los bitmaps los podemos insertar de dos maneras. Si el editor permite la inserción directa de imágenes, basta con extraer la imagen del portapapeles e insertarla en la posición deseada dentro del texto. Al guardar el archivo, el bitmap se convertirá automáticamente al formato de archivo RTF y quedará incluido en el archivo de ayuda. El segundo método consiste en incluir una referencia a un archivo .BMP. Al hacerlo podemos escoger entre insertar la imagen a la izquierda o a la derecha del texto o bien en medio de éste.

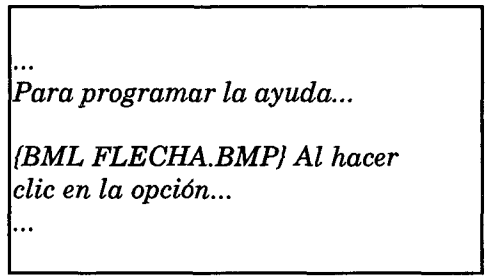


Figura A2-7

A2.1.4 Crear el archivo de ayuda

El archivo de proyecto puede escribirse con cualquier editor ASCII y debe constar de un máximo de seis apartados. La extensión de este archivo será .HPJ.

Antes de compilar un archivo de ayuda hay que borrar el archivo con extensión .PH que se crea en cada compilación.

En nuestra aplicación hemos utilizado los siguientes apartados:

[FILES] Contiene los archivos temáticos; obligatorio.

[OPTIONS] Contiene diferentes opciones.

[BITMAPS] Indica los bitmaps a enlazar.

A2.1.4.1 [FILES]

En él se indican todos los archivos de temas que formaran el archivo de ayuda binario. En la figura A2-8 podemos ver un ejemplo:

```
[FILES]
indice.rtf
menu.rtf
ventana.rtf
```

Figura A2-8

A2.1.4.2 [OPTIONS]

Puede contener hasta ocho posibilidades de elección distintas, de los cuales explicaremos los tres utilizados en este proyecto:

- **FORCEFONT**, sirve para forzar al texto a un sólo tipo de letra independientemente del tipo de letra utilizado en los archivos RTF.

- **COMPRESS**, se utiliza para reducir el tamaño del archivo de ayuda al máximo.

- **TITLE**, Si se desea asignar al tema de ayuda un título que aparecerá en la barra de títulos de la función de ayuda, utilizamos esta opción. No puede contener más de 32 caracteres.

En la figura A2-9 tenemos un ejemplo de como utilizar estas opciones:

```
...
[OPTIONS]
FORCEFONT=Roman
COMPRESS=TRUE
TITLE=Sistema de Ayuda:
```

Figura A2-9

A2.1.4.3 [BITMAPS]

Si el sistema de ayuda accede a bitmaps que no han sido importados, sino ligados al texto mediante una referencia, deben listarse en este apartado los nombres de archivo y sus vías de acceso si es necesario, como podemos ver en la figura A2-10.

```
...
[BITMAPS]
flecha.bmp
coche.bmp
```

Figura A2-10

A2.1.5 El compilador de ayuda

El compilador de ayuda puede activarse desde en entorno MS-DOS añadiéndole como parámetro el nombre del archivo de proyecto con la vía de acceso si es necesario:

```
HC AYUDA.HPJ
```

El archivo de ayuda surgido de la compilación llevará el mismo nombre que el archivo de proyecto y se creará en el directorio actual con extensión .HLP. Este archivo podrá ser cargado y leído con la aplicación de ayuda WINHELP.EXE.

A2.1.6 Programación de la ayuda

Para poder activar el sistema de ayuda desde una aplicación , debe recurrirse a la función WinHelp(). Si desea conocer la declaración y los parámetros de esta función podrá encontrarla en el archivo de cabecera WINDOWS.H y también en el fichero de ayuda del compilador de Borland C++.

A2.2 Los ficheros de TMA.HLP

A continuación veremos el contenido de todos los ficheros que han sido utilizados para la creación del sistema de ayuda de nuestra aplicación. Siendo dichos archivos los que se listan a continuación:

- **TMA.HPJ**
- **INDICE.RTF**
- **CLIENTE.RTF**
- **MENU.RTF**
- **RATON.RTF**
- **MENSAJES.RTF**
- **SISTEMAS.RTF**

A2.2.1 TMA.HPJ

En este archivo declaramos todas las opciones necesarias para compilar los archivos RTF de nuestro sistema de ayuda de Windows:

```
[OPTIONS]
FORCEFONT=Roman
TITLE=AYUDA DE TMA
COMPRESS=TRUE

[FILES]
indice.rtf
cliente.rtf
menu.rtf
raton.rtf
mensajes.rtf
sistemas.rtf

[BITMAPS]
cocanal.bmp
```

A2.2.2 INDICE.RTF

En este archivo definimos el índice del sistema de ayuda:

#$+**Indice**

->Esta aplicación posee un sistema de ayuda acerca del contenido de los menús de la aplicación y de cómo afectan al área de cliente de la ventana principal. Además incluye información sobre algunas características de los sistemas móviles celulares.

Ayuda sobre la aplicación:

~~Área de cliente~~id_areacliente
~~Menú principal~~id_menu
~~Ratón~~id_raton
~~Mensajes de información~~id_mensajes

Introducción a los sistemas celulares:

~~Sistemas Celulares~~id_sistemas

---

id_indice
$ Indice
+ Indice:05

A2.2.3 CLIENTE.RTF

En este archivo encontraremos toda la ayuda existente sobre el área de cliente de la aplicación:

#$k+**Descripción del contenido del Área de Cliente**

->El **área de cliente** de esta aplicación contiene tres partes diferenciadas y enmarcadas por rectángulos a modo de ventanas hijas:

~~La ventana superior izquierda~~id_ventanasi
~~La ventana inferior izquierda~~id_ventanaii
~~La ventana derecha~~id_ventanad

----------Salto de página----------

#$k+**Area de Cliente: 1.- Ventana superior izquierda**

->En esta ventana encontraremos un dibujo bidimensional de la configuración de un sistema celular. Se podrá visualizar un máximo de 7 grupos y un mínimo de uno. Cada grupo podrá contener 3, 4, 7 ó 12 células.
->Además existe la posibilidad de ver 5 móviles en dicha ventana, los cuales se irán desplazando dentro de ella de forma aleatoria. Uno de esos cinco móviles estará seleccionado y según se vaya desplazando activará la estación de base y el grupo en los que se encuentre localizado.

Localizado -> Color Blanco
No localizado -> Color Negro
----------Salto de página----------

```
# id_areacliente
$ Area de Cliente
k Area de Cliente
+ Indice:10
# id_ventanasi
$ Ventana superior izquierda
k Móviles
+ AreaCliente:05
```

#$k+**Area de Cliente: 2.- Ventana inferior izquierda**

->Esta ventana posee información acerca del móvil seleccionado para hacer un seguimiento de su localización y del estado de las llamadas que realice o que reciba.
->El contenido de la ventana es el siguiente:

Móvil: Indica cuál es el móvil seleccionado.

Velocidad: La velocidad de los móviles puede variarse pasando por tres estados: *Mínima*, *media* o *máxima*.

Llamada: Las llamadas pueden pasar por tres estados diferentes:
**En Curso*: Cuando se mantiene la conversación.
**Perdida*: Cuando se pierde la llamada.
**Finalizada*: Cuando se "cuelga" el teléfono.

Célula: Indica el número de la célula en la que se encuentra localizado el móvil seleccionado. El número será cero si se encuentra fuera del área de cobertura.

Grupo: Indica el número del grupo en el que se encuentra localizado el móvil seleccionado. El número será cero si se encuentra fuera del área de cobertura.

Canal: Indica el canal utilizado por el móvil seleccionado.
----------Salto de página----------

#$k+**Area de Cliente: 3.- Ventana derecha**

->Esta es la ventana de información, cuyo contenido lo podemos cambiar pulsando una de las opciones del menú *Información* o pulsando el botón derecho del ratón siempre que este se encuentre dentro del ~~área de cliente~~id_areacliente de nuestra aplicación.
->Las distintas ventanas de información son las siguientes:

~~Dimensiones~~id_vd_dimensiones
~~Radioeléctricos~~id_vd_radioelectricos
~~Tráfico~~id_vd_trafico
~~Características de los Móviles~~id_vd_moviles
----------Salto de página----------

```
# id_ventanaii
$ Ventana inferior izquierda
k Móvil; Velocidad; Llamada; Célula; Grupo; Canal
+ AreaCliente:10
# id_ventanad
$ Ventana derecha
k Información; Dimensiones; Radioeléctricos; Tráfico; Móvil
+ AreaCliente:15
```

#$K+ **Ventana Derecha: Dimensiones**

->En esta ventana se presentan todos los cálculos realizados sobre las **dimensiones** de los distintos elementos que componen nuestro sistema celular
->Estos elementos son:

~~Área de Cobertura~~id_d_areacobertura
~~Radio de cobertura de cada célula~~id_d_radio
~~Altura de antena de estación de base~~id_d_antena
~~Número de células y de grupos~~id_d_numero
----------Salto de página----------

#$K+ **Dimensiones: Área de Cobertura**

->El **área de cobertura** es la superficie en la que se puede atender a los móviles y comunicarlos entre sí o con los teléfonos fijos.

En cada célula:

->Como las células son hexagonales, su **área de cobertura** será igual a la superficie de un hexágono regular.

Scid_Sc = 2.6 * Radio²id_Radio km²

En cada grupo:

->Se define como el área de una célula multiplicado por el número de células que contiene el grupo.

Sgid_Sg = Sc * jid_j km²

En el sistema:

->Se define como el área de una célula multiplicado por el número de células que contiene el sistema y también como el área de un grupo multiplicado por el número de grupos que contiene el sistema.

Ssid_Ss = Sc * Nid_N = Sg * Qid_Q km²
----------Salto de página----------

```
# id_vd_dimensiones
$ Ventana derecha: Dimensiones
K Dimensiones; Area de Cobertura; Radio; Antena; Célula;
  Grupo; Sistemas Celulares
+ ACVentanaDerecha:05
# id_d_areacobertura
$ Dimensiones: Area de Cobertura
K Area de Cobertura; Célula; Grupo; Sistemas Celulares
+ VDDimensiones:05
```

#$k+**Dimensiones: Radio de Cobertura**

->En nuestra aplicación hemos supuesto que el **radio de cobertura** de todas las células es el mismo, y puede ser modificado por el usuario.
->El **radio de cobertura** es la distancia máxima de la que puede alejarse un móvil respecto a una estación de base, sin que se pierda la señal.
----------Salto de página----------

#$k+**Dimensiones: Altura de Antena de Estación de Base**

->Existe una expresión que relaciona el ~~radio de cobertura~~id_d_radio de una célula con la **altura de la antena de la estación de base** de dicha célula y la **altura de la antena de los teléfonos móviles**:

Radioid_radio = 4.1 * (√hbid_hb + √hmid_hm) km

Podemos despreciar la **altura de la antena de los teléfonos móviles** con respecto a la **antena de la estación de base** y así obtenemos:

hb = (Radio / 4.1)² metros
----------Salto de página----------

#$k+**Dimensiones: Número de Células y Grupos**

->Si nos referimos al **número de células** contenidas en un grupo, lo identificamos con la letra "j". En cambio, si nos referimos al **número de células** contenidas en el sistema, lo haremos con la letra "N".
->La relación entre ellas viene dada por:

Nid_N = jid_j * Qid_Q células

->Donde Q es el número de grupos que conforman el sistema.

```
# id_d_radio
$ Dimensiones: Radio de Cobertura
k Radio
+ VDDimensiones:10
# id_d_antena
$ Dimensiones: Altura de Antena de Estación de Base
k Antena; Radio
+ VDDimensiones:15
# id_d_numero
$ Dimensiones: Número de Células y Grupos
k Célula; Grupo
+ VDDimensiones:20
```

----------Salto de página----------

#$k+**Ventana Derecha: Radioeléctricos**

->En esta ventana se muestran todos los cálculos **radioeléctricos** de nuestro sistema celular.
->Su contenido es el siguiente:

~~Reutilización Cocanal~~id_r_reutilizacion
~~Distancia de Reutilización~~id_r_distancia
~~Relación Señal / Interferencia~~id_r_senal
~~Número de canales~~id_r_canales
----------Salto de página----------

#$k+**Radioeléctricos: Reutilización Cocanal**

->La relación de reutilización cocanal, viene definida en la siguiente expresión:
qid_q = Did_D / Radioid_Radio = √(3 * jid_j)

->A medida que aumenta el número de células por grupo, también aumenta la relación de **reutilización cocanal** y se mejora la relación ~~Señal / Interferencia~~id_r_senal por lo que se mejora la calidad de las llamadas telefónicas.
----------Salto de página----------

#$k+**Radioeléctricos: Distancia de Reutilización**

->Es la distancia mínima en la que pueden situarse dos estaciones de base que comparten el mismo juego de radiocanales:

Did_D = Radioid_Radio * √(3 * jid_j) Km
----------Salto de página----------

```
# id_vd_radioelectricos
$ Ventana Derecha: Radioeléctricos
k Radioeléctricos; Reutilización; Distancia de Reutilización;
  Señal/Interferencia; Canal
+ ACVentanaDerecha:10
# id_r_reutilizacion
$ Radioeléctricos: Reutilización
k Reutilización
+ VDRadioelectricos:05
# id_r_distancia
$ Radioeléctricos: Distancia de Reutilización
k Distancia de Reutilización
+ VDRadioelectricos:10
```

#$K+ **Radioeléctricos: Relación Señal / Interferencia**

->La interferencia producida por el uso común de un mismo canal, se denomina **interferencia cocanal**, y es uno de los principales problemas de los sistemas celulares. A medida que reducimos el número de células dentro de un grupo para poder reutilizar el mayor número posible de radiocanales, disminuye la distancia de reutilización y consecuentemente aumenta la interferencia cocanal.
->La relación **Señal / Interferencia** depende también del tipo de **antenas** utilizadas y del lugar donde se encuentre. Mientras más se aleje el móvil de la estación de base, mayor será la interferencia cocanal y por lo tanto peor será la relación **Señal / Interferencia**.

----------Salto de página----------

#$K+ **Radioeléctricos: Número de Canales**

->El **número de canales** disponibles dentro de una célula, viene dado por la siguiente expresión:

Ccid_Cc = Cgid_Cg / jid_j canales

->Estando el **número de canales** disponibles en cada grupo, en función del número máximo que nos permite el ancho de banda ofrecido para la transmisión de telefonía móvil celular.

Cg = Bid_B / (2 * δfid_δf) canales

->El **número de canales** disponibles en el sistema celular será:

Csid_Cs = Cc * Nid_N = Cg * Qid_Q canales

----------Salto de página----------

#$K+ **Ventana Derecha: Tráfico**

->En esta ventana podemos ver todos los cálculos realizados sobre el **tráfico** que podemos obtener a partir de la densidad de móviles por kilómetro cuadrado que requiera el usuario.
->Los elementos de esta ventana son:

~~Densidad de tráfico~~id_t_densidad

```
# id_r_senal
$ Radioeléctricos: Señal/Interferencia
k Señal/interferencia; Interferencia
+ VDRadioelectricos:15
# id_r_canales
$ Radioeléctricos: Número de Canales
k Canal
+ VDRadioelectricos:20
# id_vd_trafico
$ VentanaDerecha: Tráfico
k Tráfico; Densidad de Tráfico
+ ACVentanaDerecha:15
```

~~Tráfico~~id_t_trafico

----------Salto de página----------

#$k+ **Tráfico: Densidad de Tráfico**

->La **densidad de tráfico** en cada célula, en cada grupo, y en el sistema, es la siguiente:

Ofrecido:

DTid_DT = Acid_Ac / Scid_Sc_km²

Cursado:

DT'id_DT' = Ac'id_Ac' / Sc Erlangs / km²

----------Salto de página----------

#$k+ **Tráfico:Tráfico**

->Las magnitudes de tráfico ofrecido y cursado en cada célula, en cada canal, en cada grupo, y en el sistema se indican a continuación:

Acid_Ac = DMid_DM * eid_e * Scid_Sc Erlangs

Ac'id_Ac' = Ac * (1 - pid_p) Erlangs

Aid_A = Ac / Ccid_Cc Erlangs

A'id_A' = Ac' / Cc Erlangs

Agid_Ag = Ac * jid_j Erlangs

Ag'id_Ag' = Ac' * j Erlangs
Asid_As = Ac * Nid_N Erlangs

As'id_As' = Ac' * N Erlangs

----------Salto de página----------

---

id_t_densidad
$ Ventana Derecha: Densidad de Tráfico
k Densidad de Tráfico
+ VDTrafico:05
# id_t_trafico
$ Tráfico: Tráfico
k Tráfico
+ VDTrafico:10

#$k+**Ventana Derecha: Características de los Móviles**

->En esta ventana obtenemos los datos referentes a los **móviles** de nuestro sistema celular.
->Los elementos de esta ventana son:

~~Probabilidad de bloqueo~~id_mo_bloqueo
~~Tiempo de Llamada~~id_mo_tiempo
~~Usuarios por Canal~~id_mo_usuarios
~~Móviles~~id_mo_moviles
----------Salto de página----------

#$k+**Móviles: Probabilidad de bloqueo**

->La fórmula B de Erlang nos da el valor de la **probabilidad de bloqueo** de una llamada en función del **tráfico ofrecido** y del **número de canales**.
->Se define como la probabilidad de que al efectuar una llamada, esta se bloquee, y por lo tanto se pierda la comunicación al estar en ese momento todos los canales ocupados.
->Su expresión es la siguiente:

pid_p = $B(c,a) = (a^c / c!) / (\sum_{i=0}^{c}(a^i / y!))$ % Llamadas perdidas
----------Salto de página----------

#$k+**Móviles: Tiempo de Llamada**

->La siguiente expresión calcula el tiempo medio de duración de las llamadas en nuestro sistema celular:

Tid_T = 60 * Ac'id_Ac' / Mcid_Mc minutos
----------Salto de página----------

```
# id_vd_moviles
$ Ventana Derecha: Móviles
k Móvil; Probabilidad de Bloqueo; Tiempo de Llamada; Usuarios por Canal
+ ACVentanaDerecha:20
# id_mo_bloqueo
$ Móviles: Probabilidad de Bloqueo
k Probabilidad de Bloqueo; Erlang
+ VDMoviles:05
# id_mo_tiempo
$ Móviles: Tiempo de Llamada
k Tiempo de Llamada
+ VDMoviles:10
```

#$K+**Móviles: Usuarios por Canal**

->Puesto que estamos haciendo una reutilización de canales, podemos calcular el número de usuarios del sistema que comparten un mismo canal.

Mid_M = Mcid_Mc / Ccid_Cc móviles

----------Salto de página----------

#$K+**Móviles: Móviles**

->El número de teléfonos **móviles** está en función de la densidad de móviles y de la superficie a cubrir.
->Las expresiones son las siguientes:

Mcid_Mc = DMid_DM * Scid_Sc móviles

Mgid_Mg = Mc * jid_j móviles

Msid_Ms = Mc * Nid_N móviles

----------Salto de página----------

#Radio de cobertura de una célula.

----------Salto de página----------

#Altura de una antena de estación de base.

----------Salto de página----------

#Altura de una antena de un teléfono móvil.

----------Salto de página----------

#Superficie cubierta por una célula.

----------Salto de página----------

#Superficie cubierta por un grupo.

```
# id_mo_usuarios
$ Móviles: Usuarios por Canal
K Usuarios por Canal
+ VDMoviles:15
# id_mo_moviles
$ Móviles: Móviles
K Móvil
+ VDMoviles:20
# id_Radio
# id_hb
# id_hm
# id_Sc
# id_Sg
```

----------Salto de página----------

#Superficie de nuestro sistema de telefonía móvil celular.
----------Salto de página----------

#Número de células que conforman el sistema de telefonía móvil.
----------Salto de página----------
---

id_Ss
id_N

#Número de células que conforman un grupo del sistema de telefonía móvil celular.
----------Salto de página----------

#Número de grupos que conforman el sistema de telefonía móvil.
----------Salto de página----------

#Relación de reutilización cocanal.
----------Salto de página----------

#Distancia de reutilización cocanal.
----------Salto de página----------

#Tráfico ofrecido por un móvil.
----------Salto de página----------

#Densidad de tráfico.
----------Salto de página----------

#Densidad de Móviles.
----------Salto de página----------

#Número de canales disponibles en cada célula.
----------Salto de página----------

#Número de canales disponibles en cada grupo.
----------Salto de página----------

#Número de canales disponibles en el sistema de telefonía móvil.
----------Salto de página----------

#Anccho de banda disponibles en para el sistema de telefonía móvil.
----------Salto de página----------

#Separación entre canales de radiofrecuencia.
----------Salto de página----------

#Tráfico ofrecido por canal.

---

id_j
id_Q
id_q
id_D
id_e
id_DT
id_DM
id_Cc
id_Cg
id_Cs
id_B
id_δf
id_A

----------Salto de página----------

#Tráfico cursado por canal.
----------Salto de página----------

#Tráfico ofrecido por célula.
----------Salto de página----------

#Tráfico cursado por célula.
----------Salto de página----------

#Tráfico ofrecido por grupo.
----------Salto de página----------

#Tráfico cursado por grupo.
----------Salto de página----------

#Tráfico ofrecido por el sistema.
----------Salto de página----------

#Tráfico cursado por el sistema.
----------Salto de página----------

#Probabilidad de bloqueo o de pérdida, que es igual a la fórmula B de Erlang.
----------Salto de página----------

#Tiempo medio de duración de una llamada.
----------Salto de página----------

#Número de móviles por canal.
----------Salto de página----------

#Número de móviles por célula.
----------Salto de página----------

#Número de móviles por grupo.
----------Salto de página----------

---

id_A'
id_Ac
id_Ac'
id_Ag
id_Ag'
id_As
id_As'
id_p
id_T
id_M
id_Mc
id_Mg

*Número de móviles en el sistema de telefonía móvil celular.

id_Ms

A2.2.4 MENU.RTF

En este archivo se encuentra toda la información existenete acerca de el menú de la aplicación:

#$Menú Principal

->El **menú principal** de esta aplicación consta de los siguientes menús desplegables:

~~Móviles~~id_m_moviles
~~Información~~id_m_informacion
~~Planificación Celular~~id_m_planificacion
~~Salir~~id_m_salir
~~Ayuda~~id_m_ayuda
----------Salto de página----------

#$Menú: Móviles

->Este es el primer menú desplegable de la aplicación, donde podemos seleccionar cualquiera de las siguientes opciones:

~~Visualizar Móviles~~id_mo_visualizar
~~Borrar Móviles~~id_mo_borrar
~~Efectuar Llamada Telefónica~~id_mo_llamada
~~Cortar la Comunicación~~id_mo_cortar
~~Seleccionar Móvil...~~id_mo_seleccionar
----------Salto de página----------

```
# id_menu
$ Menú Principal
K Menú; Móvil; Información; Planificación Celular; Salir; Ayuda
+ Indice:15
# id_m_moviles
$ Menú: Móviles
K Menú; Móvil
+ Menu:05
```

Móviles: Visualizar Móviles

->Al seleccionar esta opción, podemos ver 5 móviles (coches) en pantalla, que se irán desplazando dentro de la ~~ventana superior izquierda~~id_ventanasi de la aplicación.
->A partir de ese momento, esta opción no podrá ser nuevamente seleccionada mientras estén presentes los móviles en su ventana correspondiente.
----------Salto de página----------

Móviles: Borrar Móviles

->El usuario podrá acceder a esta opción siempre que se encuentren los móviles presentes en el ~~área de cliente~~id_areacliente de la aplicación.
->También se activa cuando el usuario modifique el tamaño de la ventana principal de la aplicación ya sea utilizando las cajas de *maximizar* o *minimizar*, o también con el propio cursor del ~~ratón~~id_raton, redimensionando los bordes de la ventana, siempre que estos se encuentren presentes en pantalla.
->La última causa por la que se podría activar esta opción es que el usuario mueva alguna caja de diálogo o de mensajes, por encima de la ~~ventana superior izquierda~~id_ventanasi, donde se encuentran activados los móviles.
----------Salto de página----------

Móviles: Efectuar Llamada Telefónica

->Sólo se podrá acceder a esta opción cuando los móviles estén visualizados y la ~~ventana inferior izquierda~~id_ventanaii indique que la llamada anterior está finalizada. A medida que se vayan desplazando los móviles y esta opción esté activada, podrán producirse ciertos **mensajes** de información en pantalla.
----------Salto de página----------

Móviles: Cortar la Comunicación

```
# id_mo_visualizar
$ Móviles: Visualizar
k Móvil; Visualizar Móviles
+ MMoviles:05
# id_mo_borrar
$ Móviles: Borrar
k Móvil; Borrar Móviles
+ MMoviles:10
# id_mo_llamada
$ Móviles: Efectuar Llamada
k Móvil; Efectuar Llamada
+ MMoviles:15
# id_mo_cortar
$ Móviles: Cortar Comunicación
k Móvil; Cortar Comunicación
```

->Sólo se podrá acceder a esta opción, cuando los móviles estén visibles en pantalla y la ventana inferior izquierdaid_ventanaii indique que la llamada efectuadaid_mo_llamada está en curso o perdida.

----------Salto de página----------

Móviles: Seleccionar Móvil...

->Esta opción abre un cuadro de diálogo, que permite hacer ciertas modificaciones sobre los móviles:

Movimientoid_sm_movimiento
Mensajes en Pantallaid_sm_mensajes
Móvil Seleccionadoid_sm_seleccionado
Velocidadid_sm_velocidad
Recibir Llamadasid_sm_recibir

----------Salto de página----------

Seleccionar Móvil: Movimiento

->Dentro de esta caja de grupo podemos seleccionar el tipo de movimiento que realizará el móvil seleccionado.
->Si elegimos la opción *Aleatorio*, el móvil seleccionado se desplazará de forma aleatoria, al igual que el resto de los móviles.
->Si seleccionamos la opción *Controlado por Ratón*, el usuario podrá dirigir al móvil seleccionado a la posición que desee dentro de la ventana superior izquierdaid_ventanasi del área de clienteid_areacliente,desplazando el cursor a la posición deseada y haciendo "clic" con el botón izquierdo del ratón.

----------Salto de página----------

Seleccionar Móvil: Mensajes en Pantalla

```
+ MMoviles:20
# id_mo_seleccionar
$ Móviles: Seleccionar Móvil...
K Móvil; Seleccionar Móvil
+ MMoviles:25
# id_sm_movimiento
$ Seleccionar Móvil: Movimiento
K Seleccionar Móvil; Movimiento
+ MoSeleccionar:05
# id_sm_mensajes
$ Seleccionar Móvil: Mensajes en Pantalla
K Seleccionar Móvil; Mensajes
+ MoSeleccionar:10
```

->Dentro de esta caja de grupo podemos decidir si queremos que la aplicación nos avise cuando el móvil seleccionado ha salido del área de cobertura mientras estaba cursando una llamada telefónica. Esto lo hacemos seleccionado la opción *Salida de la zona* del menú ~~Móviles~~id_m_moviles.
->También podemos decidir si queremos que la aplicación nos avise cuando se ha perdido una llamada telefónica, seleccionado la opción *Canal Bloqueado* del menú *Móviles*.
----------Salto de página----------

#Seleccionar Móvil: Móvil Seleccionado

->Dentro de esta caja de grupo podemos seleccionar uno de los 5 móviles para hacer un seguimiento de este, a lo largo del sistema celular. En la ventana inferior izquierdaid_ventanaii de la aplicación podemos observar cuál es el móvil seleccionado.

----------Salto de página----------

#Seleccionar Móvil: Velocidad

->En el interior de esta caja de grupo, podemos seleccionar una de las tres velocidades posibles con la que se desplazarán los móviles, a través de la ventana superior izquierdaid_ventanasi de la aplicación.

Máxima -> Se desplazan cada 0.5 segundos
Media -> Se desplazan cada segundo
Mínima -> Se desplazan cada 2 segundos

----------Salto de página----------

#Seleccionar Móvil: Recibir Llamadas

->Esta caja de selección nos permite recibir llamadas de forma aleatoria con un 5% de probabilidad de que se produzca una llamada cada vez que se desplacen los móviles, y siempre que se encuentre la opción *llamada* de la ventana inferior izquierdaid_ventanaii finalizada.

----------Salto de página----------

```
# id_sm_seleccionado
$ Seleccionar Móvil: Móvil Seleccionado
k Seleccionar Móvil; Móvil Seleccionado
+ MoSeleccionar:15
# id_sm_velocidad
$ Seleccionar Móvil: Velocidad
k Seleccionar Móvil; Velocidad
+ MoSeleccionar:20
# id_sm_recibir
$ Seleccionar Móvil: Recibir Llamadas
k Seleccionar Móvil; Recibir Llamadas
+ MoSeleccionar:25
```

~~#k~~ Menú: Información

->Este menú desplegable contiene las siguientes opciones:

Cálculo de Dimensiones

*Esta opción activa la ventana ~~Dimensiones~~id_vd_dimensiones en el área de la ~~ventana derecha~~id_ventanad de la aplicación.

Cálculos Radioeléctricos

*Esta opción activa la ventana ~~Radioeléctricos~~id_vd_radioelectricos en el área de la ventana derecha de la aplicación.

Cálculos de Tráfico

*Esta opción activa la ventana ~~Tráfico~~id_vd_trafico en el área de la ventana derecha de la aplicación.

Cálculos sobre los Móviles

*Esta opción activa la ventana ~~Características de los Móviles~~id_vd_moviles en el área de la ventana derecha de la aplicación.

----------Salto de página----------

~~#k~~ Menú: Planificación Celular

->Este menú desplegable contiene las siguientes cajas de diálogo, que permiten seleccionar las condiciones requeridas por cada sistema de telefonía móvil celular.

~~Magnitudes Geométricas~~id_p_geometricas
~~Magnitudes Radioeléctricas~~id_p_radioelectricas
~~Magnitudes de Tráfico~~id_p_trafico

----------Salto de página----------

```
# id_m_informacion
$ Menú: Información
k Menú; Información
+ Menu:10
# id_m_planificacion
$ Menú: Planificación Celular
k Menú; Planificación Celular
+ Menu:15
```

Planificación Celular: Magnitudes Geométricas

->En esta caja de diálogo, el usuario puede seleccionar o definir los valores de las siguientes opciones:

~~Radio de cobertura de las Células~~id_g_radio
~~Área total de la zona a cubrir~~id_g_area
~~Células por grupo~~id_g_celulas
----------Salto de página----------

Magnitudes Geométricas: Radio de Cobertura de las Células

->En este cuadro de edición, el usuario puede introducir un número real para indicar el **radio de cobertura** de cada una de la células que forman nuestro sistema de telefonía móvil celular.
----------Salto de página----------

Magnitudes Geométricas: Área Total de la Zona a Cubrir

->En este cuadro de edición, el usuario puede introducir un número real para indicar el tamaño de la superficie total a cubrir por nuestro sistema de telefonía móvil celular.
----------Salto de página----------

Magnitudes Geométricas: Células por Grupo

->Dentro de este cuadro de grupo, podemos seleccionar el número de células que formaran cada grupo. Este número puede ser 3, 4, 7, ó 12.
----------Salto de página----------

```
# id_p_geometricas
$ Planificación: Magnitudes Geométricas
k Planificación Celular; Magnitudes Geométricas
+ MPlanificacion:05
# id_g_radio
$ Magnitudes Geométricas: Radio de Cobertura
k Magnitudes Geométricas; Radio de Cobertura
+ PGeometricas:05
# id_g_area
$ Magnitudes Geométricas: Area Total
k Magnitudes Geométricas; Area Total
+ PGeometricas:10
# id_g_celulas
$ Magnitudes Geométricas: Células por Grupo
k Magnitudes Geométricas; Célula
+ PGeometricas:15
```

Planificación Celular: Magnitudes Radioeléctricas

->En esta caja de diálogo, encontramos las siguientes variables radioeléctricas:

~~Ancho de Banda Disponible~~id_r_anchobanda
~~Separación entre canales~~id_r_separacion
~~Relación de Protección de Radiofrecuencia~~id_r_proteccion
~~Ley de Propagación de la Señal~~id_r_propagacion
~~Antenas de Estación de Base~~id_r_antenas
----------Salto de página----------

Magnitudes Radioeléctricas: Ancho de Banda Disponible

->En este cuadro de edición, el usuario puede introducir un número real para indicar el **ancho de banda disponible** en nuestro sistema de telefonía móvil celular.
----------Salto de página----------

Magnitudes Radioeléctricas: Separación entre canales

->En esta caja de edición, el usuario puede introducir un número real para indicar la **Separación entre los Canales** asignados a nuestro sistema de telefonía móvil celular.
----------Salto de página----------

Magnitudes Radioeléctricas: Relación de Protección de Radiofrecuencia

->En esta caja de edición, el usuario puede introducir un número real para indicar el valor mínimo permitido de la relación **Señal / Interferencia** en nuestro sistema de telefonía móvil celular.

```
# id_p_radioelectricas
$ Planificación: Magnitudes Radioeléctricas
K Planificación Celular; Magnitudes Radioeléctricas
+ MPlanificacion:10
# id_r_anchobanda
$ Magnitudes Radioeléctricas: Ancho de Banda
K Magnitudes Radioeléctricas; Ancho de Banda
+ PRadioelectricas:05
# id_r_separacion
$ Magnitudes Radioeléctricas: Separacion entre Canales
K Magnitudes Radioeléctricas; Separacion entre Canales
+ PRadioelectricas:10
# id_r_proteccion
$ Magnitudes Radioeléctricas: Relación de Protección
K Magnitudes Radioeléctricas; Relación de Protección
+ PRadioelectricas:15
```

----------Salto de página----------

#@~Magnitudes Radioeléctricas: Ley de Propagación de la Señal

->En este cuadro de edición, el usuario puede introducir un número real para indicar el valor de la **Ley de Propagación de la señal** donde se ubique nuestro sistema de telefonía móvil celular.

->Siendo los valores típicos:

*Antenas elevadas	2
*Zona Urbana	3.5 - 3.9
*Terreno plano	4

----------Salto de página----------

#@~Magnitudes Radioeléctricas: Antenas de Estación de Base

->A medida que las **antenas de estación de base** cubren un área o sector menor, disminuye la interferencia cocanal producida por las antenas que comparten un mismo juego de radiocanales.
->En esta caja de grupo podemos seleccionar uno de los tres tipos de antena que se utilizan en los sistemas de telefonía móvil celular.

~~Omnidireccionales 360º~~id_a_360
~~Sectoriales de 120º~~id_a_120
~~Sectoriales de 60º~~id_a_60

----------Salto de página----------

#@~Antenas de Estación de Base: Omnidireccionales

->Este tipo de antena cubre un área circular de 360º, por lo que sólo se necesita una antena para cubrir el área correspondiente a una célula. Tiene el inconveniente de producir una mayor interferencia cocanal con respecto a las antenas sectoriales.

Señal / Interferencia mínima -> q^4id_q / 6 dB
Señal / Interferencia máxima.-> $1 / (2(q-1)^{-4} + 2*q^{-4} + 2(q+1)^{-4})$ dB

```
# id_r_propagacion
$ Magnitudes Radioeléctricas: Ley de Propagación
k Magnitudes Radioeléctricas; Ley de Propagación
+ PRadioelectricas:20
# id_r_antenas
$ Magnitudes Radioeléctricas: Antenas de Estación de Base
k Magnitudes Radioeléctricas; Antena
+ PRadioelectricas:25
# id_a_360
$ Antenas de Estación de Base: Omnidireccionales
k Antena; Omnidireccionales
+ RAntenas:05
```

----------Salto de página----------

Antenas de Estación de Base: Sectoriales de 120º

->Este tipo de antena cubre un sector de 120° dentro del área de un circulo, por lo que se necesitan **3** antenas para cubrir el área correspondiente a una célula. Se mejora la interferencia cocanal con respecto a las antenas ~~omnidireccionales~~id_a_360, pero tiene el inconveniente de multiplicar por tres la posibilidad de que se produzca el ~~Handover~~id_es_handover.

Señal / Interferencia mínima -> q^4id_q / 2 dB
Señal / Interferencia máxima.-> $1 / ((q+0.7)^{-4} + q^{-4})$ dB

----------Salto de página----------

Antenas de Estación de Base: Sectoriales de 60º

->Este tipo de antena cubre un sector de 60° dentro del área de un circulo, por lo que se necesitan **6** antenas para cubrir el área correspondiente a una célula. Se mejora la interferencia cocanal con respecto a las antenas ~~sectoriales de 120º~~id_a_120, pero tiene el inconveniente de multiplicar por 6 la posibilidad de que se produzca el ~~Handover~~id_es_handover.

Señal / Interferencia mínima -> q^4id_q dB
Señal / Interferencia máxima.-> $(q + 0.7)^4$ dB

----------Salto de página----------

Planificación Celular: Magnitudes de Tráfico

->En esta caja de diálogo, podemos definir dos características que afectarán al tráfico en nuestro sistema:

Densidad de Móviles Requerida

->Esta caja de edición permite introducir un número real para indicar el número de móviles que se va a atender en cada kilómetro cuadrado de nuestro sistema de telefonía móvil.

----------Salto de página----------

```
# id_a_120
$ Antenas de Estación de Base: Sectoriales 120º
k Antena; Sectoriales 120º
+ RAntenas:10
# id_a_60
$ Antenas de Estación de Base: Sectoriales 60º
k Antena; Sectoriales 60º
+ RAntenas:15
# id_p_trafico
$ Planificación: Magnitudes de Tráfico
k Planificación Celular; Magnitudes de Tráfico
+ MPlanificacion:15
```

Menú: Salir

->Esta opción del menú principal abre una caja de mensajes, donde se le pregunta al usuario si realmente quiere abandonar la aplicación o no.

----------Salto de página----------

Menú: Ayuda

->Nuestro menú de ayuda presenta las siguientes opciones:

Índice

*Seleccionando esta opción accedemos al índice de la ayuda donde se encuentran los temas principales de la ayuda.

Área de cliente

*Con esta opción accedemos al tema de ayuda referente al área de cliente de la aplicación.

Menú principal

*Con esta opción accedemos al tema de ayuda referente al menú de la aplicación.

Ratón

*Con esta opción accedemos al tema de ayuda referente al uso del ratón en la aplicación.

Mensajes de información

*Con esta opción accedemos al tema de ayuda referente a los tipos mensajes que se producen en la aplicación.

Buscar ayuda sobre

*Con esta opción accedemos a un cuadro de diálogo donde podemos buscar ayuda sobre algún término o palabra clave que se utilice en la aplicación.

Uso de la ayuda

*Seleccionando esta opción aprenderemos a utilizar la ayuda de Windows.

```
# id_m_salir
$ Menú: Salir
k Menú; Salir
+ Menu:20
# id_m_ayuda
$ Menú: Ayuda
k Menú; Ayuda
+ Menu:25
```

Acerca de...

*Este cuadro de diálogo contiene el nombre del autor del proyecto y el año en el que realizó esta aplicación.

A2.2.5 RATON.RTF

Este es el archivo donde se explica el funcionamiento del ratón en nuestra aplicación:

#[#]Ratón

->Manejar la aplicación con el **ratón** es muy sencillo, éste se presenta en la pantalla con forma de flecha. Su forma puede variar cuando se realiza una llamada pasando por uno de los siguientes cursores:

Flecha	->	Llamada finalizada
Teléfono	->	Llamada en curso
Cruz	->	Llamada perdida

Botón Izquierdo

->Utilizaremos este botón para seleccionar alguna opción de los menús o de los cuadros de diálogo en nuestra aplicación.
->También podemos desplazar el móvil seleccionado dentro de la ~~ventana superior izquierda~~id_ventanasi al pulsar este botón, cuando el cursor se encuentre en la posición deseada.

Botón Derecho

->Mientras el cursor permanezca en el ~~área de cliente~~id_areacliente de nuestra aplicación, al pulsar este botón se cambiará la opción seleccionada en el menú *Información*, de forma consecutiva. Visualizaremos en pantalla la ventana correspondiente al tema seleccionado.

---

id_raton
$ Ratón
k Ratón
+ Indice:20

A2.2.6 MENSAJES.RTF

En este archivo se informa de los distintos tipos de mensajes que se pueden producir en nuestra aplicación:

#$kMensajes de Información

->En nuestra aplicación, aparecerán estos mensajes cuando ocurra alguno de estos sucesos:

* Pulsar el botón izquierdo del ~~ratón~~id_raton fuera de la ~~ventana superior izquierda~~id_ventanasi cuando el móvil está en el modo ~~Controlado por ratón~~id_sm_movimiento.

* Pulsar el botón izquierdo del ratón dentro del ~~área de cliente~~id_areacliente cuando el móvil está en el modo ~~Automático~~id_sm_movimiento.

* ~~Recibir una llamada telefónica~~id_sm_recibir.

* Perder la llamada telefónica por no encontrarse ningún canal disponible, si está habilitada la opción ~~Canal Bloqueado~~id_sm_mensajes.

* Perder la llamada telefónica por encontrarse el móvil fuera del área de cobertura, si está habilitada la opción ~~Salida Zona~~id_sm_mensajes.

* Especificar las ~~magnitudes geométricas~~id_p_geometricas de forma contradictoria.

* Especificar las ~~magnitudes radioeléctricas~~id_p_radioelectricas de forma contradictoria.

id_mensajes
$ Mensajes de Información
k Mensajes de información
+ Indice:25

A2.2.7 SISTEMAS.RTF

En este archivo se describe una introducción de los sistemas de telefonía móvil celular:

#@k**Introducción a los Sistemas de Telefonía Móvil Celular**

->La telefonía móvil consiste en ofrecer un acceso vía radio a un abonado de telefonía, de tal forma que pueda realizar y recibir llamadas dentro del área de cobertura del sistema.

~~Descripción~~id_sc_descripcion
~~Nodos~~id_sc_nodos
~~Geometría Celular~~id_sc_geometria
~~Estructura de las Células~~id_sc_celulas
~~Estructura del Sistema~~id_sc_sistema
~~Evolución del Sistema~~id_sc_evolucion
~~Reutilización de Canales de Frecuencia~~id_sc_reutilizacion
----------Salto de página----------

#@k**Sistemas Celulares: Descripción**

->Se basan en dividir la zona de cobertura pretendida, en zonas más pequeñas denominadas células, que serán de igual forma y tamaño. Limitando convenientemente la potencia con que se emite cada frecuencia, permite la reutilización de las mismas a distancias bastante cortas, aumentando la capacidad de los sistemas.
->Un sistema celular consta de una serie de células cubiertas cada una por un sistema de radio que permite la conexión de los terminales móviles al sistema (Estación de Base) y un sistema de conmutación que permite la interconexión entre las estaciones de base y la conexión del sistema a la red de conmutación pública.
->Para una separación suficiente entre celdas determinada por el valor admisible de la relación de protección de radiofrecuencia, se creará una superestructura de células constituida por grupos con encaje mutuo perfecto. Dentro de cada grupo no se repetirá frecuencia alguna y la asignación de frecuencias será la misma para todos ellos.

```
# id_sistemas
$ Sistemas Celulares
k TMA; Sistemas Celulares
+ Indice:30
# id_sc_descripcion
$ Sistemas Celulares: Descripción
k Sistemas Celulares; Descripción
+ Sistemas:05
```

----------Salto de página----------

#K$Sistemas Celulares: Nodos

->Los **nodos** más importantes de un sistema de telefonía móvil son:
~~Centro de Conmutación del Servicio Móvil~~id_n_centro
~~Registro de Localización~~id_n_registro
~~Área de Estaciones de Base~~id_n_base
~~Área de Localización~~id_n_localizacion
~~Área de Conmutación~~id_n_conmutacion
~~Red Pública del Servicio Móvil Terrestre~~id_n_red
~~Área de Servicio~~id_n_servicio
~~Área de Sistema~~id_n_sistema
----------Salto de página----------

#K$Nodos: Centro de Conmutación del Servicio Móvil (MSC)

->Constituye el interfaz entre la parte radio y la red telefónica pública conmutada. El **MSC** realiza todas las funciones de señalización necesarias para establecer llamadas terminadas u originadas desde las estaciones móviles. Cada estación móvil está registrada en un **MSC** para efectos de tarificación y cobro y para definir dicha estación móvil con todos los datos acerca de categoría, restricciones, etc.
->Con objeto de obtener una cobertura adecuada en un área geográfica adecuada, se precisan un cierto número de estaciones de base (transmisores/receptores de radio). Cada **MSC** puede disponer interfaces con varias estaciones de base; al objeto de cubrir un país pueden precisarse varios **MSC**.
----------Salto de página----------

#K$Nodos: Registro de Localización

->Con objeto de establecer una llamada dirigida a una estación móvil, la red debe conocer donde se encuentra dicha estación móvil. Esta información se almacena en un nodo denominado **registro de localización**.
----------Salto de página----------

```
# id_sc_nodos
$ Sistemas Celulares: Nodos
K Sistemas Celulares; Nodos
+ Sistemas:10
# id_n_centro
$ Nodos: Centro de Conmutación del Servicio Móvil
K Nodos; Centro de Conmutación del Servicio Móvil
+ SNodos:05
# id_n_registro
$ Nodos: Registro de Localización
K Nodos; Registro
+ SNodos:10
```

#K**Nodos: Área de Estaciones de Base**

->Es la parte de la red cubierta por una **estación de base**. Cada estación móvil en el área de una **estación de base** puede conectarse por radio a la misma.
----------Salto de página----------

#K**Nodos: Área de Localización**

->Se define como un área en la cuál una estación móvil puede moverse libremente sin alterar los datos del ~~registro de localización~~id_n_registro. Puede comprender varias estaciones de base.
----------Salto de página----------

#K**Nodos: Área de Conmutación**

->Consiste en la parte de la red cubierta por un ~~MSC~~id_n_centro. Puede comprender varias ~~áreas de localización~~id_n_localizacion.
----------Salto de página----------

#K**Nodos: Red Pública del Servicio Móvil Terrestre (PLMN)**

->Conjunto de ~~áreas de conmutación~~id_n_conmutacion dentro de un plan de numeración común y un plan de encaminamiento también común. Las centrales de conmutación constituyen las interfaces funcionales entre la red fija y una red pública móvil.
----------Salto de página----------

```
# id_n_base
$ Nodos: Area de Estaciones de Base
K Nodos; Estaciones de Base
+ SNodos:15
# id_n_localizacion
$ Nodos: Area de Localización
K Nodos; Localización
+ SNodos:20
# id_n_conmutacion
$ Nodos: Area de Conmutación
K Nodos; Area de Conmutación
+ SNodos:25
# id_n_red
$ Nodos: Red Pública del Servicio Móvil Terrestre
K Nodos; Red Pública del Servicio Móvil Terrestre
+ SNodos:30
```

Nodos: Area de Servicio

->Area en la cuál una estación móvil puede ser alcanzada por un abonado de la red telefónica fija, sin que tenga que conocer la situación actual del móvil dentro del área. Puede estar constituida por varias ~~PLMNs~~id_n_red y comprender un país. El sistema de registro de localización asociado con cada **área de servicio** debe contener entonces una lista de todas las estaciones móviles localizadas dentro de la misma.

----------Salto de página----------

Nodos: Área de Sistema

->El área de sistema se define como un conjunto de ~~PLMNs~~id_n_red o de ~~áreas de servicio~~id_n_servicio accesibles para una estación móvil siempre dentro de la compatibilidad completa de diversos sistemas móviles.

----------Salto de página----------

Sistemas Celulares: Geometría Celular

->Para simplificar el estudio, se suponen todos los transmisores de idénticos parámetros, altura de la antena, terreno homogéneo con las mismas condiciones de propagación en toda la ~~zona de cobertura~~id_n_servicio. Esto conduce a esquemas regulares de disposición de canales, basados en celdas homogéneas del mismo tamaño y forma.
->Si en cada celda se utilizan antenas omnidireccionales, la zona de cobertura sería aproximadamente circular. Sin embargo, las coberturas circulares o no recubren el plano o producen solapes, esto último implica una reducción de la eficacia espectral, porque en un mismo punto se emplean dos frecuencias.
->Se exigen entonces formas poligonales que recubren el plano sin solape. Se obtiene que el hexágono proporciona la mayor superficie de celda y por lo tanto el mínimo número de celdas necesario para la cobertura de una zona.

----------Salto de página----------

```
# id_n_servicio
$ Nodos: Area de Servicio
K Nodos; Area de Servicio
+ SNodos:35
# id_n_sistema
$ Nodos: Area de Sistema
K Nodos; Sistema
+ SNodos:40
# id_sc_geometria
$ Sistemas Celulares: Geometría Celular
K Sistemas Celulares; Geometría Celular
+ Sistemas:15
```

#Sistemas Celulares: Estructura de Células

->El número de células de un grupo se determina de manera que pueda repetirse de forma no interrumpida en el ~~área de cobertura~~id_n_servicio. Solamente algunas configuraciones lo permiten. Los grupos típicos se basan en 3, 4, 7, o 12 células, según la siguiente ecuación:

jid_j = $(a + b)^2 - a*b$ células.

*Donde a y b son números enteros positivos.

->El número de células en cada grupo tiene significativa importancia en la capacidad total del sistema. Cuanto menor es el número de las células mayor es el número de canales por célula y en consecuencia el tráfico es más alto. Sin embargo debe buscarse el punto de equilibrio, porque si se utilizan más canales por célula y el tamaño del grupo es menor (menos células), la distancia entre las células que utilizan los mismos canales es menor, con la consecuencia de que la interferencia entre grupos adyacentes aumenta (interferencia cocanal).
->El tráfico de un área en particular puede ser aumentado si se reduce el tamaño de la célula de manera que aumenta el número total de radiocanales disponibles en el área.
----------Salto de página----------

#Sistemas Celulares: Estructura del Sistema

->Las estaciones de base ubicadas en el centro de cada célula están unidas a una ~~central de conmutación~~id_n_conmutacion que en esencia es una central de conmutación modificada para el sistema celular. Una red celular consistirá en la práctica en varias centrales de conmutación interconectadas entre sí. Esta configuración permite la realización completa de los distintos tipos de llamada, como son llamadas de móvil a fijo, de fijo a móvil y de móvil a móvil.
->El sistema celular incluye dos prestaciones importantes a la hora de permitir el mantenimiento de las comunicaciones con el móvil:

~~Registro~~id_es_registro
~~Handover~~id_es_handover
----------Salto de página----------

```
# id_sc_celulas
$ Sistemas Celulares: Estructura de Células
k Sistemas Celulares; Célula
+ Sistemas:20
# id_sc_sistema
$ Sistemas Celulares:Estructura del Sistema
k Sistemas Celulares; Sistema
+ Sistemas:25
```

#@~Estructura de Células: Registro

->Dentro del sistema se reservan un número de canales radio como canales de señalización. Adicionalmente la red se divide en un número de área de tráfico, cada área consiste en un grupo de células.
->La estación de base genera un código de identificación correspondiente con el área de tráfico a la que pertenece, como parte de la información transmitida por los canales de señalización.
->El abonado móvil que viaja a lo largo de la red, monitoriza el canal de señalización cuya emisión es más potente. Como el móvil se desplaza de una célula ala siguiente, detecta un deterioro en la calidad de recepción en el canal común de señalización utilizado y en consecuencia comienza la búsqueda de otro canal con señal más potente.
->Una vez que el móvil ha sintonizado la nueva señal, hay dos opciones posibles. La primera es que habiendo cambiado de célula no se abandone el área de tráfico por lo que el registro no se modificará. La segunda opción es que el móvil también haya cambiado de área de tráfico, en este caso el móvil transmite su identificación a la nueva estación de base, que transfiere la información al ~~centro de conmutación~~id_n_conmutacion. De esta forma el móvil ha registrado su ubicación de manera que la red es capaz de encaminar la llamada hacia el móvil de modo rápido y eficiente.
----------Salto de página----------

#@~Estructura de Células: Handover

->A medida que el móvil se mueve en el área de cobertura, puede cruzar la frontera entre células mientras la llamada está en progreso. Puesto que la conversación no se interrumpe hasta que se cambia de célula, la estación de base supervisa la señal recibida del móvil y detectará cualquier deterioro de la señal en los bordes de la célula. En este momento la estación de base informará al móvil que es necesario el cambio de célula. La ~~central de conmutación~~id_n_conmutacion ordena a las estaciones de base de las células adyacentes que supervisen la señal del móvil, y elige la mejor célula a la que transferir la llamada. En la nueva célula se busca un canal de conversación y se instruye al móvil, todavía en la célula original para que seleccione el nuevo canal.
----------Salto de página----------

```
# id_es_registro
$ Estructura del Sistema: Registro
K Estructura del Sistema; Registro
+ SSistema:05
# id_es_handover
$ Estructura del Sistema: Handover
K Estructura del Sistema; Handover
+ SSistema:10
```

Sistemas Celulares: Evolución del Sistema

->El método clásico de establecimiento de un sistema celular, es a partir de un primer esquema con pocas células de gran tamaño. La expansión se efectúa mediante subdivisión de las células, lo que provoca un exceso de canales y equipo inicialmente.
->Por este motivo la transición debe ser gradual, aplicándose el concepto de recubrimiento, que consiste en ir añadiendo celdas dentro de la zona de servicio requerida. Estas celdas de solape pueden introducirse una a una cuando se requiera y la densidad de usuarios lo haga rentable.
->La densidad de tráfico no es homogénea, por lo que la subdivisión celular tampoco lo será. En el caso de una amplia cobertura territorial que comprenda un núcleo urbano, en el centro de éste la densidad de tráfico será máxima e irá decreciendo hacia las afueras.
->Un atributo importante de los sistemas celulares es la coexistencia de células de diferentes tamaños.
->Se puede proceder a una ulterior subdivisión sin necesidad de emplear más desplazamientos de estaciones de base, sectorizando la cobertura, con lo que se mejora notablemente la interferencia cocanal.

----------Salto de página----------

Sistemas Celulares: Reutilización de Canales de Frecuencia

->Un radiocanal consiste en un par de frecuencias, cada una para una dirección de transmisión utilizadas en un modo de transmisión Full-Duplex.
->Un radiocanal F1 utilizado en una zona geográfica llamada célula con un radio de cobertura R, puede ser utilizada en otra célula con el mismo radio de cobertura a una distancia D o distancia de reutilización.

{bmc cocanal.bmp}

->La interferencia dual producida por el uso común del mismo canal se denomina interferencia cocanal.

```
# id_sc_evolucion
$ Sistemas Celulares: Evolución
k Sistemas Celulares; Evolución
+ Sistemas:30
# id_sc_reutilizacion
$ Sistemas Celulares: Reutilización
k Sistemas Celulares; Reutilización
+ Sistemas:35
```

BIBLIOGRAFÍA

La bibliografía consultada para desarrollar este proyecto se encuentra disponible en la biblioteca de la EUITT de Las Palmas de Gran Canaria. Junto con los títulos de los libros se incluirá la referencia de estos dentro de la biblioteca.

[1] El Gran Libro del Windows 3.1.

Editorial: MARCOMBO
Referencia: 681.3 GRA

[2] Manual de Borland C++ 3.x.

Autores: CHRIS H.PAPPAS / WILLIAM H.MURRAY
Editorial: MC GRAW-HILL
Referencia: 681.3.06 PAP MAN

[3] Programación en Borland C++ 3.x.

Autores: LEE ATKINSON / MARK ATKINSON
Editorial: ANAYA MULTIMEDIA
Referencia: 681.3.06 LEE PRO

[4] Aplique Turbo C++ para Windows.

Autor: HERBERT SCHILDT
Editorial: MC GRAW-HILL
Referencia: 681.3.06 SCH APL

[5] Windows 3.1 Técnicas de Programación.

Autor: PETER NORTON

Referencia: 681.3 NOR GUI

[6] Advanced Windows Programing.

Autor: MARTIN HELLER

Referencia: 681.3 HEL ADV

[7] Fundamentos y Desarrollo de Programas en Windows 3.x.

Autor: JAIME PEÑA TRESANCOS

Editorial: ANAYA MULTIMEDIA

Referencia: 681.3.06 PEÑ FUN

[8] Programación en Windows con Borland C++.

Autor: WILLIAM ROETZEIM

Editorial: ANAYA MULTIMEDIA

Referencia: 681.3.06 ROE PRO

[9] Guía de Programación en Windows. Librería ObjectWindows.

Autor: NAMIR CLEMENT SHAMMAS

Editorial: ANAYA MULTIMEDIA

Referencia: 681.3.06 SHA GUI

[10] Sistemas de Telecomunicación Volumen II. Transmisión por Radio.

Autor: HERNANDO RABANOS

[11] Telecomunicaciones Móviles.

Editorial: MARCOMBO

Referencia: 621.395

[12] Planificación de Redes Digitales.

Autor: EMILIO LERA

Referencia: 681.327 LER PLA

[13] Mobile Cellular Telecomunications Systems.

Autor: WILLIAM C.Y. LEE

Referencia: 621.395 LEE MOB

[14] Cellular Radio Analog and Digital Systems.

Autor: ASHA MEHROTRA

Referencia: 621.396 MEH CEL

[15] Microwave Mobile Comunications.

Autor: William C. Jakes

Referencia: 621.396 MIC MIC

[16] Hacia las Comunicaciones Móviles Integradas.

Autor: V. Casares

Editorial: Mundo Electrónico

Referencia: nº 242, pp.21-38, Octubre 1993

[17] Sistemas de Comunicaciones Móviles.

Autor: F. Casadevall

Editorial: Mundo Electrónico

Referencia: nº 234, pp.51-65, Diciembre 1992