Capitulo I

Programación Orientada a Objetos

La programación orientada a objetos, ha tomado las mejores ideas de la programación estructurada y los ha combinado con varios conceptos nuevos y potentes que incitan a contemplar las tareas de programación desde un nuevo punto de vista. La programación orientada a objetos, permite descomponer mas fácilmente un problema en subgrupos de partes relacionadas del problema. Entonces, utilizando el lenguaje se pueden traducir estos subgrupos a unidades autocontenidas llamadas objetos.

El término Programación Orientada a Objetos (POO), hoy en día ampliamente utilizado, es difícil de definir, ya que no es un concepto nuevo, sino que ha sido el desarrollo de técnicas de programación desde principios de la década de los setenta, aunque sea en la década de los noventa cuando ha aumentado su difusión, uso y popularidad. No obstante, se puede definir POO como una técnica o estilo de programación que utiliza objetos como bloque esencial de construcción.

Un objeto es una unidad que contiene datos y las funciones que operan sobre esos datos. A los elementos de un objeto se les conoce como miembros; las funciones que operan sobre los objetos se denominan métodos y los datos se denominan miembros datos.

1.1 ORIGENES DE LA PROGRAMACION ORIENTADA A OBJETOS.

1.ª Etapa. Lenguajes Ensambladores. La unidad de programación es la instrucción, compuesta de un operador y los operandos. El nivel de abstracción que se aplica es muy bajo.

2.ª Etapa. Lenguajes de Programación: Fortran, Algol, Cobol. Los objetos y operaciones del mundo real se podían modelar mediante datos y estructuras de control separadamente. En esta etapa el diseño del software se enfoca sobre la representación del detalle procedimental y en función del lenguaje elegido. Conceptos como: refinamiento progresivo, modularidad procedimientos y programación estructurada son conceptos básicos que se utilizan en esta etapa. Existe mayor abstracción de datos.

3.ª Etapa. Se introducen en esta etapa los conceptos de abstracción y ocultación de la información.

4.ª Etapa. A partir de los años setenta se trabaja sobre una nueva clase de lenguajes de simulación y sobre la construcción de prototipos tales como Simula-70 y basado en parte de éste, el Smalltalk. En estos lenguajes, la abstracción de datos tiene una gran importancia y los problemas del mundo real se representan mediante objetos de datos a los cuales se les añade el correspondiente conjunto de operaciones asociados a ellos. Términos como Abstracción de datos, objeto, encapsulación entre otros, son conceptos básicos sobre la que se fundamenta la POO.

1.2 CONCEPTOS DE LA PROGRAMACION ORIENTADA A OBJETOS.

La POO representa una metodología de programación que se basa en las siguientes características:

1) Los diseñadores definen nuevas clases (o tipos) de objetos.
2) Los objetos poseen una serie de operaciones asociadas a ellos.
3) Las operaciones tienden a ser genéricas, es decir, operan sobre múltiples tipos de datos.
4) Las clases o tipos de objetos comparten componentes comunes mediante mecanismos de herencia.

Objeto: Una estructura de datos y conjunto de procedimientos que operan sobre dicha estructura. Una definición más completa de objeto es: una entidad de programa que consiste en datos y todos aquellos procedimientos que pueden manipular aquellos datos; el acceso a los datos de un objeto es solamente a través de estos procedimientos, únicamente estos procedimientos pueden manipular, referenciar y/o modificar estos datos.

Para poder describir todos los objetos de un programa, conviene agrupar éstos en clases.

Clase: Podemos considerar una clase como una colección de objetos que poseen características y operaciones comunes. Una clase contiene toda la información necesaria para crear nuevos objetos.

Encapsulación: Es una técnica que permite localizar y ocultar los detalles de un objeto. La encapsulación previene que un objeto sea manipulado por operaciones distintas de las definidas. La encapsulación es como una caja negra que esconde los datos y solamente permite acceder a ellos de forma controlada.

Las principales razones técnicas para la utilización de la encapsulación son:

1) Mantener a salvo los detalles de representación, si solamente nos interesa el comportamiento del objeto.
2) Modificar y ajustar la representación a mejores soluciones algorítmicas o a nuevas tecnologías de software.

Abstracción: En el sentido mas general, una abstracción es una representación concisa de una idea o de un objeto complicado. En un sentido mas especifico, la abstracción localiza y oculta los detalles de un modelo o diseño para generar y manipular objetos.

Una abstracción tiene un significado más general que la encapsulación, pudiendo hablar de abstracción de datos en lugar de encapsulación de datos.

Como resumen de los 3 conceptos expuestos anteriormente podemos decir que:

1) Los objetos son encapsulaciones de abstracciones en la POO.
2) La unidad de encapsulación en la POO es el objeto.

Una clase es un tipo: Un objeto es una instancia de ese tipo. Además, la clase es un concepto estático: una clase es un elemento reconocible en el texto del programa.

Un objeto es un concepto puramente dinámico, el cual pertenece, no al texto del programa, sino a la memoria de la computadora, donde los objetos ocupan un espacio en tiempo de ejecución una vez que haya sido creado.

La programación orientada a objetos, ha tomado las mejores ideas de la programación estructurada y los ha combinado con varios conceptos nuevos y potentes que incitan a contemplar las tareas de programación desde un nuevo punto de vista. La programación orientada a objetos, permite descomponer mas fácilmente un problema en subgrupos de partes relacionadas del problema. Entonces, utilizando el lenguaje se pueden traducir estos subgrupos a unidades autocontenidas llamadas objetos.

Objetos: Un objeto es una entidad lógica que contiene datos y un código que manipula estos datos; el enlazado de código y de datos, de esta manera suele denominarse encapsulación.
Cuando se define un objeto, se esta creando implícitamente un nuevo tipo de datos.
Polimorfismo: Significa que un nombre se puede utilizar para especificar una clase genérica de acciones.

Herencia: La herencia es un proceso mediante el cual un objeto puede adquirir las propiedades de otro objeto.

1.3 PRESENTACION DE LAS CLASES Y LOS OBJETOS

Objeto: Un objeto es una entidad abstracta que tiene las características de un objeto real.
Los objetos se crean y eliminan durante la ejecución del programa, además interactúan con otros objetos. Los objetos son construcciones de programación que se obtienen a partir de entidades llamadas clases. La definición de una clase se conoce como *instanciación* de clases.

Para crear un objeto, es preciso definir primero su forma general utilizando la palabra reservada **class**. Una class es parecida a una estructura, es un tipo definido por el usuario que determina las estructuras de datos y las operaciones asociadas con este tipo.
Las clases son como plantillas o modelos que describen como se construyen ciertos tipos de objetos, cada vez que se construye un objeto de una clase se crea una instancia de esa clase, por consiguiente; los objetos son instancias de clases.

Una clase es una colección de objetos similares y un objeto es una instancia de una definición de una clase; una clase puede tener muchas instancias y cada una es un objeto independiente.
Una clase es simplemente un modelo que se utiliza para describir uno o mas objetos del mismo tipo.

Así, por ejemplo sea una clase ventana, un tipo de dato, que contenga los miembros dato:

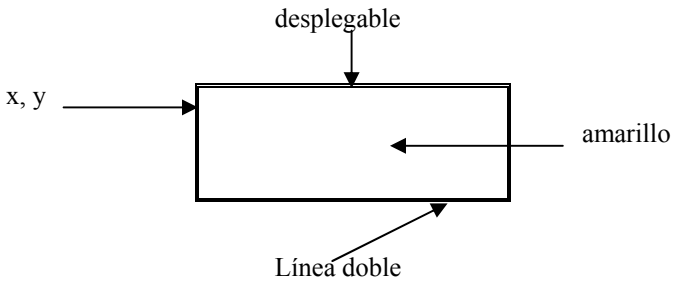
posx, posy
tipo_ventana
tipo_borde
color_ventana

y unas funciones miembro:

mover_horizontal
mover_vertical

Un objeto de la clase ventana, es una ventana concreta (una instancia de la clase) cuyos datos tienen por valores:

posx	x
posy	y
tipo_ventana	desplegable
tipo_borde	linea doble
color_ventana	amarillo



1.3.1 DEFINICIÓN DE UNA CLASE.

Una *clase* es la evolución natural de una estructura, la existencia de clases es la característica más significativa que convierte a C++ en un lenguaje orientado a objetos. Las clases son estructuras que contienen no sólo declaraciones de datos, sino también declaraciones de funciones. Las funciones se conocen como funciones miembro, e indican qué tipos de cosas puede hacer una clase. La palabra reservada ***class*** introduce una declaración de clase.

1.3.2 IDENTIFICADORES DE CLASE.

La longitud máxima para un identificador de clase es 32 caracteres. Una convensión que se adopta en todas las clases de borland es utilizar nombres que comiencen con una letra mayúscula para denotar clases y estructuras globales.

1.3.2.1 CUERPO DE UNA CLASE.

La forma general de la declaración de una clase es:

```
class Nombre_de_la_clase{
        datos y funciones privados
                .
                .
                .
 public:
datos y funciones publicas
                .
                .
                .
}lista de objetos;
```

Una clase puede contener tanto partes publicas como partes privadas, por defecto, todos los elementos que se definen en la clase son privados; esto significa que no pueden acceder a ellas ninguna función que no sea miembro de la clase.

```
 Class Counter{

        long count;                     //  variable privada , variable miembro de la clase

Public:

void SetValue(long);            //  Funciones públicas, funciones miembro de la clase
        long GetValue();
};
```

La variable long count, no está disponible o no se puede usar por otras funciones que no están declaradas en la clase, por lo que tratar de hacer esto es erróneo:

```
void main()
{

count = 3.111;

}
```

Una clase puede tener tantas variables como necesite. Estas pueden ser de cualquier tipo, incluyendo otras clases, apuntadores a objetos de clases e incluso apuntadores a objetos dinámicamente asignados.

Las funciones miembro SetValue(long) y GetValue(). Solo están declaradas dentro de la clase, la definición de estas funciones sería así:

```
void Counter::SetValue(long value)
{ count = value;  }

long Counter::GetValue()
{
        return count;
}
```

1.3.3 USO DE UNA CLASE.

Ya que se ha definido la clase, se debe definir un objeto con ella. Las variables de una clase se definen de igual manera que como se definen las variables de tipo estructura.
Para el ejemplo de la clase anterior, si quiere declarar un objeto Gente de tipo Counter, lo podría hacer asi:

```
Class Counter{
        .
        .
public:
        .
        .
}Gente;
```

O la declaración la podría hacer de la siguiente forma:

```
Counter  Gente;
```

En algunos lenguajes orientados a objetos, Smalltalk en particular, la definición de una variable de clase se denomina *instanciación de la clase*.

Una instanciación es simplemente una instancia de una clase en la forma de una variable específica.

Las variables instanciadas a partir de clases son objetos.

El objeto Gente se podría usar así en un programa:

```
void main()
{
        Counter  Gente;                 //    Declaración de un objeto
        Gente.SetValue(1000);           //     Invocación a función miembro de Counter
        long value = GetValue();        //     Invocación a función miembro de Counter
}
```

La iniciación se tiene que hacer a través de sus funciones miembro, por lo que hacer lo siguiente sería un error.

```
void main()
{

        Counter  Gente;
        Gente = 1000;       //   error, la variable no esta disponible en la función main()
        long value = GetValue();
}
```

El código anterior no hace mucho, pero ilustra 2 aspectos importantes:
La declaración de un objeto dentro de una función y la invocación de funciones miembro de un objeto.

En otro ejemplo, ésta clase define un tipo llamado cola, que se utiliza para crear un objeto de tipo cola.

```
# include <iostream.h>

class cola{
        int c[100];
        int posfin, posprin;

public:
        void iniciar(void);
        void ponent(int i);
        int quitaent(void);

};
```

Cuando llega el momento de codificar realmente una función que es miembro de una clase, es preciso decir al compilador a que clase pertenece la función, calificando el nombre de la función con el nombre de la clase del cual es miembro. p.e.

```
void cola :: ponent(int i)
{
        if(posfin>=100)
          {
             cout<<"la cola esta llena ";
        return;
           }
        posfin++;
        c[posfin] = i;
}
```

El :: se llama operador de resolución de ámbito; indica al compilador que la función ponent(int i) pertenece a la clase cola, o dicho de otra manera, ponent(int i) está dentro del ámbito de cola.

Para llamar a una función miembro desde una parte del programa que no sea parte de la clase, se debe utilizar el nombre del objeto y el operador punto. p.e.

```
Cola a, b;      //   se crean 2 objetos tipo cola.

a.iniciar();    //  llama a la función iniciar para el objeto a.
```

Consideremos el siguiente ejemplo, de un programa en C++, aunque en una aplicación real la declaración de las clases debe estar contenida en un archivo de cabecera.

```
# include <iostream.h>

class cola{
        int c[100];
        int posfin, posprin;
public:
        void iniciar(void);
        void ponent(int i);
        int quitaent(void);
};


main(void)
{
        cola a, b;
        a.iniciar();
        b.iniciar();
```

```
        a.ponent(15);
        b.ponent(39);
        a.ponent(55);
        b.ponent(19);
        cout<<a.quitaent() << " ";
        cout<<b.quitaent() << " ";
        cout<<a.quitaent() << " ";
        cout<<b.quitaent() << " ";
        return 0;
}

void cola::iniciar()
{
        posprin=posfin=0;
}

void cola::ponent(int i)
{
        if(posfin==0)
          {
             cout<<"la cola esta llena ";
        return;
           }
        posfin++;
        c[posfin] = i;

}
int cola::quitaent(void)
{
        if(posfin==posprin)
        {
          cout<<"la cola está vacía";
        return 0;
        }
        posprin++;
        return c[posprin];
}
```

1.4 CONTROL DE ACCESO A UNA CLASE.

La tarea de una clase consiste en ocultar la mayor cantidad de información posible. Por lo tanto es necesario imponer ciertas restricciones a la forma en que se puede manipular una clase. Existen 3 tipos de usuario de una clase:

1.- La clase misma.
2.- Usuarios genéricos.
3.- Clases derivadas.

Cada tipo de usuarios tiene privilegios de acceso asociados a una palabra clave:

1.- Private.
2.- Public.
3.- Protected.

Ejemplo:

```
class  controlAcceso{
        int a;
public:
        int b;
        int fi(int a);
protected:
        int c;
        float C1(float t);
};
```

Cualquier declaración que aparezca antes de cualquiera de las tres palabras clave, por default es private; así, int a; es private.

1.4.1 MIEMBROS DE LA CLASE PRIVATE.

Los miembros de la clase private tienen el mas estricto control de acceso. Solo la clase misma puede tener acceso a un miembro private. En este ejemplo nadie puede usar la clase ya que todo es private.

```
Class Privada{

        long valor;
        void F1();
        void F2();

};
```

```
void main()
{

        privada objeto1;            //  Se crea objeto1 de clase privada.
        long L = &objeto.valor;     //  acceso no valido por ser private.
        objeto1.F1();               //  acceso no valido por ser private.
        objeto1.F2();               //  acceso no valido por ser private.

}
```

Para poder tener acceso necesitaría que las funciones miembro fueran declaradas en la sección public.

1.4.2 MIEMBROS DE LA CLASE PUBLIC:

Para utilizar un objeto de una clase, usted debe tener acceso a datos miembro, a funciones miembro o a ambos. Para hacer que algunos datos o funciones sean accesibles, se declaran en la sección public.

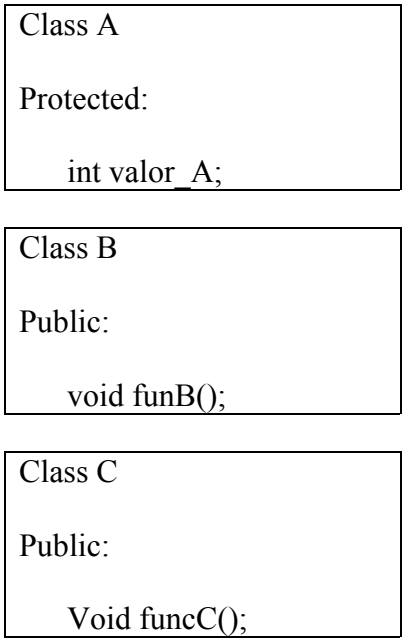
```
class Ej_public{

public:
        int variable;
        void función1();

};

void Ej_public::función1(){}
void main()
{

        Ej_public Objeto2;
        int i = Objeto2.variable;
        Objeto2.función1();

}
```

Cualquier cosa que se declara en la sección public, hace posible el acceso ilimitado a cualquier persona.

1.4.3 MIEMBROS DE LA CLASE PROTECTED.

Cuando se define una clase que se utiliza subsiguientemente como clase de base para otras clases, se puede hacer que los miembros estén accesibles solo para funciones de las clases derivadas mediante el uso de la palabra clave protected.

Considere una jerarquía de objetos como se ilustra a continuación:



La jerarquía de clase se puede expresar con código así:

```
class A{
Protected:
        int valor_A;
};

class B{
public:
        void funB();
};

class C{
public:
        void funC();
};
```

La propiedad de ser protected se extiende indefinidamente hacia abajo en un árbol de herencia, en tanto que se declare que las clases derivadas tengan clases de base public. Por ejemplo el código siguiente es aceptable.

```
void funB()
{
        valor_A = 0;
}

void funC()
{
        valor_A = 1000;
}
```

1.5 APUNTADORES COMO MIEMBROS DE DATOS.

Los miembros de datos pueden ser también apuntadores. Los apuntadores no se pueden inicializar dentro de la declaración de una clase. Si un miembro de datos apunta a un tipo de clase, el apuntador no se iniciliza, ni las construcciones son llamadas de manera automática. Ejemplo.

```
class Segunda{

        int id;

public:

        Primera* Object;
        Segunda();
        Int getnom() { return id; }

};
```

Aquí, el miembro se declara para apuntar a un elemento de otra clase llamada primera. En la construcción de la clase segunda, el espacio de almacenamiento se asigna al objeto apuntador, pero el apuntador se deja sin inicializar.

1.5.1 APUNTADORES A MIEMBROS DE DATOS DE CLASES.

Las clases no son objetos, pero aveces puede utilizarlas como si lo fueran. Un ejemplo es la declaración de un apuntador a un miembro de clase.

```
class Ejemplo{

public:

        int valor;
        int identificador;

};

void SetValue(Ejemplo& Objeto)
{

        int Ejemplo::*ip = & Ejemplo::valor;
        Objeto.*ip = 3;

}

void main()
{

        Ejemplo Objeto1;
        Ejemplo Objeto2;

        SetValue(Objeto1);
        SetValue(Objeto2);

}
```

La función SetValue() tiene la declaración inusual:

```
        Int Ejemplo::*ip = & Ejemplo::valor;
```

Esta instrucción declara la variable ip que apunta a un valor de miembro de datos int en un objeto de clase Ejemplo sin indicar un objeto especifico.

1.6 CONSTRUCTORES.

Un constructor es una función especial que es miembro de esa clase y que tiene el mismo nombre de la clase.

Es muy frecuente que una cierta parte de un objeto necesite una iniciación antes de que pueda ser utilizada; como el requisito de iniciación es tan frecuente C++ permite que los objetos se den a sí mismos valores iniciales cuando se crean. Esta iniciación automáticamente se lleva a cabo mediante el uso de una función de construcción o constructor.
Por ejemplo este es el aspecto que tiene la clase cola cuando se modifica para utilizar las iniciaciones:

```
# include <iostream.h>

class cola{
        int c[100];
        int posfin, posprin;

public:
        cola(void);             // este es el constructor de la clase cola
        void ponent(int i);
        int quitaent(void);

};
```

Obsérvese que no se especifica un tipo de dato proporcionado para el constructor cola(). En C++, los constructores pueden proporcionar valores.
La función cola() se codifica de la siguiente manera:

```
cola::cola(void)
{
        posfin=posprin=0;
        cout<<"la cola ya tiene valores iniciales \n";
}
```

La función de construcción de un objeto se invoca cuando se crea el objeto. Esto significa que se invoca cuando se ejecuta la declaración del objeto. Además, para los objetos locales, el constructor se invoca cada vez que se llega a la declaración del objeto.
Como lo dice el nombre, un constructor es una función que se utiliza para construir un objeto de una clase dada; esto puede implicar la presencia de diferentes escenarios.

1.- Creación de objetos con iniciación definida.
2.- Creación de objetos con iniciación especifica.
3.- Creación de objetos copiando otro objeto.

Cada uno de estos procesos implica un tipo diferente de constructor. Un constructor tiene el nombre de la clase a la que pertenece.

Un sub objeto es un objeto de clase que se declara dentro de otra clase. Cuando se tiene una instancia en una clase, su constructor debe crear un objeto de esa clase. Si la clase tiene sub objetos declarados en ella, el constructor tiene que invocar los constructores de estos objetos. Considere el ejemplo siguiente.

```
class counter{
        int value;
public:
        Counter() { value = 0; }
};

class Example{
        int value;
public:
        Counter cars;
        Example() { value = 0; }
};

void main()
{
        Example e;
}
```

Cuando se crea el objeto e en main(), se llama al constructor de la clase Example; en este caso, la función Example::Example() antes de ejecutar su cuerpo, invoca al constructor Counter::Counter() del sub objeto cars. Cuando se completa este constructor, se ejecuta el cuerpo de Example::Example().

1.6.1 CONSTRUCTORES PRIVATE.

Obsérvese que el constructor anterior aparece en la sección public de la clase; este no es un requisito, pero normalmente es el caso.

Un constructor private, impediría que los usuarios genéricos crearan objetos a partir de esa clase y forzarán el cumplimiento de una de las condiciones siguientes antes de que se pueda crear un objeto.

1.- Un miembro estático de la clase invoca al constructor.
2.- Una clase friend de esa clase invoca al constructor.
3.- Un objeto existente de la clase tiene una función miembro que crea nuevos objetos invocando al constructor.

1.6.2 CONSTRUCTORES CON ARGUMENTOS.

La función básica de un constructor consiste en inicializar un objeto antes de usarlo.

```
Counter(long);

Counter::Counter(long value)
{
        count = value;
}

void main()
{
        Counter object(5);
}
```

Observe los paréntesis después del nombre de la variable, que hacen que la definición del objeto se asemeje a una llamada a función. La definición del objeto es en realidad una llamada a función con argumentos.
Suponga que desea crear una clase counter que sea lo suficientemente flexible para aceptar cualquier tipo de inicialización, utilizando elementos float, long, int, cadena o incluso ningún argumento. Estas son las construcciones que se deben declarar.

```
class Counter{

public:
        Counter(int = 0);
        Counter(long);
        Counter(double);
        Counter(char *);
};

//          declaración de constructores.

Counter::Counter(long val_inic)
{
        count = val_inic;
}

Counter::Counter(double val_inic)
{
        count = val_inic;
}

Counter::Counter(char* val_inic)
{       count = atol(val_inic);     }
```

```
//        uso de los constructores.

void main()
{

        Counter Object("5");        //  Utilizando constructor char*
        Counter Object1(5);         //  Utilizando constructor int
        Counter Object2(5L);        //  Utilizando constructor long
        Counter Object3(5.0);       //  Utilizando constructor double
        Counter Object4();          //  Utilizando constructor por omisión

}
```

El compilador puede determinar automáticamente a que constructor llamar en cada caso examinando los argumentos.

1.6.3 CONSTRUCTORES PARA COPIAR OBJETOS.

Cuando se crea un objeto, a menudo no se desea inicializar ningún valor de manera específica; simplemente se desea que un objeto "sea como otro". Esto implica hacer una copia de un objeto preexistente, lo cual requiere un tipo especial de construcción, llamada en general: constructor de copia. Ejemplo.

```
class Counter{
        .
        .
        .
public:

        Counter(Counter&);
        .
        .
        .
};



Counter::Counter(Counter &referencia)
{

        count = referencia.count;

}
```

```
void main()
{

        Counter Object(5);              //  Constructor entero
        Counter Object1 = Object;       //  Constructor de copia

}
```

1.7 DESTRUCTORES.

Los destructores entran en la misma categoría que los constructores. Se utilizan para realizar ciertas operaciones que son necesarias cuando ya no se utiliza un objeto como es la liberación de memoria.

Existen algunas diferencias importantes entre los constructores y los destructores:

1.- Los destructores pueden ser virtuales, los constructores NO.
2.- A los destructores no se les puede mandar argumentos.
3.- Sólo se puede declarar un destructor para una clase dada.

El destructor se nombra como la clase pero este va precedido de un tilde (~).

Se podría escribir una clase que se encargue de manejar todas las gráficas generadas por un programa de la siguiente manera:

```
class Graphics{

public:
        Graphics();
        ~Graphics();
        void DrawCircle(int x, int y, int radio);
        void DrawDot(int x, int y);
                .
                .
};
```

El destructor se utiliza para cerrar el dispositivo gráfico y rechazar cualquier espacio de memoria asignado al objeto.

Por ejemplo vea la clase cola con su constructor y destructor (en el ejemplo de la clase cola no es necesario un destructor, pero en este caso se pone para ejemplificar su uso).

```
# include <iostream.h>

class cola{
        int c[100];
        int posfin, posprin;

public:
        cola(void);             // este es el constructor de la clase cola
        ~cola(void);            // este es el destructor de la clase cola
        void ponent(int i);
        int quitaent(void);

};
//   Función de Construcción
cola::cola(void)
{
        posfin=0;
        posprin=0;
}


// Función de destrucción.
cola::~cola(void)
{
        cout<<"la cola ha sido destruida \n";
}
```

Veamos como funcionan los constructores y destructores en la nueva versión del programa que crea una cola.

```
# include <iostream.h>

class cola{
        int c[100];
        int posfin, posprin;
public:
        cola(void);             // este es el constructor de la clase cola
        ~cola(void);            // este es el destructor de la clase cola
        void ponent(int i);
        int quitaent(void);
};
```

```
main(void)
{
        cola a, b;
        a.ponent(15);
        b.ponent(39);
        a.ponent(55);
        b.ponent(19);
        cout<<a.quitaent() << " ";
        cout<<b.quitaent() << " ";
        cout<<a.quitaent() << " ";
        cout<<b.quitaent() << " ";
        return 0;
}


//   Función de Construcción

cola::cola(void)
{
        posfin=0;
        posprin=0;
        cout<<"La cola ya tiene valores iniciales \n"
}


// Función de destrucción.

cola::~cola(void)
{
        cout<<"La cola ha sido destruida \n";
}


void cola::ponent(int i)
{
        if(posfin>=100)
          {
              cout<<"la cola esta llena ";
        return;
           }
        posfin++;
        c[posfin] = i;

}
```

```
int cola::quitaent(void)
{
        if(posfin==posprin)
        {
          cout<<"la cola está vacía";
        return 0;
        }

        posprin++;
        return c[posprin];
}
```

Este programa da como resultado lo siguiente:

```
La cola ya tiene valores iniciales.
La cola ya tiene valores iniciales.
15      39
55      19
La cola ha sido destruida.
La cola ha sido destruida.
```

1.8 CLASES AMIGAS (Palabra reservada *friend*)

A veces se necesitan 2 clases que son tan conceptualmente cercanas que usted desearía que una de ellas tuviera acceso irrestricto a los miembros de la otra. Considere la implantación de una lista asociada: necesita una clase que represente nodos individuales, y una que se encargue de la lista misma. El acceso a los miembros de la lista es a través del manejador de la misma, pero el manejador debe tener acceso absoluto a los miembros de la clase o una función. Una clase que no ha sido declarada aún se puede definir como friend de esta manera:

```
class Node{
        friend class ObjectList;
        int value;
        Node* Predecesor;
        Node* Sucesor;

public:
        void value(int i){ value = i; }
```

```
};
class ObjectList{

        Node* head;
        Node* tail;
        Node* current;
public:
        void InsertNode(Node * ) {}
        void DeleteNode(Node *) {}
        int CurrentObject(Node* node) { return node->value;}
};
```

1.8.1 FUNCIONES AMIGAS.

Es posible que una función de una clase que no sea un miembro tenga acceso a las partes privadas de esa clase, declarando que se trata de un *friend* (amigo) de esa clase. Por ejemplo amg() se declara como *friend* de la class C1

```
Class C1{
        .
        .
        .
public:
        friend void amg(void);
        .
        .
        .
};
```

Como se puede ver, la palabra reservada *friend* precede a toda la declaración de la función, que es lo que se hace en general.

La razón por la cual se permite en C++ las funciones *friend* es la de resolver situaciones en las cuales dos clases deban compartir una misma función, para así aumentar la eficiencia. Para ver un ejemplo, consideremos un programa que defina dos clases llamadas linea y recuadro. La clase linea contiene todos los datos y código necesarios para dibujar una línea horizontal discontinua de cualquier longitud, empezando en la coordenada X, Y que se indique y utilizando un color especificado, La clase recuadro contiene todo el código y los datos necesarios para dibujar un recuadro en las coordenadas especificadas para la esquina superior izquierda y para la esquina inferior derecha, y con el color que se indique. Las dos clases tienen la misma función mismo_color() para determinar si una línea y un recuadro están pintados del mismo color. Las clases se declaran según se muestra a continuación:

```
class linea;

class recuadro{
        int color;
        int xsup, ysup;
        int xinf, yinf;
public:
        friend int mismo_color(linea l, recuadro b);
        void pon_color(int c);
        void definir_recuadro(int x1, int y1, int x2, int y2);
        void mostrar_recuadro(void);
};

class linea{
        int color;
        int xinicial, yinicial;
        int longitud;
public:
        friend int mismo_color(linea l, recuadro b);
        void pon_color(int c);
        void definir_linea(int x, int y, int l);
        void mostrar_linea();
};
```

La función mismo_color(), que no es miembro de ninguna de ellas pero es friend de ambas, proporciona un valor verdadero si tanto el objeto línea como el objeto recuadro, que son sus argumentos, se dibujan del mismo color; en caso contrario, proporciona un valor nulo. La función mismo_color se muestra a continuación:

```
Int mismo_color(linea l, recuadro b)
{
        if(l.color == b.color) return 1;
        return 0;
}
```

También puede declarar una función no miembro como *friend* antes que el identificador de la función esté en el campo de acción. Por ejemplo.

```
class Node{

        friend int GetObject(Node*);
        int value;
        Node* Predecesor;
        Node* Sucesor;
```

```
public:
        void value(int i){ value = i; }
};

int GetObject(Node* n)
{
        return n->value;
}
```

1.8.2 PROPIEDADES DE LA PALABRA RESERVADA **friend.**

Las funciones y clases declaradas *friend* para otras clases gozan de privilegios especiales. Si la función FUN0() es un elemento *friend* de la clase B y la clase B se deriva de la clase A. FUN0(), tiene acceso también a los miembros de datos de las clases A, B, y C:

```
class A{
        friend class FRIEND;
        int a1;
protected:
        int a2;
public:
        int a3;
};


class B{
        int b1;
protected:
        friend class FRIEND;
        int b2;
public:
        int a3;
};
```

La clase friend puede tener acceso a todos los miembros de datos de C.

```
class C{
        int c1;
protected:
        int c2;
public:
        friend class FRIEND;
        int c3;
};
```

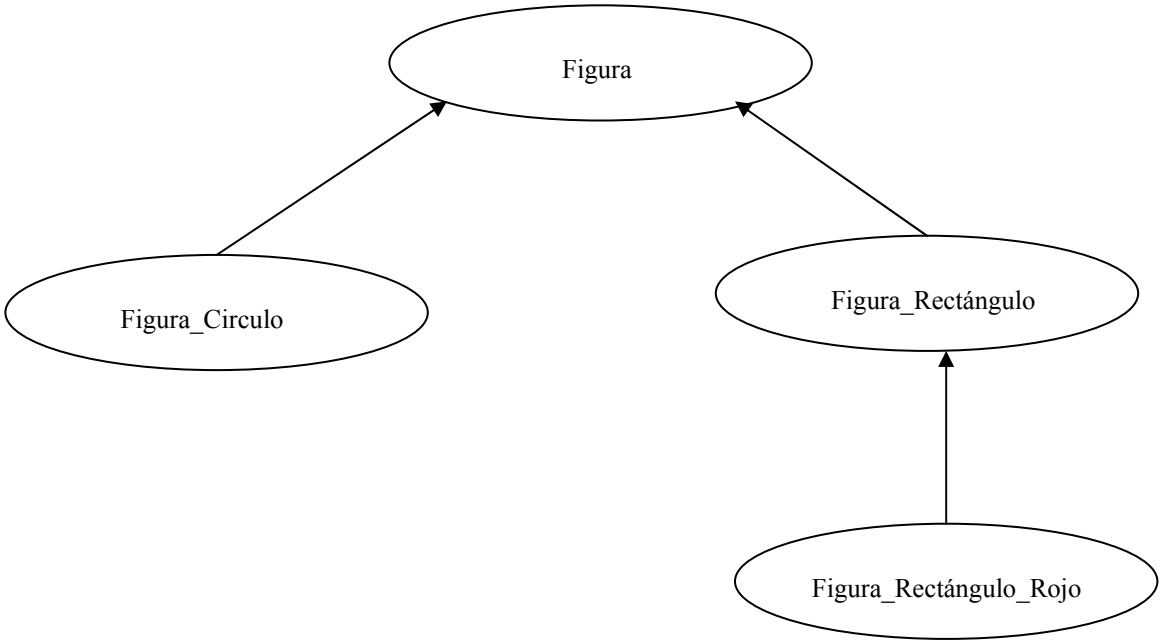
1.9 HERENCIA

La herencia es uno de los rasgos fundamentales de un lenguaje de programación orientado a objetos. En C++, la herencia se basa en permitir que una clase contenga a otra clase en su declaración; supongamos una clase Figura:

```
Class Figura{
        .
        .
public:
        .
        .
};
```

Una clase derivada Figura_Circulo se declara así.

```
Class Figura_Circulo:public Figura
{
public:
        .
private:
double x_centro, y_centr;
double radio;
};
```



La declaración general de la herencia es la que se muestra a continuación:

```
Class Nombre_de_la_clase_nueva : acceso clase_heredada{

        .
        .
        .
};
```

Aquí *acceso* es opcional, sin embargo, si está presente tiene que ser *public*, private o protected.
El uso de *public* significa que todos los elementos *public* del antecesor también serán *public* para la clase que lo hereda.

El siguiente ejemplo muestra 2 clases donde la segunda de ellas hereda las propiedades de la primera.

```
class Box{

public:

        int width, height;
        void SetWidth(int w) { width = w; }
        void SetHeight(int h) { height = h; }

};


class ColoredBox:public Box{

public:

        int color;
        void Setcolor(int c) { color = c; }

};
```

La clase Box recibe el nombre de clase base de la clase ColoredBox. Que a su vez recibe el nombre de clase derivada. La clase Colored Box se declara solo con una función, pero también hereda 2 funciones y 2 variables de su clase base. Así, se puede crear el código siguiente:

```
ColoredBox  Cb;                         // se crea una instancia de ColoredBox
void main()
{

        Cb.Setcolor(5);         //  función miembro de ColoredBox.
        Cb.SetWidth(30);        //  función heredada.
        Cb.setHeight(50);       //  función heredada.

}
```

Observe como las funciones heredadas se utilizan exactamente como si fueran miembro.

1.9.1 LIMITACIONES DE LA HERENCIA.

Cómo y cuándo se deriva una clase de otra es puramente decisión del programador. Esto puede parecer obvio, pero es una limitación. El diseñador de un programa debe decidir al momento de la compilación quien hereda qué, de quién, cómo y cuándo se lleva a cabo la herencia.

1.9.2 QUE NO SE PUEDE HEREDAR.

Tal y como en la vida real, en C++ no todo se puede transmitir a través de la herencia. Esto se puede considerar en un principio como una desventaja o limitación, pero en realidad solo algunos casos especiales inconsistentes por definición con la herencia:

1.- Constructores.
2.- Destructores.
3.- Nuevos operadores definidos por el usuario.
4.- Relaciones *friend*.

El constructor de una clase de base no puede ser invocado de manera explícita en una clase derivada como otras funciones heredadas. Considere el código siguiente:

```
class Parent{
        int value;

public:
```

Continua…

```
        Parent(){ value = 0; }
        Parent(int v){ value = 0; }

};
class Child:public Parent{
        int total;
public:
        Child(int t)  { total = t; }
        void  SetTotal(int t);
};

void Child::SetTotal(int t)
{
        Parent::Parent(i);              // Esto no se puede hacer, ya que el constructor de la
                                        // clase no es heredado como otras funciones.
        Total = t;
}
```

De manera análoga, los destructores están diseñados para ser invocados automáticamente cuando un objeto sale del campo de acción.

La relación *friend* no es heredada. Esto es similar a la vida real; los amigos de sus padres no son automáticamente amigos suyos.

1.9.3 HERENCIA MULTIPLE.

Una clase puede heredar los atributos de dos o más clases. Para lograr esto, se utiliza una lista de herencia separada mediante comas en la lista de clases base de la clase derivada. La forma General es:

```
Class Nombre_clase_derivada : lista de clases base  {
        .
        .
        .
};
```

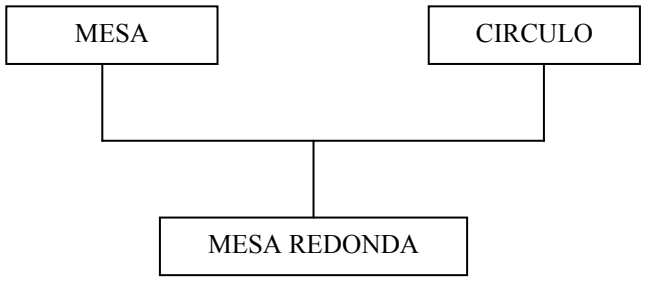
Por ejemplo en este programa Z hereda tanto a X como a Y.

```
# include <iostream.h>

class X{

protected:
```

```
        int a;

public:
        void hacer_a(int i);
};

class Y{

protected:
        int b;
public:
        void hacer_b(int i);
};

// Z hereda tanto a X como a Y

class Z : public X, public Y {

public:
        hacer_ab(void);
};


void X::hacer_a(int i)
{
        a = i;
}

void Y::hacer_b(int i)
{
        b = i;
}

int Z::hacer_ab(void)
{
        return a*b;
}


main(void)
{
        Z var;
        var.hacer_a(10);
        var.hacer_b(25);
        cout << var.hacer_ab();
        return 0;
```

}

En este ejemplo, Z tiene acceso a las partes public y protected tanto de X como de Y. Remarcando, una clase puede tener muchos padres y heredar propiedades de cada una de sus clases base. Considere crear una clase MesaRedonda, que no solo tenga las propiedades de las mesas, sino también la característica geométrica de ser redonda.



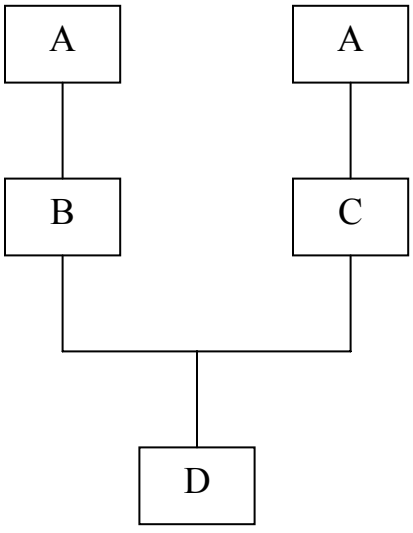

```
#include <stdio.h>

class Circle{

        float radio;
public:
        Circle(float r){ radio = r; }
        Float Area(){ return radio*radio*3.1416; }
};

class Mesa{
        float height;
public:
        Mesa(float h) { height = h; }
        float  Height() { return height; }
};

class MesaRedonda:public Mesa, public Circle{
        int color;
public:
        MesaRedonda(float h, float r, int c);
        int Color() { return color; }
};

MesaRedonda::MesaRedonda(float h, float r, int c): Circle(r),Mesa(h)
{
        color = c;
}
```

```
void main()
{
        MesaRedonda Mesa1(15.0, 3.0, 5);
        printf("\n Las propiedades de la Mesa son:");
        printf("\n Altura = %f ", Mesa.Height());
        printf("\n Area = %f ", Mesa.Area());
        printf("\n Color = %d ", Mesa.Color());

}
```

La función main() invoca las tres funciones miembro MesaRedonda::Height(), MesaRedonda::Area() y MesaRedonda::Color(). Todo sin indicar cuales son funciones heredadas y cuales no.

1.9.4 USO DE CLASES DE BASE VIRTUAL.

Las clases de base virtual se utilizan sólo en el contexto de la herencia múltiple. Dada la complejidad de relaciones que pueden surgir en un árbol de herencia construido en torno a la herencia múltiple, existen situaciones en las que el programador necesita tener cierto nivel de control sobre la forma en que se heredan las clases de base. Considere el árbol de herencia de la siguiente figura.



La clase D tiene a A como clase de base. El problema es que hay dos clases A diferentes que aparecen como clases de base de D, cada una con datos propios.

Esto se ejemplifica con el siguiente código.

```
class A{
public:
        int value;
};

class B : public A{};
class C : public A{};
class D : public B, public C {
public:
        int value() {return value; }
};
```

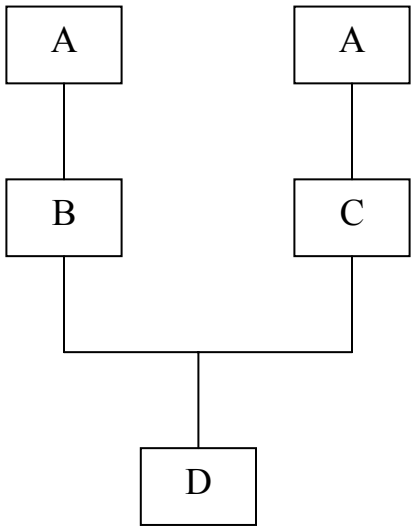
El valor miembro de acceso a la instrucción en D es ambiguo. Borland C++ genera error.

Tener múltiples copias de la misma clase de base en árbol de herencia no sólo es confuso, sino que puede ser un desperdicio de espacio de almacenamiento.

Declarar una base *virtual* resuelve el problema. Obliga al compilador a admitir sólo una copia de la clase de base dada en la declaración de una clase derivada. Por lo tanto el ejemplo anterior lo podemos corregir de la siguiente manera:

```
class B : public virtual A{};
class C : public virtual A{};
class D : public B, public C {
public:
        int value() {return value; }
};
```

y el árbol de herencia quedaría como realmente lo deseamos.



1.10 SOBRECARGA.

Una de las maneras que tiene el C++ de llegar al polimorfismo es a través de la sobrecarga de funciones. En C++ , dos o mas funciones pueden compartir un nombre, siempre y cuando en la declaración, sus parámetros sean diferentes.

Sobrecarga se refiere a la práctica de cargar una función con más de un significado. Básicamente, el término expresa que se cargan uno o más identificadores de función sobre un identificador previo.

1.10.1 PORQUE USAR LA SOBRECARGA.

La sobrecarga no es un concepto nuevo en los lenguajes de programación, por ejemplo el operador = está sobrecargado en muchos lenguajes de alto nivel y se utilizan en instrucciones de asignación y en expresiones condicionales como:

```
a = b;

if( a = b)
```

La sobrecarga otorga flexibilidad, permite a las personas utilizar código con menos esfuerzo, ya que extiende operaciones que son conceptualmente similares en naturaleza.

1.10.2 SOBRECARGA DE FUNCIONES.

Las funciones sobrecargadas se distinguen por el número y tipo de sus argumentos.

El tipo de retorno no se utiliza para distinguir funciones sobrecargadas, por lo tanto las funciones:

```
void Muestra(int q);
long Muestra(int q);
```

No son distinguibles y producen un error del compilador.

Cualquier nombre de función puede ser sobrecargada en C++, pero la sobrecarga solo funciona dentro de un campo de acción dado.
Cuando se declara en una clase más de una función miembro con el mismo nombre, se dice que el nombre de la función está sobrecargado en esa clase, y su campo de acción será en el ámbito de esa clase.

```
class Ejemplo{

        int value;

public:

        void value(int v) { value = v; }
        int value() { return value; }

};

void main()
{

        Ejemplo Ee;
        Ee.value(3);
        Int i = Ee.value();

}
```

Este código demuestra que la clase Ejemplo tiene 2 funciones sobrecargadas: una función para escribir y una para leer una variable.

Las funciones sobrecargadas necesitan diferir en una u otra o en las dos formas siguientes:

1.- Las funciones deben contener un número de argumentos diferente.
2.- Cuando menos uno de los argumentos debe ser diferente.

Considere el siguiente programa en el cual la función al_cuadrado se sobrecarga 3 veces.

```
# include<iostream.h>

int al_cuadrado(int i);
double al_cuadrado(double d);
long al_cuadrado(long l);

main(void)
{
        cout <<  al_cuadrado(10)    <<" \n";
        cout <<  al_cuadrado(1.25)  <<" \n";
        cout <<  al_cuadrado(9L)    <<" \n";

        return 0;
}

int al_cuadrado(int i)
{
cout<<" función al_cuadrado con parámetro entero"

        return i*i;
}

double al_cuadrado(double d)
{
cout<<" función al_cuadrado con parámetro doble"

        return d*d;
}


long al_cuadrado(long l)
{
cout<<" función al_cuadrado con parámetro largo"

        return l*l;
}
```

La ventaja de sobrecargar las funciones es que permite acceder a conjuntos de funciones que están relacionadas utilizando un solo nombre. En el programa anterior se crean 3 funciones similares que se llaman, al_cuadrado(), y cada una de las cuales regresa el cuadrado de su argumento; en cierto sentido, la sobrecarga de funciones permite crear un nombre genérico para alguna operación, y el compilador resuelve que función es la adecuada para llevar a cabo la operación.

Así, al_cuadrado(), representa la acción general que se realiza; el programador solo necesita recordar la acción general que se lleva a cabo, por lo tanto al aplicar el polimorfismo se han reducido a una las 3 cosas que había que recordar. Aunque este ejemplo es bastante trivial,

si expande el concepto se puede ver que el polimorfismo puede ayudarnos a entender programas muy complejos.

Para sobrecargar la función de construcción de una clase, solo hay que declarar las diferentes formas que tiene que adoptar y hay que definir su acción con respecto a esas formas. Por ejemplo el programa siguiente declara una clase llamada temporizador, que se comporta como un temporizador descendente. Cuando se crea un objeto del tipo temporizador, se le da un valor inicial de la hora. Cuando se invoca a la función ejecutar(), el temporizador cuenta hasta llegar a cero, y hace sonar el timbre. En ese ejemplo, se ha sobrecargado el constructor para especificar la hora como un entero, como una cadena, o como dos enteros que especifican los minutos y los segundos.

```
# include <iostream.h>
# include <stdlib.h>
# include <time.h>

class temporizador{
        int segundos;
public:
                        //  Se especifican los segundos como una cadena.
        temporizador(char *t) {segundos = atoi(t);}
                        //  Se especifican los segundos como un entero
        temporizador(int t) {segundos = t;}
                        //  Se especifica la hora en minutos y segundos
        temporizador(int min, int seg) {segundos =  min* 60 + seg; }
        void ejecutar(void);
};


void temporizador::ejecutar(void)
{
        clock_t t1, t2;
        t1 = t2 = clock()/CLK_TCK;
        while(segundos)  {
                if(t1/CLK_TCK+1 <= (t2=clock()) /CLK_TCK){
                  segundos --;
                  t1 = t2;
                }
        }
        cout << "\a";   // toca el timbre
}


main(void)
```

```
{
        temporizador a(10), b("20"), c(1, 10);
        a.ejecutar();    // cuenta 10 segundos
        b.ejecutar();    // cuenta 20 segundos
        c.ejecutar();    //cuenta 1 minuto, 10 segundos

        return 0;
}
```

Como se puede ver, cuando a, b, y c se crean dentro de main(), se les dan valores iniciales utilizando los tres métodos diferentes que admiten las funciones de construcción sobrecargadas.

1.10.3 CONSTRUCTORES SOBRECARGADOS.

Uno de los usos más comunes de la sobrecarga de funciones es con los constructores. La razón es que cuando se instancía una clase, se deben conservar las cosas lo más flexibles que sea posible; de modo que los usuarios pueden realizar diferentes clases de instancias.

Considere una clase ventana desplegable en una interfaz gráfica de usuario.

```
PopupWindow Window;                 // Genera una ventana con parámetros por
                                    // omisión.
PopupWindow Window_1(x, y);         // Genera una ventana con coordenadas
                                    //  específicas.
PopupWindow Window_2(x, y, width, Height);      // Genera una ventana con
                                                // dimensiones controladas.
PopupWindow Window_3 = Window_2;    // Genera una ventana igual a la
                                    // anterior.
```

La implantación de esta clase podría parecerse al siguiente código.

```
class PopupWindow{

        Int x, y, Width, Height;
public:
        PopupWindow();
        PopupWindow(int, int);
        PopupWindow(int, int, int, int);
        PopupWindow(PopupWindow&);

};
```

```
PopupWindow:: PopupWindow()
{
        x = y = 100;
        Widht = Heigth = 100;
}


PopupWindow:: PopupWindow(int px, int py)
{
        x = px;
        y = py;
        Widht = Heigth = 100;
}

PopupWindow:: PopupWindow(int px,int py, int w, ,int h)
{
        x = px;
        y = py;
        Widht = w;
        Heigth = h;
}


PopupWindow:: PopupWindow(PopupWindow& pw)
{
        x = pw.x;
        y = pw.y;
        Widht = pw.Widht;
        Heigth = pw.Height;
}
```

La clase utiliza cuatro funciones sobrecargadas que realizan el trabajo.

Con constructores sobrecargados, usted puede permitir que el usuario especifique qué variables han de ser inicializadas de manera explícita y cuales deben asumir valores definidos.

1.10.4 SOBRECARGA DE OPERADORES.

Otra forma en que se logra el polimorfismo en C++ es mediante la sobrecarga de operadores. En general se puede sobrecargar cualquiera de los operadores de C++ definiendo lo que significa con respecto a una cierta clase.

Aunque los operadores están asociados comúnmente con operaciones matemáticas o lógicas, simplemente son una notación alternativa para una llamada a una función

La sobrecarga de operadores se utiliza en otros lenguajes de programación, pero sin un nombre especial. En algunos lenguajes como Pascal, es posible hacer lo siguiente:

```
StructureA := structureB + structureC;
```

Lo que da lugar a una adición byte por byte de las estructuras b y c que se copiarán en la estructura a. Esta sintaxis implica que el operador de adición está sobrecargado para estructuras, aunque con ciertas reglas de apego a tipos. En el ejemplo anterior se utiliza también un operador de asignación sobrecargado, ya que la instrucción y no una asignación escalar definida, fue la que activó una operación de copia de estructuras.

1.10.4.1 OPERADORES COMO LLAMADAS A FUNCION.

Hay dos formas en las que se pueden implantar operadores para objetos de clase: como funciones miembro y como amigos. Un operador unario aplicado a un objeto es equivalen te a una llamada a una función. Dados un objeto W y un operador unario @ , la expresión @W es equivalente a las llamadas a funciones:

```
W.operator@()          //  uso de un operador con función miembro.
operator@(W)           //  uso de un operador con función friend.
```

Un operador binario aplicado a los objetos es equivalente también a una llamada a función. Dados los objetos X y Y, y un operador @, la expresión X @ Y es equivalente a las llamadas a funciónes:

```
X.opertor@(Y)          // uso de un operador con función miembro.
Operator@(X, Y)        // uso de un operador con función friend.
```

El código anterior demuestra que los operadores pueden invocar dos funciones diferentes; una función que es miembro y otra que es amigo.

1.10.4.2 OPERADORES SOBRECARGADOS COMO FUNCIONES MIEMBRO.

Las funciones que implantan operadores son un tanto inusuales. Para comenzar, sus nombres deben comenzar con la cadena *operator*, seguida de los caracteres que representan el operador que se implanta. Por ejemplo la función miembro para implantar el operador de

adición tendría que llamarse *operator+* . La segunda restricción se aplica al número de argumentos que pueden tomar estas funciones. Las funciones miembro que implantan operadores unarios no deben tomar argumentos, en tanto que las que implantan operadores binarios pueden tomar solo un argumento.

El siguiente código muestra operadores sobrecargados implantados como funciones miembro.

```
class Counter{

public:
        int value;
        Counter(int i) { value = i; }
        Counter operator!();                    //      operador unario.
        Counter operator+(Counter & c);         //      operador binario.

};
Counter Counter::operator!()
{
        return Counter( ! value );
}


Counter Counter::operator+( Counter & c)
{
        return Counter( value + c.value);
}



void main()
{

        Counter c1(3), c2(5);           //      se crean 2 objetos tipo counter.
        c1 =  ! c1;                     //      se aplica el operador unario
        c1 = c1 + c2;                   //      se aplica el operador binario

}
```

El uso de los operadores en la función main() es completamente intuitivo, y no requiere que el usuario conozca los detalles de la implantación de clases para averiguar cual será el recultado de las operaciones.

La restricción en el número de argumentos (solamente uno), limita las posibilidades; pero esto lo podemos resolver sobrecargando el mismo operador tantas veces como sea necesario, vea un ejemplo de como se sobrecarga el operador de adición tres veces.

```
class M{

public:
        int value;
        M(int i) { value = i; }
        M operator+(M&  m);
        M operator+(int i);
        M operator+(double d);
};

M M::operator+(M&  m)
{
        return M(value + m.value);
}


M M::operator+(int i)
{
        return M(value + i);
}

M M::operator+(double d)
{
        returnM(value + d);
}


void main()
{
        M m1(3), m2(10);            //      se crean dos objetos de clase M.
        m1 = m1 + m2;               //      uso M::operator+(M&)
        m1 = m2 + 200;              //      uso M::operator+(int)
        m1 = m2 + 3.14159;          //      uso M::operator+(double)
}
```

1.10.4.3 OPERADORES SOBRECARGADOS COMO FUNCIONES FRIEND.

En el siguiente ejemplo el operador + se declara como función friend y el operador = como función miembro de la clase X, ya que el operador = solo puede sobrecargarse como función miembro, lo que implica que el operador = global no puede ser sobrecargado. Los diseñadores del lenguaje decidieron que permitir que las funciones friend cambiaran el significado del operador de asignación causaría mas problemas de los que resolvería.

```
class  X{
```

```
        friend X operator+(X&, X&);
public:
        int value;
        X(int i) { value = i; }
        X& operator = (X&);
};


X& X::operator = (X& b)
{
        value = b.value;
        return *this;
}


X operator+(X& a, X& b)
{
        return X(a.value + b.value);
}

void main()
{

        X g(2), h(5), i(3);
        G = h +h +i;

}
```

En general, los operadores friend sobrecargados se comportan de manera muy similar a las funciones miembro.

1.11 POLIMORFISMO

El origen del término polimorfismo es simple: proviene de las palabras griegas *poly* (muchos) y *morphos* (forma) multiforme. El polimorfismo describe la capacidad del código C++ de comportarse de diferentes maneras dependiendo de situaciones que se presenten al momento de la ejecución.

El concepto de polimorfismo es crucial para la programación orientada a objetos. En su concepción relativa a C++, el término polimorfismo se utiliza para describir el proceso mediante el cual se puede acceder a diferentes implementaciones de una función utilizando el mismo nombre. Por esta razón el polimorfismo se define a veces mediante la frase “una interface métodos múltiples”. Esto significa que en general se puede acceder a toda una

clase de operaciones de la misma manera, aunque las acciones concretas que estén asociadas a cada una de las operaciones pueda ser diferente.

En C++, el polimorfismo se admite tanto en el momento de la ejecución como en el momento de la compilación. La sobrecarga de operadores y de funciones es un ejemplo de polimorfismo en el momento de la compilación. Sin embargo, aunque la sobrecarga de operadores y de funciones son muy potentes, no pueden llevar a cabo todas las tareas que requiere un verdadero lenguaje orientado a objetos. Por tanto, C++ permite también el polimorfismo en el momento de la ejecución mediante el uso de clases derivadas y de funciones virtuales.

1.11.1 FUNCIONES VIRTUALES.

El polimorfismo en el momento de la ejecución se consigue mediante el uso de tipos derivados y funciones virtuales. En pocas palabras, una función virtual es una función que se declara como virtual en una clase base y que se define en una o más clases derivadas.

Lo que hace especiales a las funciones *virtual* es que cuando se accede a una de ellas utilizando un puntero de clase base señala a un objeto de clase derivada, C++ determina qué función debe llamar en el momento de la ejecución, basándose en el tipo del objeto al cual apunta. Por tanto, si apunta a diferentes objetos, se ejecutan versiones diferentes de la función *virtual*.

Como ejemplo examine el siguiente código:

```
#include<iostream.h>

class Base {
public:
        virtual void quien()  { cout << "Base \n"; }
};

class primera_deriv : public Base {
public:
        void quien() { cout << " Primera derivación \n"; }
};


class seguna_deriv : public Base {
public:
        void quien() { cout << " Segunda derivación \n"; }
};
```

```
main(void)
{
        Base obj_base;
        Base *p;
        Primera_deriv   obj_primera;
        Segunda_deriv  obj_segunda;
        p = &obj_base;
        p->quien();
        p = &obj_primera;
        p->quien();
        p = &obj_segunda;
        p->quien();
        return 0;
}
```

El programa produce la siguiente salida.

Base
Primera derivación
Segunda derivación.

La clave de la utilización de funciones virtual para lograr el polimorfismo en el momento de la ejecución es que se debe acceder a esas funciones mediante el uso de un puntero declarado como puntero de la clase base.

Parte de la clave para aplicar con éxito el polimorfismo consiste en comprender que la base y las clase derivadas forman una jerarquía, que va desde la mayor generalización a la menor. Por tanto la clase base cuando se utiliza correctamente, proporciona todos los elementos que puede utilizar directamente una clase derivada, mas aquellas funciones que la clase derivada debe implementar por sí misma. Sin embargo dado que la forma de la interface está determinada por la clase base todas las clases derivadas van a compartir esa interface .

El código siguiente utiliza la clase figura para derivar dos clase concretas llamadas cuadro y triángulo.

```
#include<iostream.h>

class Figura{
protected:
        double x, y:
```

```
public:
        void pon_dim(double I, double j) { x = I; y = j; }
        virtual void mostrar_area() { cout << "función no implementada \n";}
};


class triángulo : public Figura {
public:
        void mostrar_area() { cout<<"Triangulo de altura   " << x   <<" y base "
                                        << y  << " y área " x * y * 0.5; }
};

class cuadrado : public Figura {
public:
        void mostrar_area() { cout << "Cuadrado de lado " << x  <<" por " <<  y
                                        << "área  = " << x * y <<"\n"; }
};


main (void )
{
        Figura *p;
        triangulo trian;
        cuadrado cuad;
        p = &trian;
        p->pon_dim(10.0, 5.0);
        p->mostrar_area();
        p = &cuad;
        p->pon_dim(10.0, 5.0);
        p->mostrar_area();
        return 0;
}
```

Como se puede ver al examinar este programa, la interface de cuadrado y de triángulo es la misma, aunque cada uno de ellos proporcione sus propios métodos para calcular el área de cada uno de sus objetos.

Capítulo II

ESTRUCTURA DEL C++ BUILDER.

C++ tiene la reputación de ser un lenguaje muy poderoso, pero con el poder viene la responsabilidad, el programador debe de conocer una serie de conceptos y reglas de uso. Esto requiere de suficiente experiencia, un gran trato y conocimiento del lenguaje cuando está programando aplicaciones windows complejas. El hecho es: C++ es un lenguaje complejo con demasiadas reglas y terminos confusos, y el usuario tiene además que aprender técnicas de programación orientadas a objetos.

Productos como Borland Delphi y Microsoft Visual Básic, viene a provocar un cambio irreversible en la programación visual, ya que usando esas herramientas, los programadores pueden crear aplicaciones mucho mas fácil y rápido de lo que lo hacían anteriormente, ahora los desarrolladores solo tienen que mover los objetos que requieren usar, modificarlos de acuerdo a sus necesidades (solo llenando las funciones vacias), y es todo, una aplicación ejecutable es creada.

C++ Builder trabaja sobre la misma línea que Delphi y Visual Basic. Este desarrollo Rapido de aplicaciones hacen mas fácil la construcción de sus proyectos. Con estas herramientas de programación usted tiene mas poder con menos responsabilidad.

C++ Builder habilita y produce aplicaciones ejecutables bajo windows95 o windows NT con soporte de 32 bits; El código se ejecutará mucho más rápido.

Y puede ser que de otra manera C++ Builder lo haga un mejor programador.

2.1 EL C++ BUILDER.

Borland ofrece tres diferentes versiones de C++ Builder y cada versión será de acuerdo a sus necesidades.

C++ BUILDER STANDAR. Proporciona todas las herramientas que necesita para el desarrollo de aplicaciones de bases de datos, ésta versión ocupa aproximadamente 75 MB de espacio en su disco duro.

C++ BUILDER PROFESIONAL. Ofrece C++ Builder con el equipo de desarrollo y otras herramientas potentes como librerías y código fuente del que puede aprender sirviendo como tutorial. Esta versión proporciona algunas secciones especiales que probablemente no requiera inmediatamente, pero se alegrará al familiarizarse mas con C++ builder, esta versión ocupa cerca de 100 MB de espacio en su disco duro.

C++ BUILDER, SERIE CLIENTE/SERVIDOR. Estamos hablando de compatibilidad con bases de datos gigantes, esta versión está equipada con desarrolladores profesionales que necesitan enlazar con grandes corporaciones de Bases de Datos, tomando la arquitectura cliente servidor. Esta versión ocupa cuando menos 130 MB de espacio en su disco duro.

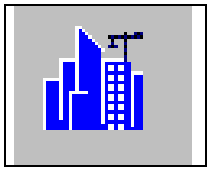
2.2 REQUERIMIENTOS DE HARDWARE.

Necesita una Pc con un procesador lo suficientemente poderoso y una amplia memoria. Mientras más poderosa sea su Pc, más productiva puede ser. C++ Builder es un producto bastante poderoso y como tal hace significantes demandas en su máquina.

Para una correcta ejecución de sus aplicaciones, se recomienda un procesador 486 y preferentemente uno basado en un procesador pentium, por lo menos 16 MB en RAM aunque Borland recomienda 24 MB en RAM. Ya que el desarrollo en C++ Builder corre en sistemas operativos de 32 bits, bajo Windows 95 o Windows NT; la capacidad del disco duro depende de la versión de C++ Builder que esté usando.

2.3 INSTALACION DE C++ BUILDER.

Para instalar C++ Builder en tu PC, coloca el CD ROM Borland C++ Builder en tu lector de discos compactos, no necesitas teclear nada, automáticamente el CD comienza la ejecución, iniciando por las preferencias de instalación del programa (Completa, Compacta, Personalizada) y su propia información para registrar el producto, si no se tiene suficiente espacio en disco duro, puedes instalar la versión mínima y tendrá que tener el CD en tu lector de disco compacto cada vez que quieras modificar una aplicación.



Al final de las rutinas de instalación, aparecerá un nuevo grupo de programas, podrás ejecutar la aplicación haciendo doble click en el icono de C++ Builder.

FIG 2.3.1 Icono de aplicación de C++ Builder.

2.4 AMBIENTE DE DESARROLLO INTEGRADO.

Cuando usted inicia C++ Builder, espera ver una solo ventana para desarrollar sus aplicaciones; pero C++ Builder le presenta un grupo de ventanas dispersas al rededor de su ventana principal. *Fig II.4.1* Al hacer doble click en el icono de C++ Builder.

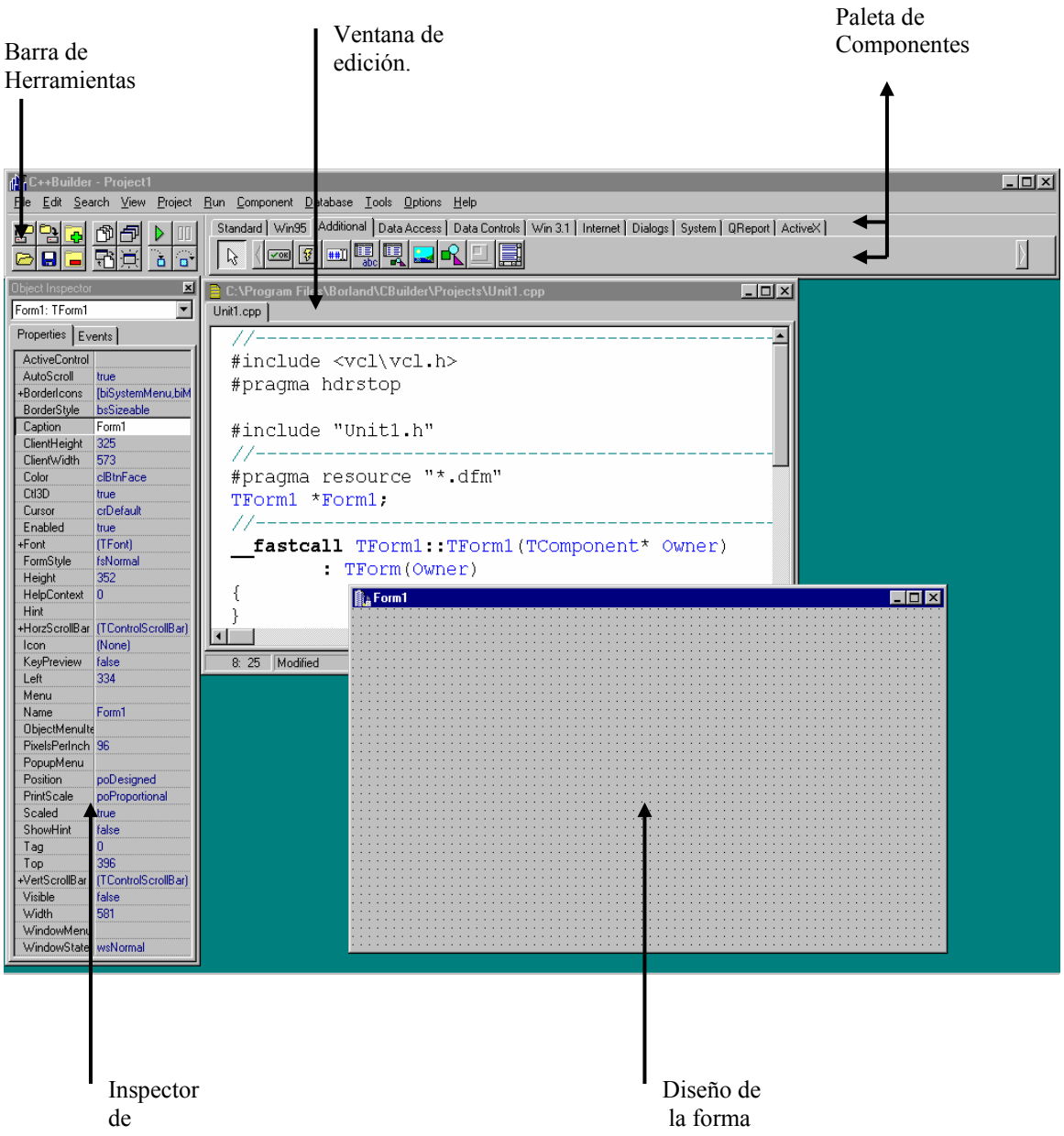


FIG. 2.4.1 Ambiente de desarrollo integrado C++ Builder.

En la figura se presenta los principales elementos de C++ Builder (Ambiente de Desarrollo Integrado, por sus siglas en Ingles IDE). Cada parte en el ambiente de desarrollo trabaja conjuntamente, diseños visuales y editor de código donde la edición es similar a otros editores; solo que con el ambiente de dasarrollo integrado, usted puede observar realmente lo que esta construyendo al momento de crearlo.

Idealmente se desearía trabajar con una resolución de 800 x 600 o tal vez mayor en su monitor, ya que estas resoluciones dan una sensación de amplitud; pero recuerde que al diseñar sus aplicaciones el usuario final puede trabajar con una resolución diferente a la suya, la manera mas sencilla de evitar problemas sería realizar sus aplicaciones para una de 480 x 640.

2.4.1 MENU PRINCIPAL Y BARRA DE HERRAMIENTAS.

Muchas de las opciones que puedes accesar desde el menú principal, están disponibles a través del panel de botones aceleradores en la barra de herramientas. En general la barra de herramientas provee de una manera rapida de ejecutar operaciones del programa con un simple click con su mouse.

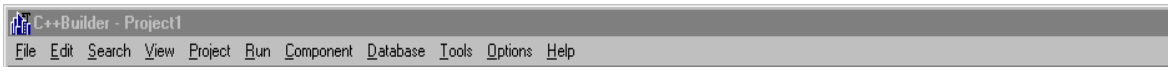


FIG. 2.4.1.1 Menú principal.

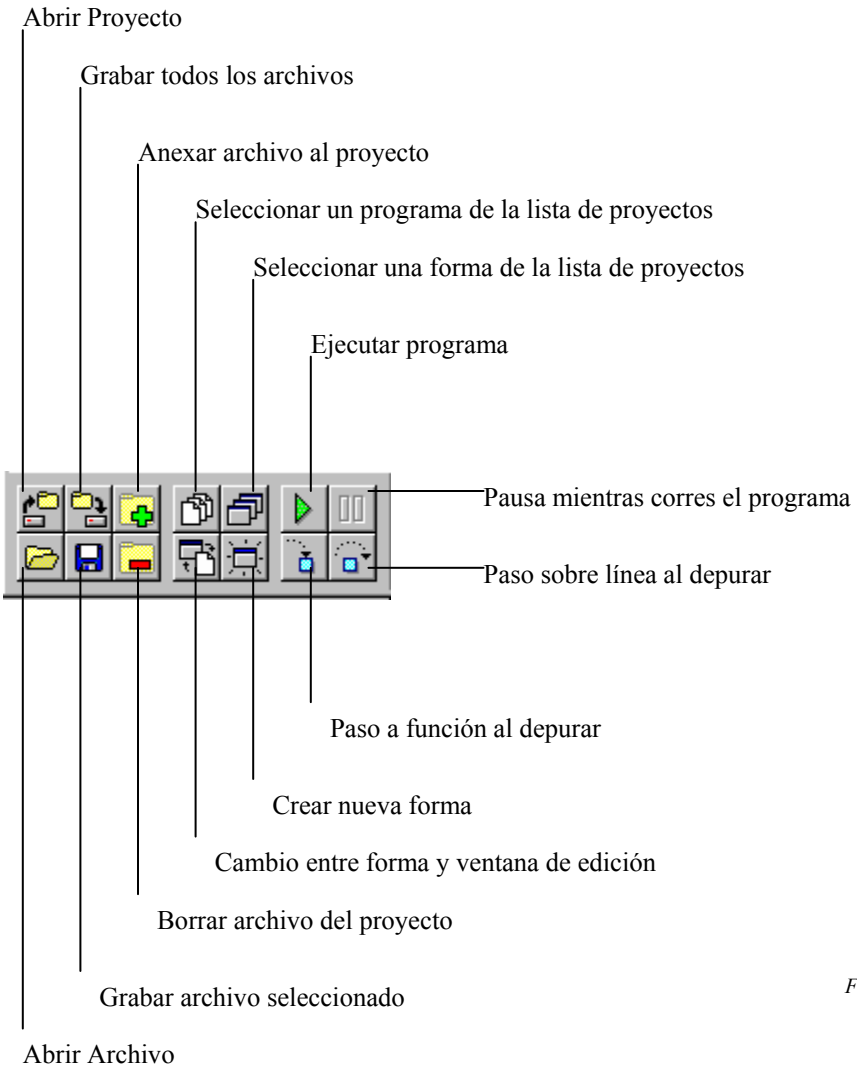


FIG. 2.4.1.2 Barra de herramientas.

Usted puede configurar la barra de herramientas, seleccionándola con un click, y nuevamente haciendo click con el botón derecho del mouse, se activará un menú, y al elegir la opción de propiedades, tendrá a su disposición el editor de la barra de herramientas donde podrá tomar las propiedades que necesite para anexarla a la barra de herramientas solamente jalándola con el mouse al panel de la barra de herramientas.

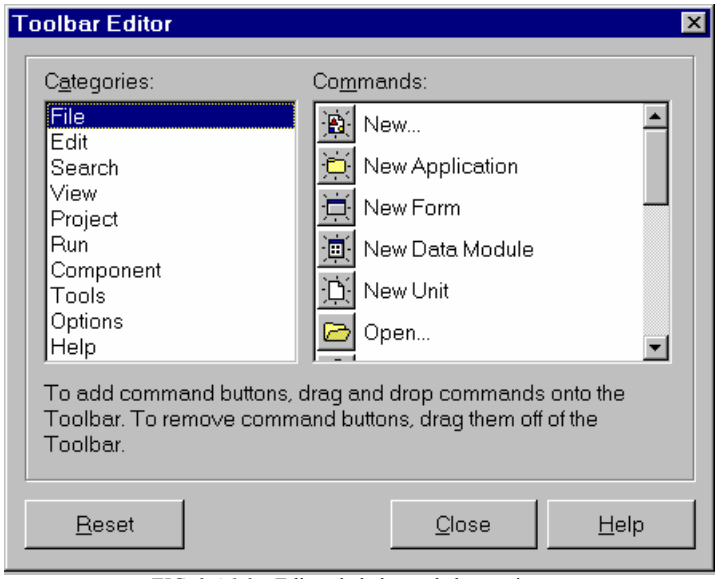


FIG. 2.4.1.3 Editor de la barra de herramientas.

Si lo que desea es remover alguna propiedad de la barra de herramientas, solamente jálelo con el mouse fuera del panel de la barra de herramientas y desaparecerá.

2.4.2 PALETA DE COMPONENTES.

La paleta de componentes es algo como un catálogo de objetos que puedes usar de acuerdo a las necesidades de construcción de tus aplicaciones. Está dividida en páginas o grupos de acuerdo a sus funciones. Para implantar uno de estos componentes en tu aplicación, solo tienes que seleccionarlo con el mouse haciendo un click en el objeto deseado y hacer click en la forma principal (Forma de edición, ventana punteada) para que ya puedas utilizar ese objeto. C++ Builder soporta docenas de componentes.

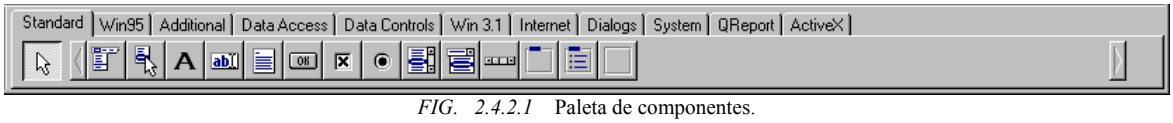


FIG. 2.4.2.1 Paleta de componentes.

Standard: Esta tabla contiene los objetos para hacer eficaces y elegantes tus aplicaciones Windows, incluye componentes para desplegar y editar texto, botones, barras de estado y menús.

FIG. 2.4.2.2 Paleta de componentes Standard.

Win95: Estos componentes permiten el acceso a los controles de usuario-interface de Windows95. Uno de los principales es la vista del árbol de directorio, (conocido como windows explorer), control de página, etc.

FIG. 2.4.2.3 Paleta de componentes Win95.

Additional: La tabla de adicionales contiene algunos de los mejores y variados de la paleta de componentes, como mapas de bits, botones aceleradores y componentes de apariencia.

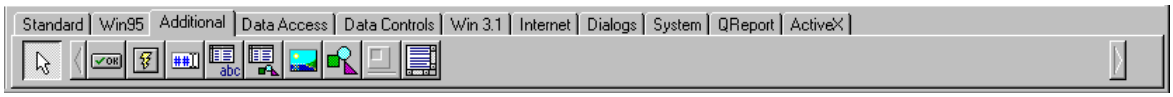


FIG. 2.4.2.4 Paleta de componentes Additional.

Data Access y Data Controls: Se pueden accesar bases de datos y hacer consultas dentro de las aplicaciones que construyas con las facilidades que permite estos 2 grupos de objetos.

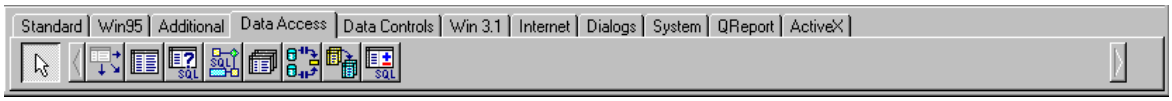


FIG. 2.4.2.5 Paleta de componentes Data Access.

FIG. 2.4.2.6 Paleta de componentes Data Controls.

Win31: Muchos de los controles en Win31 tienen equivalentes en Win95 pero estas pueden usarse para dar sentido a aplicaciones para windows V. 3.1 además de proporcionar un block de notas.

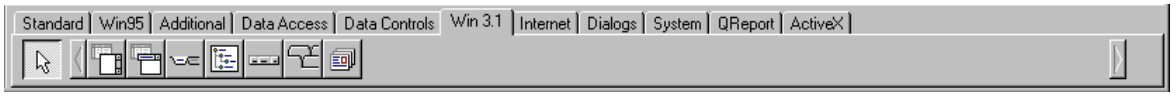


FIG. 2.4.2.7 Paleta de componentes Win 3.1.

Internet: Esta tabla dada por C++ Builder, comprende lo referente al grupo de herramientas de internet.

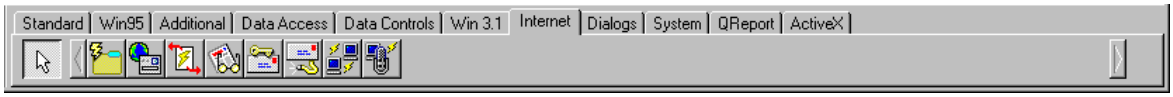


FIG. 2.4.2.8 Paleta de componentes Internet.

Dialogs: Permite hacer cajas de dialogo que agilizan el desarrollo de tus aplicaciones como el abrir y grabar archivos, seleccionar tipos de letras, colores e impresoras y mucho más.

FIG. 2.4.2.9 Paleta de componentes Dialogs.

System: Proporciona controles individuales para seleccionar archivos, directorios o drives.

FIG. 2.4.2.10 Paleta de componentes System.

Qreport: (o Quick Reports) provee de componentes para que pueda fácilmente organizar sus reportes y presenta la facilidad de una vista preliminar.

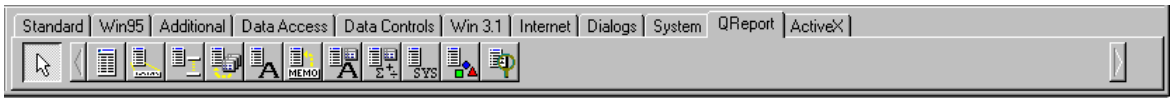


FIG. 2.4.2.11 Paleta de componentes Qreport.

ActiveX: Esta tabla de componentes, contiene un checador de ortografía así como objetos gráficos impresionantes.

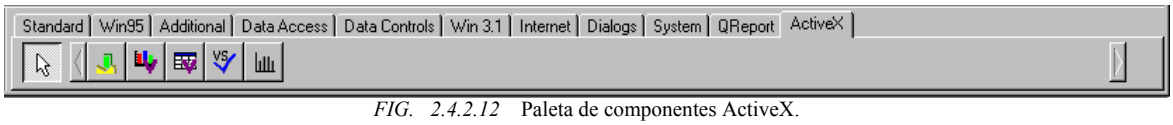


FIG. 2.4.2.12 Paleta de componentes ActiveX.

Durante el desarrollo de este trabajo, solo presentaré el uso de algunos de los objetos de la tabla de componentes, pero la idea es la misma para todos los objetos de esta paleta.

2.4.3 EL EDITOR DE LA FORMA.

Cuando comienzas a trabajar con C++ Builder, tu espacio de trabajo y resultados se suple por la forma principal (Referida en la Fig. II.4.1 Como diseño de la forma).

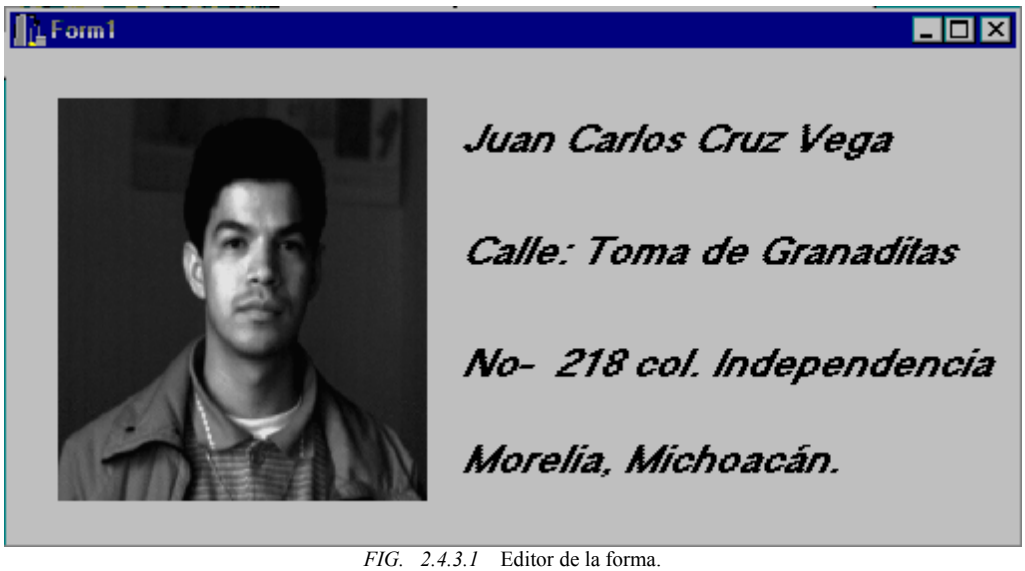


FIG. 2.4.3.1 Editor de la forma.

Cada forma representa una ventana individual en tu aplicación; en la forma puedes diseñar, añadir, eliminar reconfigurar los componentes según las necesidades de tu aplicación.

2.4.4 EL INSPECTOR DE OBJETOS.

El inspector de objetos permite ver las propiedades o características de los objetos que comprendan tu proyecto, por medio de él se pueden cambiar las propiedades de los objetos, también muestra los eventos asociados a los objetos de la aplicación.

Cuando se selecciona un objeto, el inspector de objetos automáticamente cambia al contenido y propiedades de este objeto. Si se oculta, o pierdes el inspector de objetos, lo puedes llamar oprimiendo la tecla de función F11.

El inspector de objetos podría llamarse "editor de objetos", por la propiedad antes mencionada de poder modificar las propiedades de los objetos.

Propiedades: Cuando se comienza un proyecto el inspector de objetos despliega las propiedades de la forma principal como son: nombre, color, altura, ancho, posición etc.
Recordemos que al seleccionar otro objeto, automáticamente mostrará las propiedades de ese objeto.

Eventos: La tabla de eventos despliega para cada objeto los eventos como son:
Al activar el objeto, al oprimir una tecla, al oprimir el mouse, al soltar el mouse, etc. esos eventos son disparados con acciones del usuario, o del sistema operativo mismo. Por ejemplo el evento; *al hacer click en el mouse*, se dispara o hace una acción cuando el usuario hace click con el mouse para ese objeto.

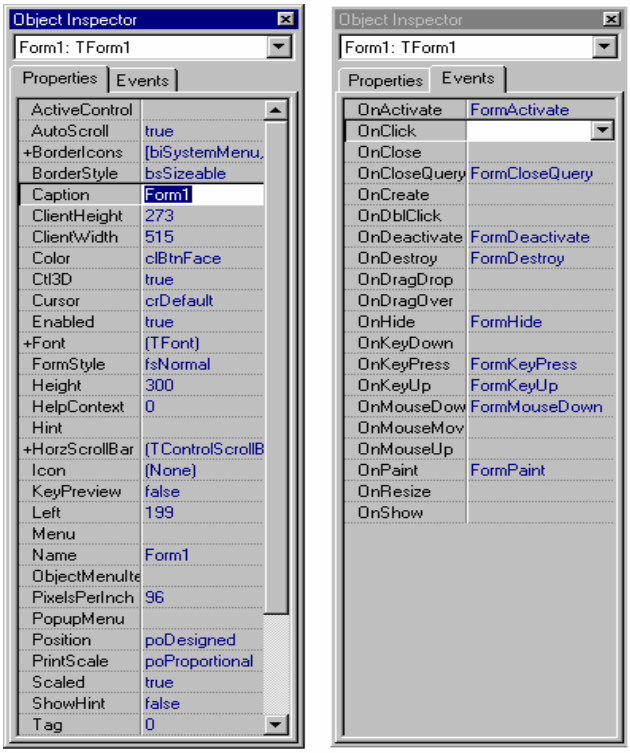


FIG. 2.4.4.1 Inspector de objetos.

2.4.5 EL EDITOR DE CODIGO.

La ventana de edición de código muestra el código actual de tu aplicación C++ Builder. Al añadir objetos y hacer doble click sobre ellos, automáticamente se editará en la ventana de edición la llamada a la función que asociará al evento de ese objeto, dejando el espacio en blanco para que se codifique la acción que se desee para ese evento.

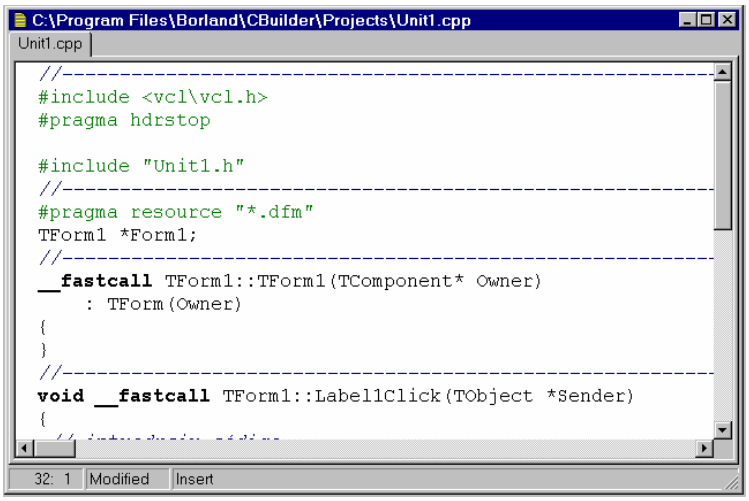


FIG. 2.4.5.1 Ventana de edición de código.

2.5 EL MANEJADOR DE PROYECTOS.

Un sencillo proyecto de C++ Builder está conformado por solo una forma y su código, pero en aplicaciones muy grandes, puede conformarse un proyecto por varias formas, código y varios archivos de cabecera distintos a las librerías que por omisión ya necesita la aplicación, por tanto un proyecto puede integrar varios archivos, para saber cuales son los archivos que comprende un proyecto, usaremos el manejador del proyecto, que muestra el árbol de archivos involucrados en el orden en que fueron añadidos.

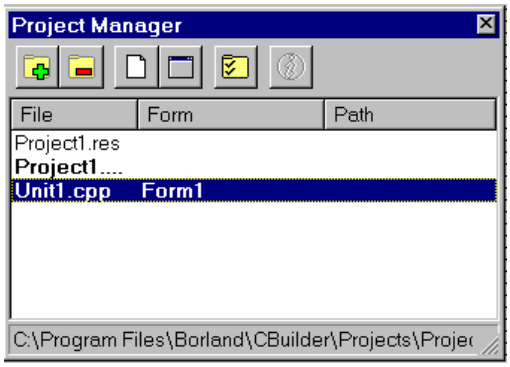


FIG. 2.5.1 Manejador de proyectos.

2.5.1 COMO AÑADIR ARCHIVOS A UN PROYECTO.

Para añadir archivos a un proyecto, seleccione del menú principal: View ⇒ Proyect Manager ⇒ y al hacer doble click obtendrá una caja con la información de los componentes actuales del proyecto (FIG. II.5.1), haga click en el botón de integración al proyecto,  , y saldrá una ventana de adición "Add to proyect", donde puede buscar La ruta de acceso donde se encuentre el archivo a añadir, y haga click en el botón Ok para cerrar la caja de dialogo y obtendrá ya añadido a su proyecto el archivo seleccionado.

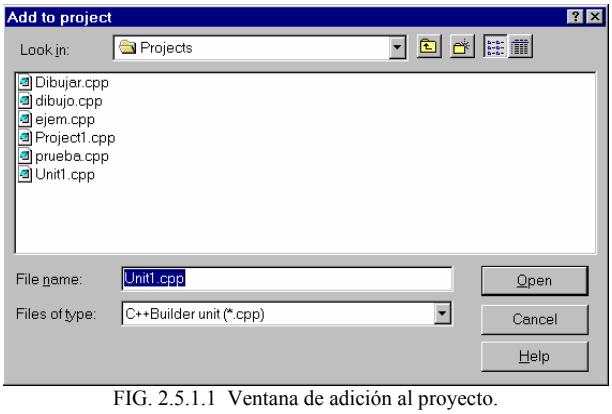


FIG. 2.5.1.1 Ventana de adición al proyecto.

2.5.2 COMO ELIMINAR ARCHIVOS DE UN PROYECTO.

Para eliminar un archivo del proyecto, abra su manejador de proyectos, seleccione el archivo que quiera eliminar y haga click en el botón de remover archivo  y será borrado de la lista. (no de su máquina, solo del proyecto).

2.6 OPCIONES DEL PROYECTO.

Puede accesar a detalles del ambiente de configuración del proyecto actual eligiendo del menú principal Options ⇒ Proyect.

Este comando abre la caja de dialogo de opciones del proyecto, toma el ambiente para cada proyecto y puede ser accesado en cualquier momento durante el desarrollo de la aplicación.

Puede explorar las 6 hojas para ver las opciones que están disponibles. Al iniciar un proyecto, comienza con la configuración que tiene por omisión.

2.6.1 PAGINA DE FORMAS.

Por omisión está incluida la auto creación de una forma, al abrir un nuevo proyecto, evitando que usted al crear una nueva aplicación tenga que pedir una forma. Al crear un nuevo proyecto se crea automáticamente su forma principal y su ventana de edición de código.

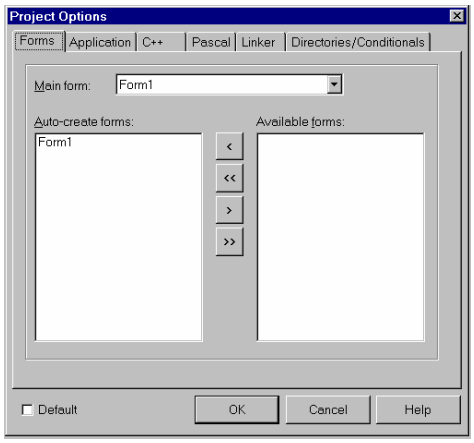


FIG. 2.6.1.1 Opciones del proyecto, página de Formas.

2.6.2 LA PAGINA DE APLICACIONES.

La página de aplicaciones contiene 3 mascaras de configuración:

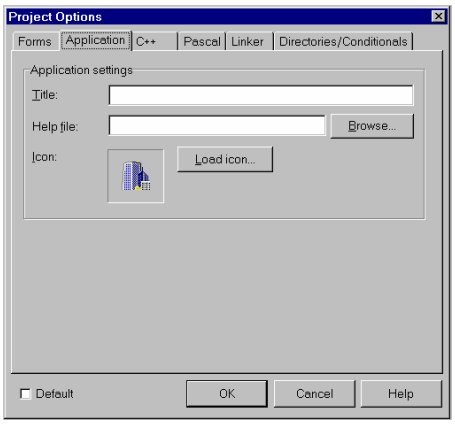


FIG. 2.6.2.1 Opciones del proyecto, página de Aplicaciones.

Titulo: El texto que introduzca será el titulo de la aplicación y será desplegado con el icono cuando se minimice la aplicación.

Help_File: Asociará un archivo de ayuda a su aplicación.

Icon: El archivo ejecutable contendrá el icono que seleccione, por omisión mostrará el de aplicaciones C++ Builder.

2.6.3 LA PAGINA DE C++.

La página C++ es donde inicializa las opciones que desea usar para configurar su particular proyecto. Como opciones de depuración, optimización de código y librerías precompiladas.

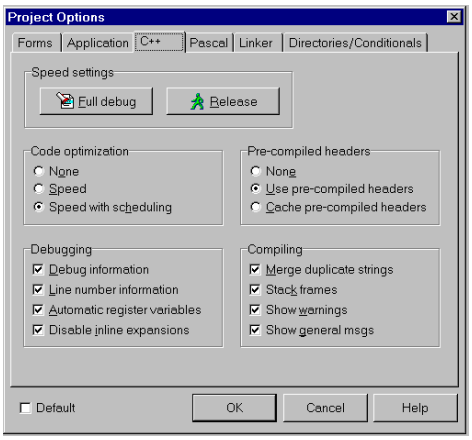


FIG. 2.6.3.1 Opciones del proyecto, página de C++.

2.6.4 LA PAGINA PASCAL .

C++ Builder tiene un hermano mayor llamado Delphi que es una herramienta RAD. Y sus aplicaciones se construyen en lenguaje pascal. Si ha manejado delphi, verá que el ambiente de desarrollo es prácticamente el mismo.
Puede usar funciones hechas en delphi dentro de C++ Builder.

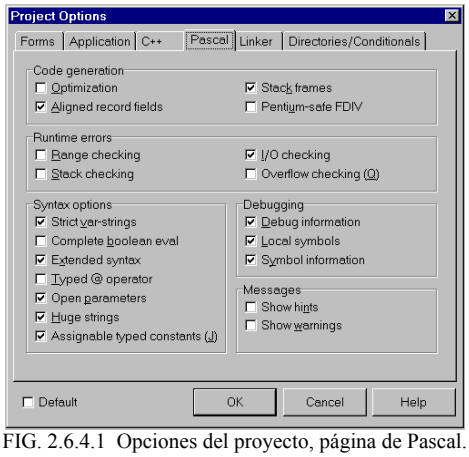


FIG. 2.6.4.1 Opciones del proyecto, página de Pascal.

2.6.5 LA PAGINA DE ENLAZADO.

La mayoría del tiempo se querrá que la aplicación genere un archivo ejecutable, en otras ocasiones querrá que su construcción sea una librería para diferentes programas, en este caso usted puede llamar a la opción de librerías de enlazado dinámico (DLL) y las puede generar cambiando la opción a generar DLL´s.

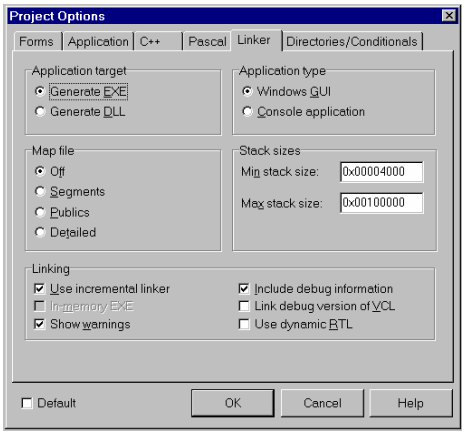


FIG. 2.6.5.1 Opciones del proyecto, página Linker.

2.6.6 PAGINA DIRECTORIOS/CONDICIONAL.

Usted no necesita cambiar la configuración en esta pagina, a menos que tenga en diferentes directorios de los que por omisión genera la instalación de C++ Builder.
Y debe tener cuidado en estos parámetros.

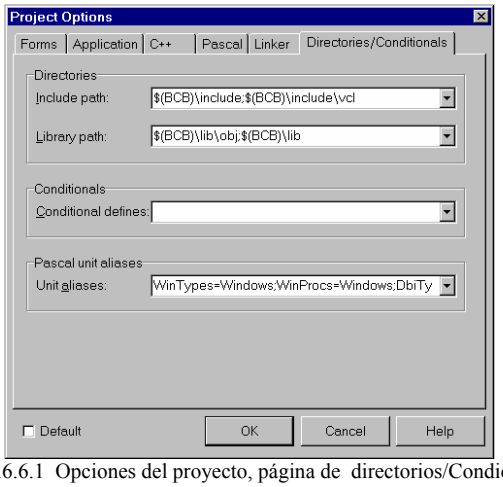


FIG. 2.6.6.1 Opciones del proyecto, página de directorios/Condicional.

2.7 ARCHIVOS FUENTE GENERADOS POR C++ BUILDER.

Cada una de las siguientes extenciones son vitales para cualquier proyecto de C++ Builder.

Los siguientes archivos contienen detalles del diseño de sus proyectos y formas, tenga cuidado en no perder ninguno ya que puede utilizarlos si quisiera modificar alguna aplicación hecha.

Project1.mak: Este es el archivo principal de opciones del proyecto. Un archivo .mak se requiere en cada aplicación; es un archivo de texto que puede examinar eligiendo del menú principal: View ⇒ Project MakeFile. Este archivo contiene instrucciones de cómo C++ Builder construirá el archivo (.exe) ejecutable para el proyecto.

Project1.Cpp: Este archivo contiene el código principal de la aplicación, comparte el mismo nombre del proyecto, lo crea automáticamente C++ Builder al darle nombre al proyecto; contiene el código de iniciación y terminación del programa. si quiere examinar este archivo, solo elija del menú principal View ⇒ Project Source.

Unit1.cpp: Este es el código que usted le da a los eventos de los objetos que tendrá en su forma final, este código es el que introduce en la ventana de edición, o en su editor de código.

Unit1.h: Para cada archivo .cpp, C++ Builder crea automáticamente un .h correspondiente. El archivo de cabecera contiene la declaración de la forma y menciona a C++ Builder la lista de componentes y los eventos que tendrá la aplicación.

Unit1.dfm: El archivo .dfm contiene la información, definición y declaración de la forma y otros detalles importantes como: tamaño, color, títulos, fondos etc. así como detalles del resto de los componentes utilizados en la forma.

La extensión .dfm indica que este archivo oculta los datos de la forma en formato binario. Este archivo no se puede leer, pero puede convertirlo para observar su contenido, solo seleccione la forma, con el botón derecho del mouse haga click y saldrá un menú de opciones, escoja View as text, para ver la información de su forma.

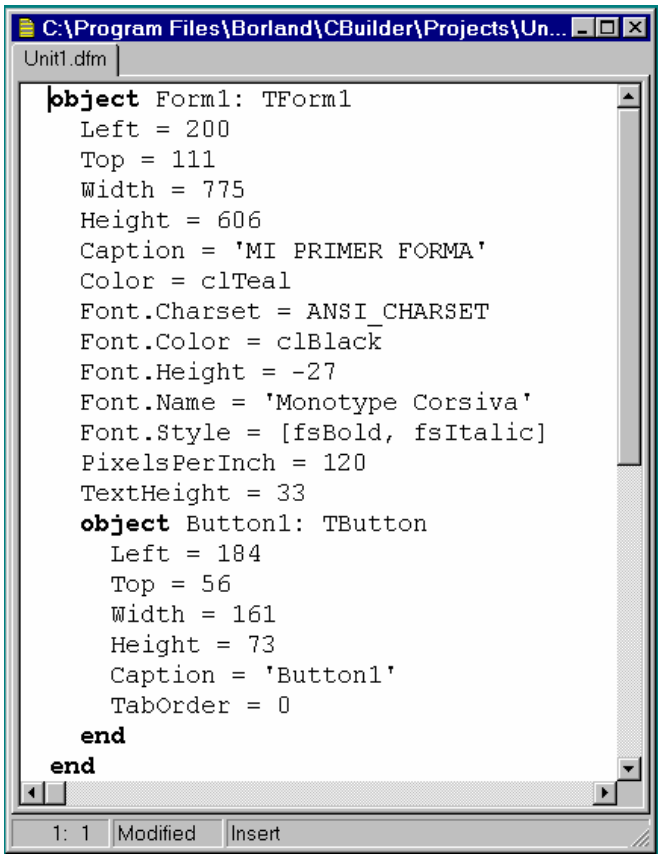


FIG. 2.7.1 Ver la forma como texto

y de igual manera, para regresar a verlo como forma, seleccione con el botón derecho del mouse, escoja View as Form, y regresará a su forma.

Unit1.obj Cuando se compila el proyecto se crea el archivo binario con extensión .obj. Cada aplicación contendrá un archivo .obj, este archivo cambiará cada vez que se reconstruya el proyecto.

Project1.exe o Project1.dll: Este es el archivo final según se halla elegido en su proyecto, el .exe podrá ejecutarse, y los dll son librerías dinámicas que pueden utilizarse desde otros programas.

Project1.dsk: Contiene la información de la configuración que tenía su hardware al momento de crear o finalizar su proyecto, para al momento de ejecutarse lo haga sobre la misma configuración.

Project1.il?: Al ver la extensión .il? indica que el archivo es usado al enlazar, C++ Builder usa una tecnología de compilación muy rápida, lo que hace que la compilación o recompilación de sus proyectos sea en cuestión de segundos.

Capítulo III

Cómo Crear un Nuevo Proyecto

Con C++ tenemos una poderosa herramienta para la construcción de aplicaciones; todos estamos de acuerdo.

Si ahora le anexamos como parte de su estructura docenas de los objetos mas utilizados por todos los desarrolladores, encontramos un software que facilita la construccion de aplicaciones, por lo que es aún mas poderoso y efectivo; ahora el desarrollador se tiene que preocupar por que sus funciones asociadas a los objetos que seleccione para su aplicación hagan lo que se desea que realicen, y no se tiene que preocupar por como lograr la visualización en pantalla.

La creación de botones, menus, ventanas de dialogo, barras de estado, la presentación de campos tipo memo, la visualización de directorios, la presentación de imagenes, secuencias de impresión, y muchas más cosas que comunmente se realizan con demasiadas líneas de código, las podrá tener con solo elegir un componente que C++ Builder le ofrece.

Veamos la facilidad con que podemos utilizar estos componentes.

3.1 CREANDO UN NUEVO PROYECTO.

Este capitulo lo iniciaremos con un pequeño ejemplo que muestre los pasos a seguir para crear un nuevo proyecto y cambiar las propiedades de la forma, y conforme avancemos se mostrarán algunas de las propiedades de la paleta de componentes, como mencioné anteriormente; no presentaré cada uno de ellos, pero lo que mostraré es lo esencial para el uso todos ellos.

Comencemos con una nueva forma:

1.- Elija del menú principal File $\Rightarrow$ New Application, para tener una forma en blanco y un nuevo programa.

2.- Seleccione la forma haciendo clic con el mouse en ella. Observe el inspector de objetos, si no está presente, presione F11.

3.- En el inspector de objetos localice la propiedad Caption (TITULO), seleccione esta propiedad y cambie Form1 por el titulo: Mi primer Programa C++ Builder.

4.- Presione la tecla de función F9 lo que ejecutará su programa y tendrá el siguiente resultado; La ventana de la forma con el texto: Mi primer Programa C++ Builder.

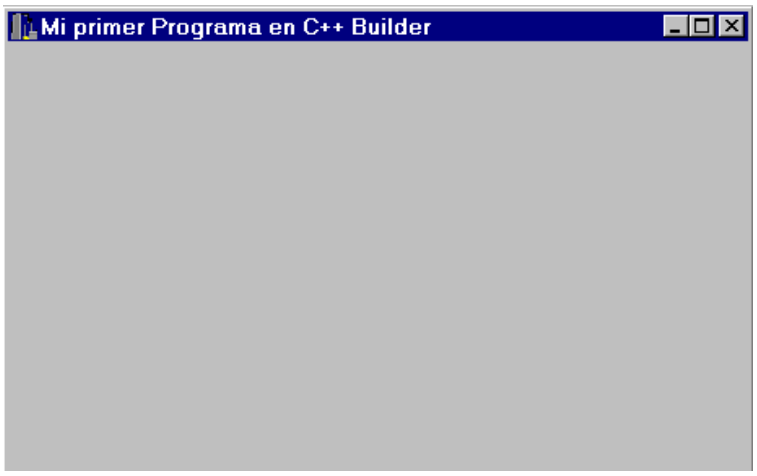


FIG. 3.1.1 Ejecución de un programa sin código en C++ Builder.

Usted ha hecho cambios a las propiedades de la forma usando el inspector de objetos.
A continuación daremos un breve repaso por las propiedades de la forma mostrados en el inspector de objetos.

LAS PROPIEDADES HINT Y SHOWHINT. La propiedad **Hint** es un pequeño texto que mostrará un mensaje al usuario cada vez que pase lentamente el puntero del mouse sobre la forma, en este espacio puede introducir cualquier texto, y si quiere activarlo, tendrá que activar la propiedad **ShowHint** (cambiarlo a true), con esto el texto que halla tecleado en Hint, aparecerá cuando pase el puntero sobre la forma.

LAS PROPIEDADES HEIGHT Y WIDTH. Cuando se cambia el tamaño de una forma, los valores numéricos de esta propiedad cambian automáticamente.
Usted puede dar dimensiones exactas a su forma dando valores numéricos (en pixeles) a estas propiedades Largo y Ancho, que también las puede cambiar con otras dos propiedades que darán posición en su ventana principal a las dimensiones escogidas para su forma; estas propiedades son **Top**(Número de pixeles desplazados desde la parte superior.) y **Left**(Número de pixeles desplazados desde la parte izquierda de su monitor.)

LA PROPIEDAD VISIBLE. Esta propiedad permite elegir entre 2 valores para su objeto, true o false, cuando la propiedad visible esta en verdadero(true), el usuario puede observar la forma (o el objeto) cuando la aplicación se está ejecutando, si la propiedad visible se

encuentra en falso(False), al momento de la ejecución de la aplicación el usuario no podrá ver este objeto.

LA PROPIEDAD COLOR. La propiedad color cambia inmediatamente la apariencia de la forma al color seleccionado.

LAS PROPIEDADES SETTYPE. Una de las formas de distinguir que propiedades pueden tener un tipo inicial, es que en la parte izquierda de las propiedades tienen un signo (+). Esto es que tienen varias características más que puede adoptar esta propiedad, usted puede verlas y modificarlas seleccionando la propiedad y elegir de el menú de opciones que se presentará la opción expand.

LA PROPIEDAD FORMSTYLE. Tiene dos opciones principales, fsMDIform y fsMDIChild; usted puede desear en su aplicación que una forma contenga a otras, este tratamiento de formas, llamado Interfase Documento Múltiple, hace muy fácil este tipo de aplicaciones, veamos cuan sencillo es hacer un ejemplo:

1.- Seleccione del menú de principal File $\Rightarrow$ New Application; Para obtener una nueva aplicación.

2.- Con el inspector de objetos, ponga un nuevo nombre a su forma como lo hizo en el ejemplo anterior a: *Forma Principal*, y en la propiedad FormStyle cambie a la opción fsMDIForm.
3.- Añada una segunda forma al proyecto eligiendo del menú principal File $\Rightarrow$ New Form y aparecerá una nueva forma.

4.- Con el inspector de objetos cambie ahora a esta forma la propiedad caption, por el siguiente texto: *Ventana secundaria*,(Puede tener varias ventanas secundarias) y en la propiedad FormStyle cambie a la opción fsMDIChild. Y es todo lo que necesita para correr esta simple aplicación.

5.- Oprima la tecla de función F9 para ejecutar la aplicación MDI.

Cuando corra la aplicación obtendrá el siguiente resultado.

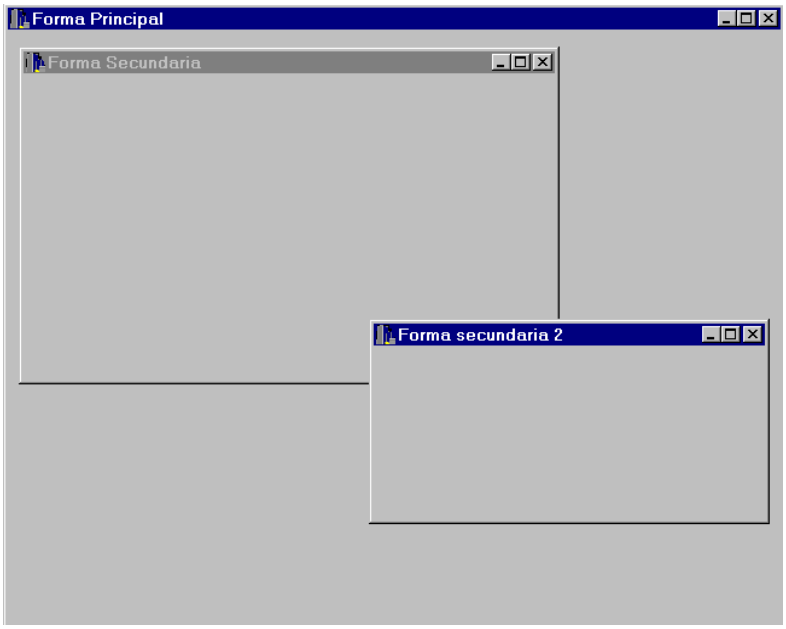


FIG. 3.1.2 Interfase Documento Múltiple.

La ventana hija será parte permanente de la aplicación hasta que se cierre, puede minimizar mover o maximizar la ventana secundaria dentro de la ventana principal.

LA PROPIEDAD BORDERICONS. Esta propiedad permite especificar el tipo de borde e iconos que llevará en la barra de titulo su aplicación.

Subpropiedad	Que sucede cuando se selecciona la subpropiedad.
BiSystemMenu	Cuando selecciones con un clic con el botón derecho del mouse cualquier parte de la barra de titulo de la forma, presenta un menú de control con las opciones que permite la configuración de su proyecto.
BiMinimize	Coloca en la barra de titulo un pequeño icono que permite minimizar la forma con que se está trabajando al momento de ejecutarse.
BiMaximize	Coloca en la barra de titulo un pequeño icono que permite maximizar la forma con que se está trabajando al momento de ejecutarse.
BiHelp	Coloca un botón "?" en la forma. Cuando el usuario hace clic proporciona una pequeña definición del objeto en contexto sensitivo.

Si usted selecciona en todas las opciones False, no tendrá forma de cerrar su aplicación por medio de estos componentes así que tendrá que implementar una forma de cerrar y salir.

LA PROPIEDAD BORDERSTYLE. Estas subpropiedades están divididas en la que permiten que la forma se escale y las que no.

Subpropiedad	Que sucede cuando se selecciona la subpropiedad.
BsNone	No le permite al usuario ni mover, ni cerrar, ni maximizar ni escalar. Deberá implementar una forma de cerrar su aplicación ya que esta opción no muestra una barra de titulo..
BsSingle.	Permite mover, minimizar o maximizar pero no escalar.
BsDialog	Solo permite mover, ni maximizar ni minimizar ni escalar.
BsSingle	Permite mover, minimizar o maximizar pero no escalar.
BsSizeWin	Solo permite mover y escalar, no permite ni maximizar ni minimizar ni escalar.
BsToolWindow	Solo permite mover, ni maximizar ni minimizar ni escalar.
BsSizeable	Permite todas las acciones, mover, escalar, minimizar, maximizar.

LA PROPIEDAD ICON. Por omisión si no se selecciona ninguno, el C++ Builder presenta el icono de las aplicaciones C++ Builder.

Pero se puede cambiar haciendo doble clic en el botón Load, para seleccionar de algún directorio el archivo del icono deseado. esta propiedad busca archivos con extensión .ico.
C++ Builder tiene una biblioteca de archivos .ico en el subdirectorio:
cbuilder\images\icons. Y presenta un editor de iconos, puede experimentar con esta herramienta seleccionando del menú principal: Tools $\Rightarrow$ Image Editor.

LA PROPIEDAD POSITION. Determina donde será desplegada la forma en la pantalla de la computadora posición (x,y).

Subpropiedad	Que sucede cuando se selecciona la subpropiedad.

PoDesigned	Aparece la forma en la posición en la que estaba al momento de correr el programa.
PoScreenCenter	Si se escala se escalará de manera de quedar en el centro de la pantalla.
PoDefault	Deja la forma en la posición en que la coloque el usuario.
PoDefaultPosOnly	Deja la forma en la posición en que la coloque el usuario.
PoDefaultSizeOnly	Deja la forma en la posición en que la coloque el usuario.

LA PROPIEDAD WINDOWSTATE. Esta propiedad muestra las características de la forma al momento de ejecutarse

WsNormal	La forma aparece de la manera en que se definió al momento de crearse (dimensiones y posición).
WsMinimized	Al momento de ejecutar la aplicación, la forma aparecerá minimizada en su pantalla.
WsMaximized	Al momento de ejecutar la aplicación, la forma abarcará la totalidad de la pantalla.

3.2 COMO FUNCIONAN LOS EVENTOS.

Los eventos son acciones relacionadas a los objetos y su comportamiento que llevarán a cabo durante la ejecución del programa.

3.2.1 EVENTOS DEL MOUSE.

OnClick: Este evento ocurre cuando el usuario hace clic dentro del área de la forma los siguientes pasos muestran un ejemplo de cómo se usaría este evento.

1.- Elija del menú principal File $\Rightarrow$ New Application.

2.- En la hoja de eventos del inspector de objetos haga doble clic en el evento OnClick y el editor de código presentará la estructura de la función llamada Form1Click y el cursor se coloca para que pueda introducir la lista de instrucciones.

3.- Teclee la siguiente instrucción en el editor de código.

ShowMessage("ha ocurrido un Clic del Mouse");

de manera que su editor de código luzca así:

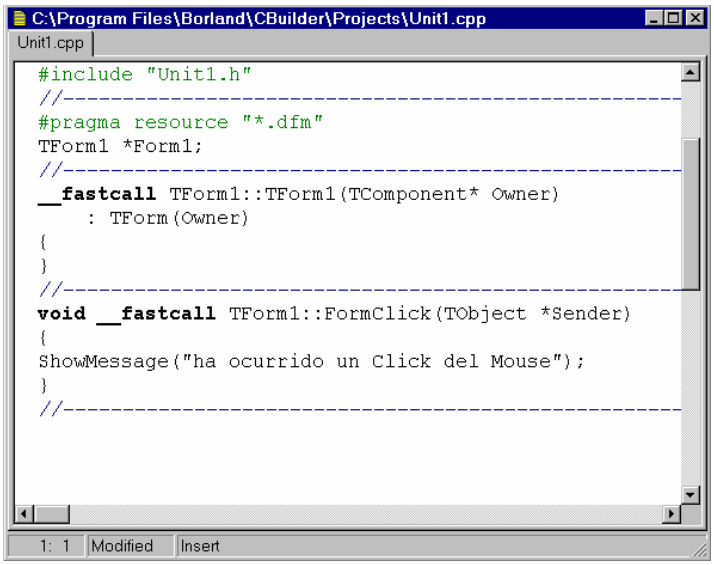


FIG. 3.2.1.1 Edición de una línea de código en el evento OnClick de la Forma principal.

4.- Presione la tecla de función F9 para ejecutar la aplicación.

5.- Haga clic en la forma y observe el resultado del programa.

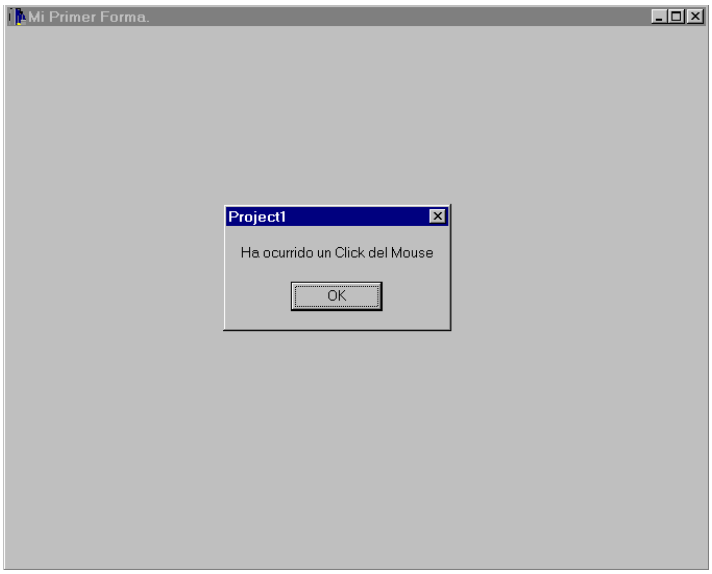


FIG. 3.2.1.2 Resultado de ejecutar el programa OnClick.

Para experimentar otro evento del mouse siga el siguiente ejemplo.

1.- Elija del menú Principal File $\Rightarrow$ New Application.

2.- Si no esta presente el inspector de objetos presione la tecla de función F11.

3.- En la hoja de eventos del inspector de objetos haga doble clic en el evento OnMouseDown para obtener la función en el editor de código. Introduzca en la función el siguiente código. Caption = “Se realizó un clic con el Mouse”;

4.- Haga doble clic en el evento OnMouseUp y teclee en el editor de código la instrucción Caption = “ “;

5.- Haga doble clic en el evento OnMouseMove e introduzca el siguiente código en esa función.

```
MessageBeep(0);
```

6.- Presione la tecla de función F9 para ejecutar su programa.

El resultado es que cuando arrastra el mouse sobre la forma se escucha un sonido, cuando se hace clic sobre la forma en la barra de titulo aparece: Se realizó un clic con el Mouse y se borra.

3.2.2 EVENTOS DEL TECLADO.

Estos eventos suceden cuando el usuario presiona un determinada tecla, experimente con el siguiente aplicación.

1.- Elija del menú principal File ⇒ New Application.

2.- Si no esta presente el inspector de objetos presione la tecla de función F11.

3.- En la hoja de eventos del inspector de objetos haga doble clic en el evento OnKeyPress y teclee el siguiente código:

```
if((Key>=”0”) && (Key <= “9”))
MessageBeep(0);
```

4.- Presione la tecla de función F9 para ejecutar la aplicación.

El resultado es que si se oprime una tecla del 0 al 9, se escuchará un beep.

3.2.3 EVENTOS DE SISTEMA.

No todos los eventos ocurren por acciones que realice el usuario, algunos son ejecutados por el sistema operativo Windows.

Evento	Cuando ocurre	Para que se usa.
OnActivate	Cuando la forma se activa.	Mostrar la forma.
OnClose	Después de cerrar la forma.	Cierra la aplicación libera los recursos que fueron usados por la aplicación.
OnCloseQuery	Cuando se intenta cerrar la forma.	Checa que los datos se hallan grabado antes de cerrar las bases de datos.
OnCreate	Cuando se crea la forma	Inicializar el código.
OnDeactivate	Cuando se para la ejecución de una aplicación.	Guarda información en variables temporales.
OnDestroy	Al intentar cerrar la aplicación.	Libera memoria utilizada en la aplicación.

OnPaint	Cuando la forma necesita ser redibujada.	Redibuja y muestra datos en el área que se redibujó.
OnResize	Cuando cambia el tamaño de la forma	Mover o escalar la forma.

Siga el siguiente ejemplo de uso del evento OnResize.

1.- Elija del menú principal File ⇒ New Application.

2.- Si no esta presente el inspector de objetos presione la tecla de función F11.

3.- En la hoja de eventos del inspector de objetos haga doble clic en el evento OnResize

4.- En la hoja standard de la paleta de componentes haga clic en el componente Label (Etiqueta) y colóquelo en la forma haciendo otro clic.

5.- Use el inspector de objetos para cambiar el titulo (Caption) de Label1 a "Usando una etiqueta"

6.- Seleccione la forma haciendo clic en ella.

7.- En el editor de código haga doble clic en el evento OnResize y escriba el siguiente código.

```
Label1->Left = (clientWidth/2) – (Label1->Width/2);
Label1->Top = (clientHeight/2) – (Label1->Height/2);
```

8.- Presione la tecla de función F9 para ejecutar el programa.

Si usted escala la forma, verá que el texto siempre aparecerá en el centro de la forma.

3.3 EXPLORANDO LA PALETA DE COMPONENTES.

Usted puede seleccionar cualquier cantidad de objetos de la paleta de componentes y colocarlos en su forma, primero seleccionando con doble clic del mouse en el componente deseado y aparecerá en algún lugar de su forma, acomódelos donde usted quiera que aparezcan al momento de ejecutar la aplicación.

3.3.1 USO DE PANELES, ETIQUETAS, LINEAS DE EDICION, Y BOTONES DE SELECCION.

1.- Elija del menú principal File ⇒ New Application.

2.- Si no esta presente el inspector de objetos presione la tecla de función F11.

3.- De la paleta de componentes en la hoja Standard seleccione 2 objetos Panel haciendo doble clic en el objeto (Panel)y acomódelos de la siguiente manera:

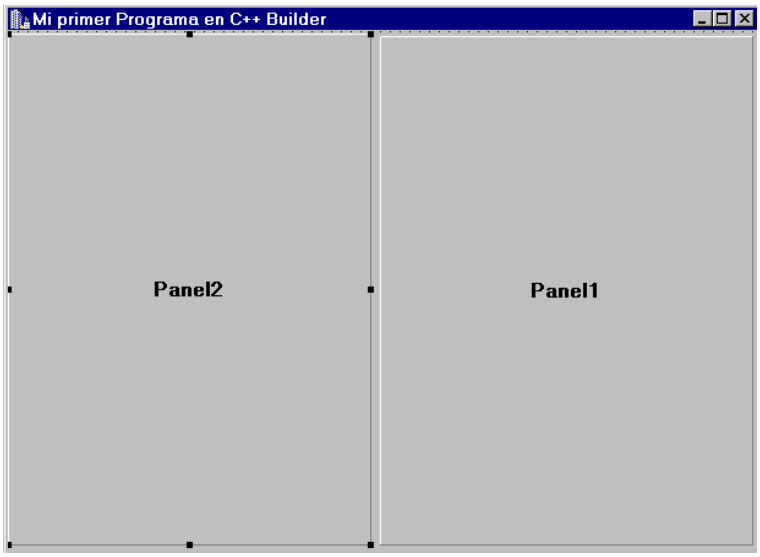


FIG. 3.3.1.1 Acomodo de dos componentes Panel en la Forma para una aplicación particular.

4.- De la paleta de componentes en la hoja Standard seleccione 2 objetos Etiqueta(Label) haciendo doble clic en el objeto (Label), con el inspector de objetos cambie el titulo a la Label1 por el siguiente “Etiqueta en panel1 “ y acomódelos de la siguiente manera:

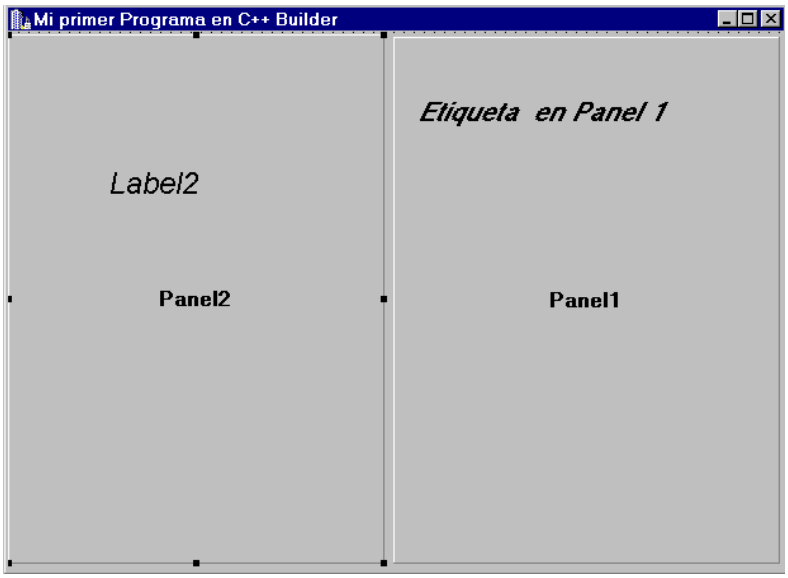


FIG. 3.3.1.2 Acomodo de dos componentes Label en la Forma para una aplicación particular.

5.- De la paleta de componentes en la hoja Standard seleccione 2 objetos Línea de Edición haciendo doble clic en el objeto (Edit), con el inspector de objetos cambie el titulo (en la propiedad Text) a Edit1 por el siguiente "Línea de edición en panel1 " y acomódelos de la siguiente manera:

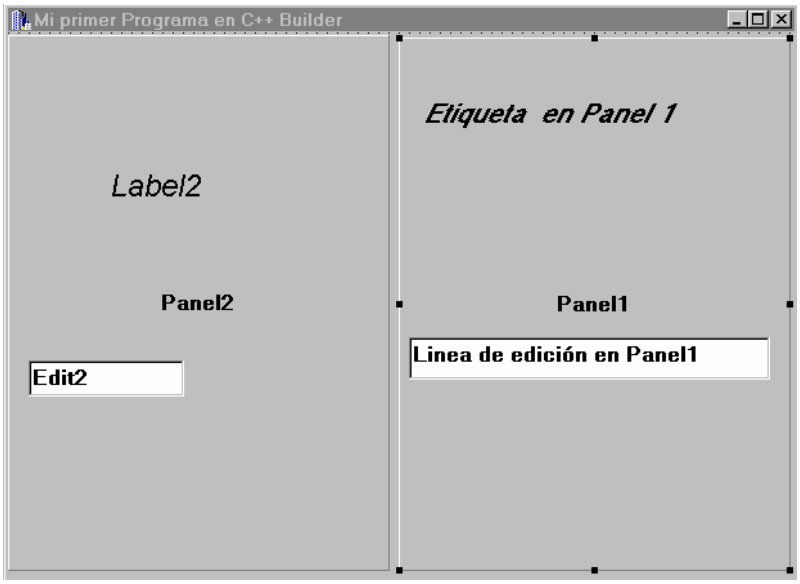


FIG. 3.3.1.3 Acomodo de dos componentes Edit en la Forma para una aplicación particular.

6.- De la paleta de componentes en la hoja Standard seleccione 2 objetos Botón haciendo doble clic en el objeto (Button), con el inspector de objetos cambie el titulo a Botton1 por el siguiente "Copia a Panel2" y botton2 por el siguiente: "Salir" y acomódelos de la siguiente manera:

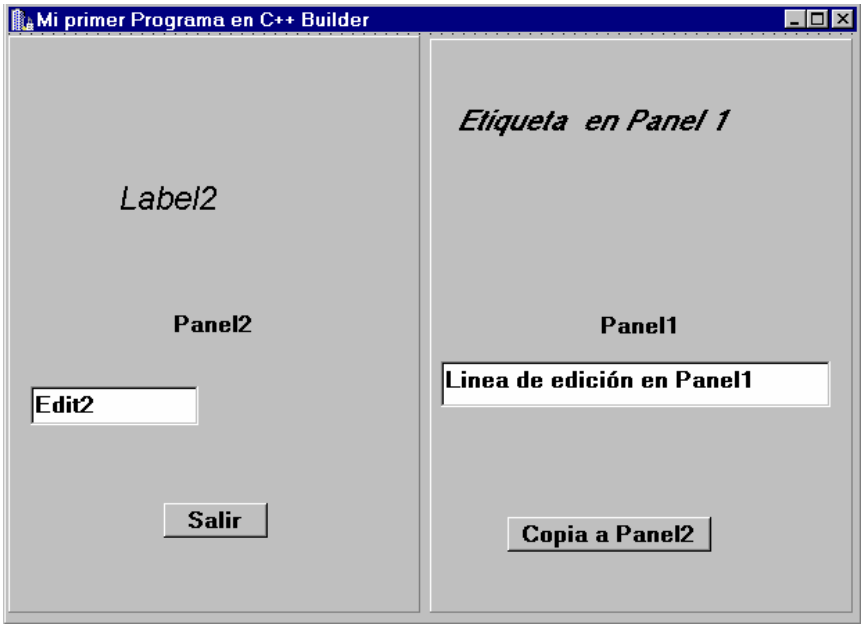


FIG. 3.3.1.4 Acomodo de dos componentes Button en la Forma para una aplicación particular.

7.- Haga doble clic sobre el botón “Copia a Panel2” Para obtener su función.

8.- Repita la acción pero ahora en el botón Salir.

9.- Teclee el siguiente Código en la Función. Button1Click (Copia a Panel2):

```
Label2->Color=clOlive;
Label2->Caption = Label1->Caption;
Edit2->Color=clTeal;
Edit2->Text = Edit1->Text;
```

10.- Teclee el siguiente Código en la Función. Button2Click (Salir):

```
exit(0;)
```

11.- Presione la tecla de función F9 para ejecutar el programa.
Y oprime el botón de copia.

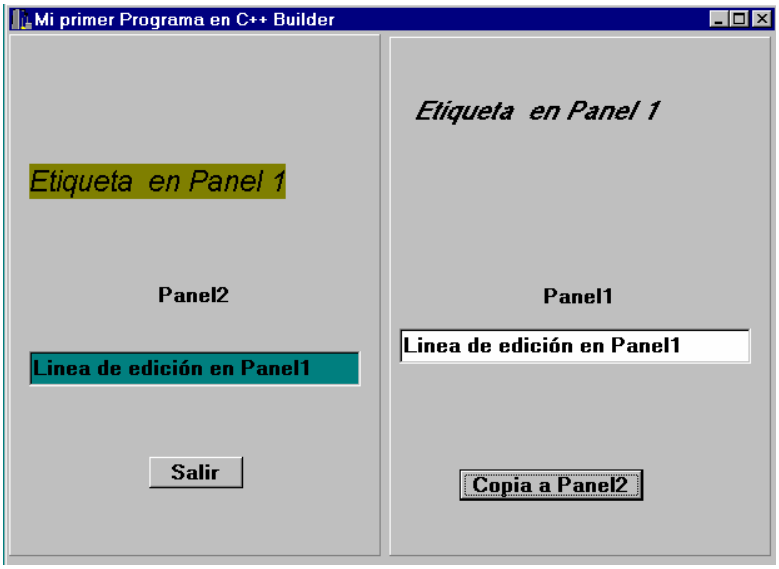


FIG. 3.3.1.5 Resultado de la ejecución de ésta aplicación particular.

Y si oprime el Botón salir se destruirá la forma.

3.3.2 USO DE COMPONENTES MEMO.

El componente Memo sirve para presentar varias y grandes líneas de texto.

1.- Elija del menú principal File ⇒ New Application.

2.- Si no esta presente el inspector de objetos presione la tecla de función F11. Seleccione del inspector de objetos el evento OnCreate haciendo doble clic y escriba el siguiente código.Memo1->Lines->LoadFromFile("Unit1.cpp");

3.- De la paleta de componentes en la hoja Standard seleccione 1 objeto Panel haciendo doble clic en el objeto  ()y acomódelo de manera que abarque toda la forma:

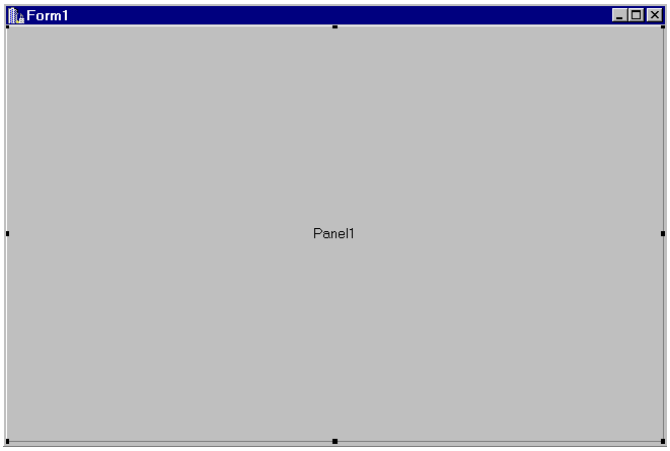


FIG. 3.3.2.1 Acomodo de un componente Panel en la Forma para una aplicación particular.

4.- Seleccione de la paleta de componentes Standard, el objeto Memo  (), y extiéndalo sobre el panel de la siguiente manera:

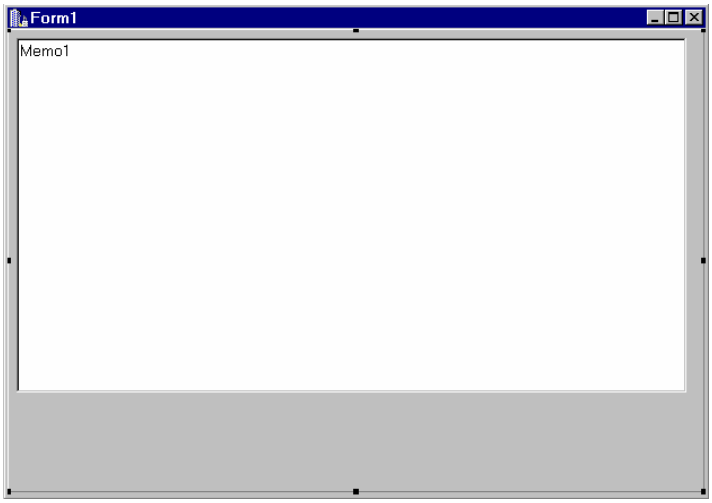


FIG. 3.3.2.2 Acomodo de un componente Memo en la Forma para una aplicación particular.

5.- Seleccione de la paleta de componentes Standard, el 3 objetos Botón  (), y acomódelos sobre el panel, cambie el titulo de cada botón (Grabar, Limpiar, Salir), de la siguiente manera:

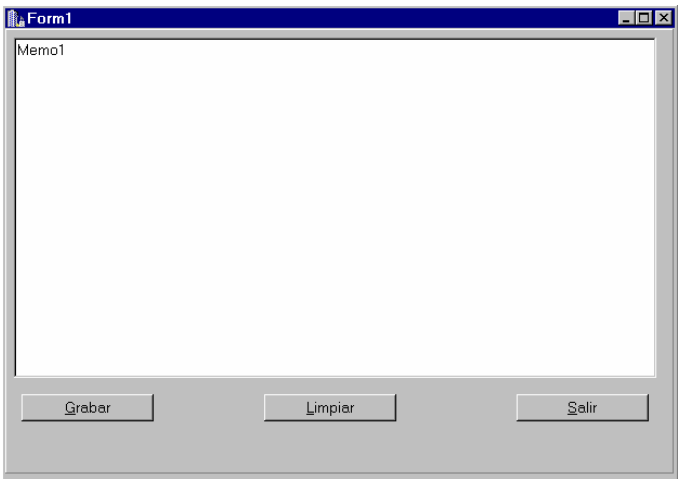


FIG. 3.3.2.3 Acomodo de tres componentes Button en la Forma para una aplicación particular.

6.- De la paleta de componentes en la hoja Win95 seleccione el objeto de barra de progreso  (ProgressBar) y colóquelo de la siguiente manera:

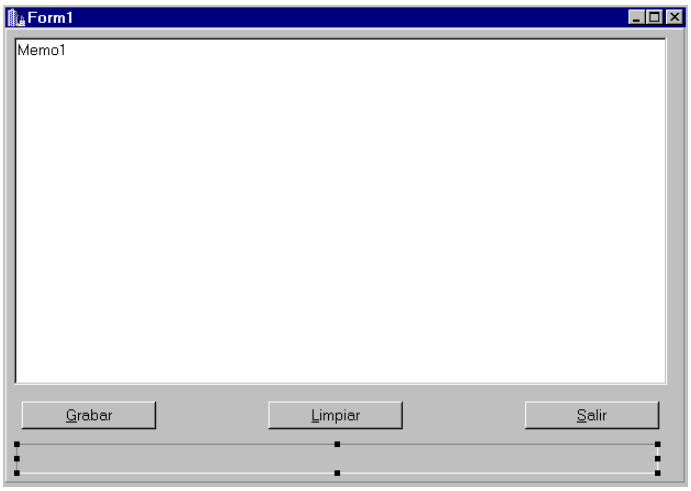


FIG. 3.3.2.4 Acomodo de un componente ProgressBar en la Forma para una aplicación particular.

7.- Con el inspector de objetos, modifique la propiedad Visible a False y la propiedad Max a 40 y la propiedad step en 10.

8.- Haga doble clic en el botón Grabar y teclee el siguiente código:

```
Memo1->Lines->SaveToFile("Unidad1.txt");
ProgressBar1->Visible=true;
for(ProgressBar1->Position=0;ProgressBar1->Position<=ProgressBar1->
Max;ProgressBar1->Position+=10)
 {  Application->ProcessMessages();
  sleep(1); }
 ProgressBar1->Visible=false;
```

9.- Haga doble clic en el botón Limpiar y teclee el siguiente código:

```
Form1->Invalidate();
```

10.- Haga doble clic en el botón Salir y teclee el siguiente código:

```
exit(0);
```

11.- Salve su proyecto en el directorio raíz de c:\ Con los nombres que por default da el C++ Builder.

12.- Incluya el archivo de cabecera #include <dos.h>,y presione la tecla de función F9 para ejecutar su aplicación. Y al correr su programa obtendrá el siguiente resultado.

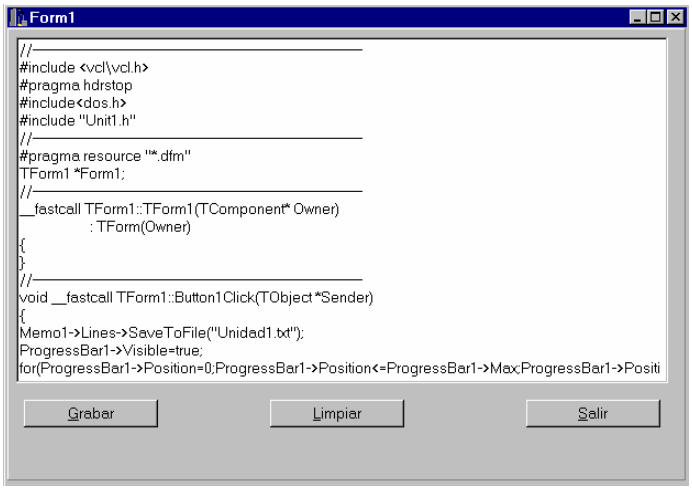


FIG. 3.3.2.5 Resultado de la ejecución de ésta aplicación particular.

Y al oprimir el botón de grabar se verá lo siguiente:

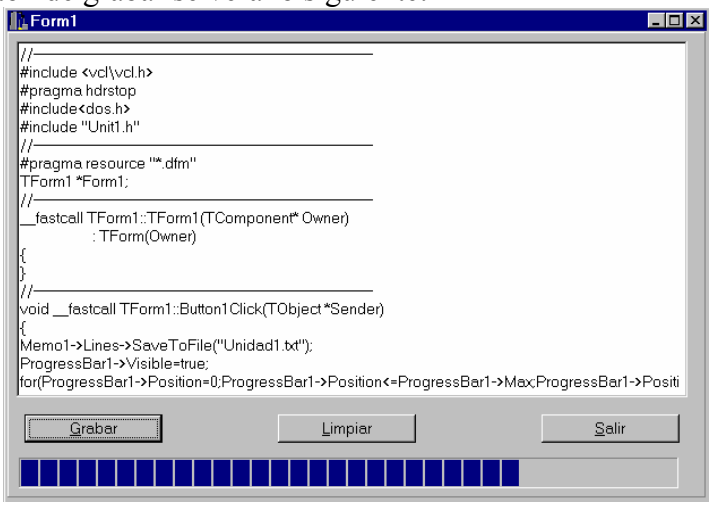


FIG. 3.3.2.6 Resultado de la ejecución de ésta aplicación particular en la opción Grabar.

3.3.3 USO DEL COMPONENTE LISTBOX.

El componente ListBox muestra una lista de opciones de las que se puede elegir, puede elegir una o varias, no permite editar (no por esta ventana, por otros medios si), lo que permite que el usuario solo seleccione de entre las opciones que presenta este componente. Este componente puede contener hasta 32,000 artículos de los que el usuario puede escoger.

Para comprender mejor esta opción vea el siguiente programa:

1.- Elija del menú principal File ⇒ New Application.

2.- Si no esta presente el inspector de objetos presione la tecla de función F11.

3.- Seleccione un objeto ListBox (ListBox) de la paleta de componentes Standadrd.

4.- En la propiedad Items haga doble clic; aparecerá una ventana de edición, en ella; teclee una lista de datos como la que se muestra en la figura.

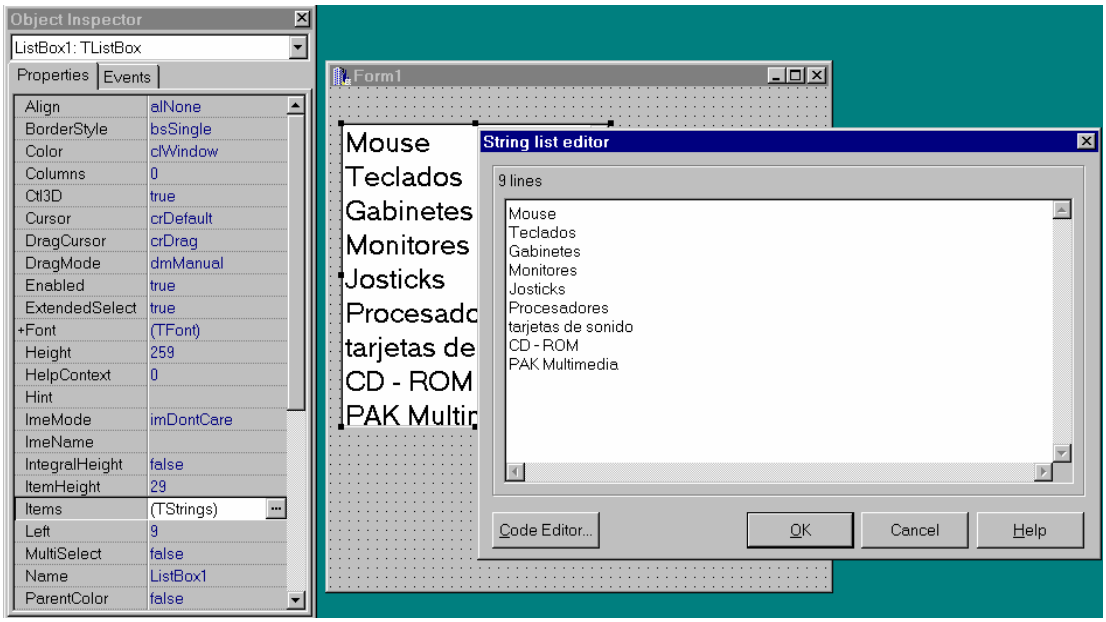


FIG 3.3.3.1 Editor de líneas para un ListBox.

5.- Cuando finalices selecciona Ok. Con el mouse.

6.- Seleccione un objeto etiqueta (Label), de la paleta de componentes Standard y colócalo en la forma de manera que su forma se luzca así:

FIG 3.3.3.2 Componente ListBox Editado.

7.- Selecciona el objeto ListBox de tu forma con un clic del mouse; posteriormente con el inspector de objetos en la hoja de eventos haz doble clic y en el editor de código en su función teclea el siguiente código:

```
Label1->Caption = ListBox1->Items->Strings[ListBox1->ItemIndex];
```

8.- Ejecute su programa y vea que al seleccionar una opción de la lista se presenta ese texto en la etiqueta.

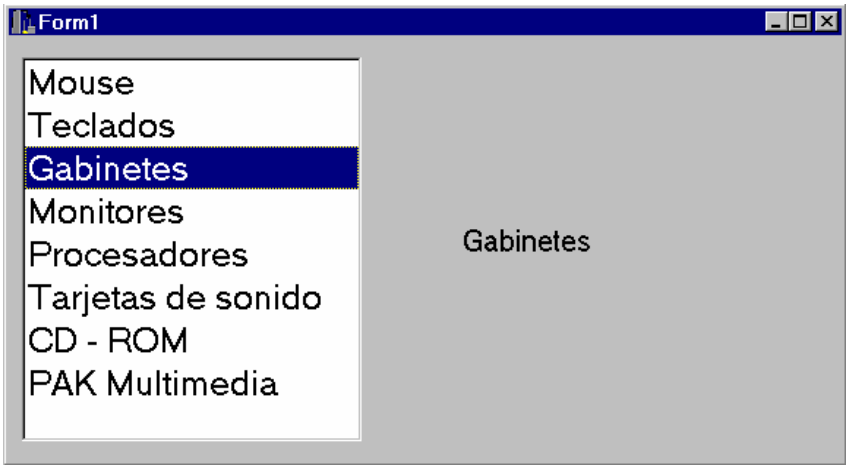


FIG 3.3.3.3 Componente ListBox Editado.

3.3.4 USO DEL COMPONENTE COMBOBOX.

El componente ComboBox combina las ventajas de EditBox y ListBox. Este componente ofrece a sus usuarios una selección de cajas de edición de manera que no ocupen mucho espacio en su forma.
Presenta 3 tipos de ComboBox.

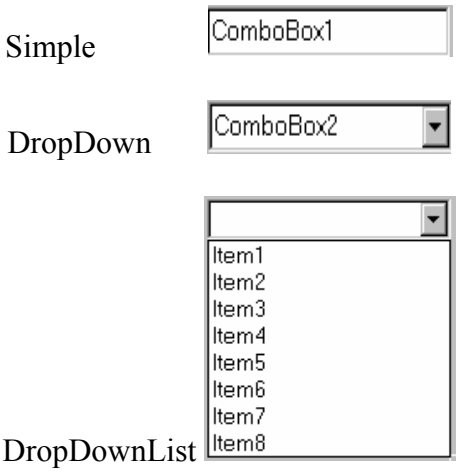


FIG 3.3.4.1 Tipos del Componente ComboBox.

Usted verá en su aplicación sólo una línea de edición, pero podrá desplegarla al momento de seleccionar su opción. Y se verá de la siguiente manera:

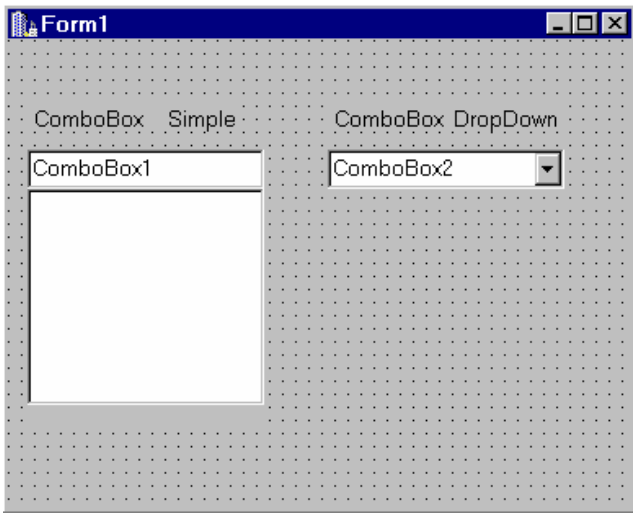


FIG 3.3.3.2 Componente ComboBox.

Experimente siguiendo los siguientes pasos:

1.- Elija del menú principal File ⇒ New Application.

2.- Si no esta presente el inspector de objetos presione la tecla de función F11.

3.- Elija de la barra de componentes el botón de caja de edición  (). y colóquelo en la forma.

4.- Elija de la barra de componentes el botón de *ComboBox* (

). y colóquelo en la forma. Cambie la propiedad de Style de este objeto con el inspector de objetos a la propiedad csDropDown.

5.- En la hoja de eventos haga doble clic en el evento *onDropDown* e introduzca el siguiente código.

```
ComboBox1->Items->Add(Edit1->Text);
```

6.- En la hoja de eventos del inspector de objetos seleccione para este mismo objeto el evento *OnDblClick* y teclee el siguiente código:

```
ComboBox1->Items->Clear();
```

7.- Ejecute su programa y teclee cualquier palabra en la caja de edición y luego con el mouse seleccione la flecha hacia debajo de la caja de selección y verá que la palabra que tecleó en la caja de edición se copiará en la caja de selección.

3.3.5 USO DEL COMPONENTE SCROLLBAR.

El componente ScrollBar se usa la mayor parte de las veces combinado con otros controles. Las propiedades mas importantes de este componente son:

Kind	Permite dos opciones para la barra, vertical u horizontal.
Max - Min.	Número mínimo o máximo de pasos en los que está dividida la barra (en una posición relativa).
Position	Toma el valor actual del numero de pasos de la barra según se modifique por medio de sus botones de incremento y decremento. .
Large Change	Determina la distancia que la barra se moverá cuando el usuario mueva sus controles.
SmallChange	Numero de movimientos que hará del extremo de la barra (número que se definió en Min.) al otro extremo de la barra hasta llegar al número que se definió en Max.

Siga esta nueva aplicación para ejemplificar su uso.

1.- Elija del menú principal File ⇒ New application.

2.- Si no esta presente el inspector de objetos presione la tecla de función F11.

3.- Seleccione ScrollBarr de la paleta de componentes Standard (

)y colóquela en la Forma.

4.- Cambie las siguientes propiedades de la barra con el inspector de objetos.

Propiedad	Valor
Large Change	32
Max	255
SmallChange	16

5.- Copie la barra de la siguiente forma seleccione del menú principal Edit ⇒ Copy.

6.- Seleccione la forma haciendo clic en ella.

7.- Pegue en su forma 2 copias de la barra que capturó en el paso 5 de la siguiente manera: Edit ⇒ Paste. Edit ⇒ Paste.

8.- Seleccione de la paleta de componentes un objeto panel (

) y colóquelo en la forma. Borre el nombre de Panel1 en la propiedad Caption.

9.- Seleccione de la paleta de componentes un objeto Label (

) y colóquelo en la forma, haga tres copias de ella, cámbiele nombres(Label1 = Azul, Label2 = Verde, Label3 = Rojo) y acomódelos de la siguiente manera.

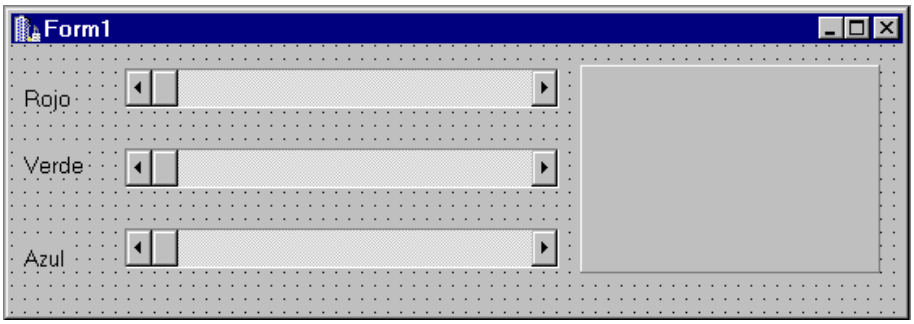


FIG 3.3.5.1 Acomodo de los componentes ScrollBar para una aplicación de color.

10.- Seleccione las tres barras oprimiendo la tecla Shift y haciendo clic en las tres barras sin soltar la tecla Shift.

11.- En la hoja de eventos del inspector de objetos haga doble clic en el evento OnChange y escriba el siguiente código.

```
Panel1->Color = RGB(ScrollBar1 -> Position,
                    ScrollBar2 -> Position,
                    ScrollBar3 -> Position );
```

```
Label1->Caption = IntToStr(ScrollBar1 -> Position);
Label2->Caption = IntToStr(ScrollBar2 -> Position);
Label3->Caption = IntToStr(ScrollBar3 -> Position);
```

12.- Ejecute su programa y juegue con los controles.

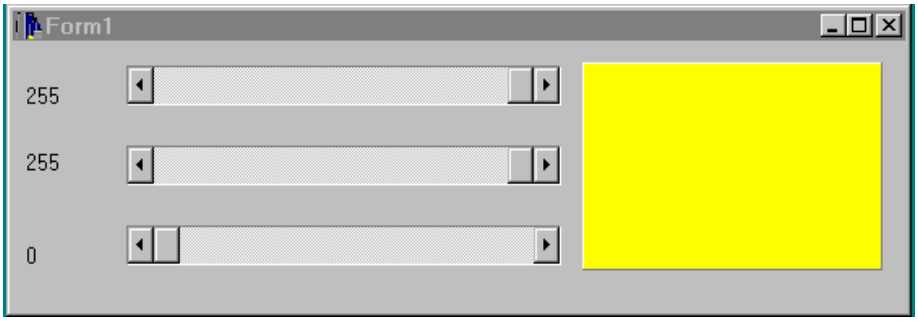


FIG 3.3.5.1 Resultado de la ejecución de la aplicación de color.

3.3.6 USO DEL COMPONENTE BITBTN.

Usar el componente BitBtn provee de una mejor apariencia a sus aplicaciones, para anexar una imagen a sus botones, no tiene que dibujar nada, amenos que lo desee, ya que C++ Builder proporciona un biblioteca de imágenes en el directorio **cbuilder\images\buttons**.

La forma de poner una imagen en un botón se muestra en el siguiente ejemplo.

1.- Elija del menú principal File ⇒ New Application.

2.- En la hoja Additional de la paleta de componentes, seleccione el objeto BitBtn (

) y colóquelo en su forma.

3.- Con el inspector de objetos haga doble clic en la propiedad Glyph, y aparecerá un editor de imágenes en el que podrá seleccionar la imagen que desee en su botón.

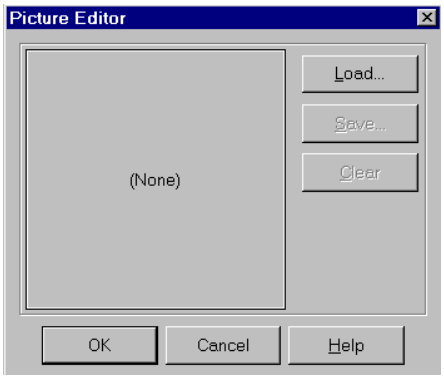


FIG 3.3.6.1 Interfase de Imágenes del componente BitBtn.

4.- Haga clic en el botón Load, para seleccionar una imagen, si no se encuentra en ese directorio puede buscarla en esta misma opción.

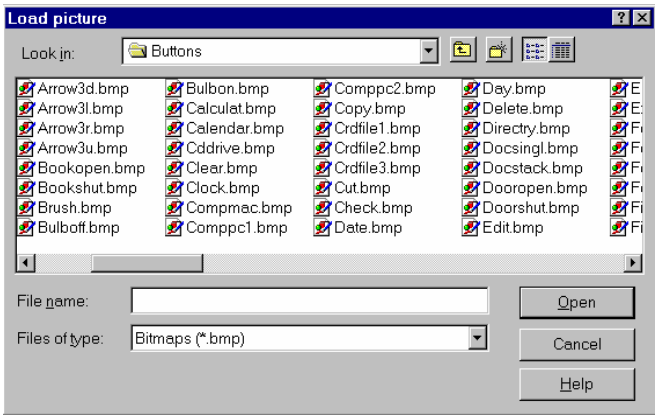


FIG. 3.3.6.2 Directorio de Imágenes de C++ Builder.

5.- Seleccione la imagen deseada y oprima el botón Open. Y en el editor de imágenes el botón OK.

Puede repetir esta acción y poner barios BitBtn en su aplicación.

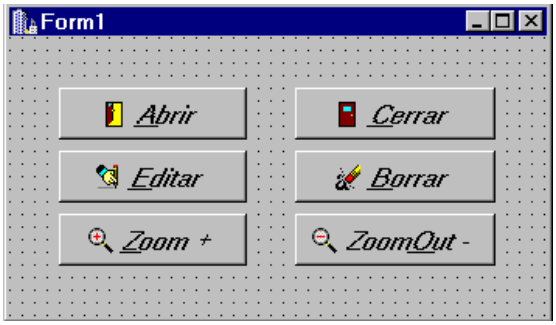


FIG. 3.3.6.3 Uso de imágenes en componentes BitBtn.

3.3.7 USO DE BOTONES ACELERADORES.

Los botones aceleradores proveen al usuario de una forma alternativa de ejecutar acciones que pueden de otra manera ser accesados por la barra de menús. Muchas aplicaciones Windows hacen uso de botones aceleradores.

Siga los siguientes pasos para ver un ejemplo.

1.- Elija del menú principal File ⇒ New Application.

2.- Seleccione de la paleta de componentes, en la hoja additional, el objeto SpeedButton () y colóquelo en la forma.

3.- Repita la acción 3 veces mas.

4.- Utilice el inspector de objetos para colocar un titulo (Caption) en lo botones aceleradores como se muestra en la figura.

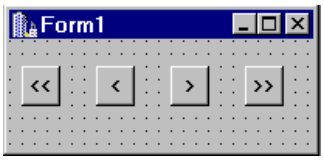


FIG. 3.3.7.1 Botones aceleradores.

5.- Y en la propiedad GroupIndex coloque el número 1. Lo que indicará que los botones trabajarán como grupo y cuando uno sea seleccionado permanecerá presionado hasta que otro botón del grupo sea seleccionado, presentando una elección exclusiva para el usuario.

6.- Ejecute su aplicación y vea como trabajan los botones aceleradores.

FIG. 3.3.7.2 Ejecución del programa Botones aceleradores.

3.3.8 USO DE CAJAS DE VERIFICACION.

Cuando dentro de sus aplicaciones necesite verificar de entre varias opciones; las cajas de verificación son un componente que puede ofrecerle grandes ventajas y una excelente presentación. Las cajas de selección las puede usar cuando necesite seleccionar entre varias opciones un grupo de ellas o su totalidad.
Usted puede seleccionar este componente de la paleta de componentes Standard, su uso comúnmente se combina con botones, así que como ejemplo puede utilizar el programa que construyó en la sección de botones aceleradores e incluirle los siguientes pasos.

1.- Seleccione de la hoja Standard de componentes el objeto CheckBox (

) y coloque 4 copias de este objeto en su forma.

2.- Con el inspector de objetos cambie los títulos de las cajas como se muestra en la figura:

FIG. 3.3.8.1 Botones aceleradores y cajas de verificación.

3.- Haga doble clic en cada un de los botones aceleradores e incluya el siguiente código en cada función como se muestra:

```
void __fastcall TForm1::SpeedButton2Click(TObject *Sender)
{
CheckBox2->Checked = true;
CheckBox1->Checked = false;
CheckBox3->Checked = false;
CheckBox4->Checked = false;
}
void __fastcall TForm1::SpeedButton3Click(TObject *Sender)
{
 CheckBox3->Checked = true;
 CheckBox1->Checked = false;
 CheckBox2->Checked = false;
 CheckBox4->Checked = false;
}
void __fastcall TForm1::SpeedButton4Click(TObject *Sender)
{
CheckBox4->Checked = true;
CheckBox1->Checked = false;
CheckBox2->Checked = false;
CheckBox3->Checked = false;
}
void __fastcall TForm1::SpeedButton1Click(TObject *Sender)
{
 CheckBox1->Checked = true;
 CheckBox2->Checked = false;
 CheckBox3->Checked = false;
 CheckBox4->Checked = false;
}
```

4.- Ejecute su programa y oprima los botones aceleradores, y observe su acción sobre las cajas de verificación.

FIG. 3.3.8.2 Ejecución del programa verificación acelerada.

3.3.9 USO DE BOTONES CIRCULARES (RADIO BUTTON).

Los botones circulares trabajan semejante a las cajas de verificación, solo que en estos solo puedes tener una sola selección y en las cajas de verificación puedes tener todas las cajas seleccionadas.
Para ejemplificar su uso siga los siguientes pasos donde también utilizará el componente RadioGroup de la hoja Standard de componentes, ya que su uso es prácticamente igual, la diferencia estriba en que el RadioButton es un solo componente, y el Radiogroup pueden ser varios.

1.- Elija del menú principal File ⇒ New Application.

2.- Seleccione de la paleta de componentes, en la hoja Standard, el objeto RadioButton  (RadioButton), y haga 5 copias de el en la forma.

3.- Con el inspector de objetos cambie la propiedad de titulo (Caption), como se muestra a continuación.

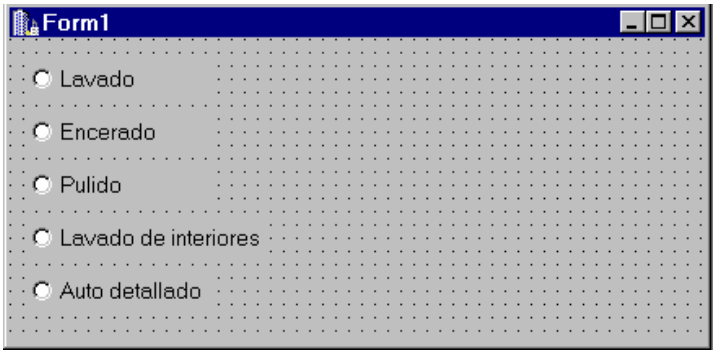


FIG. 3.3.9.1 Acomodo de componentes RadioButton.

4.- Seleccione de la paleta de componentes, en la hoja Standard, el objeto RadioGroup  (RadioGroup), y colóquelo en la forma. Cambie el Titulo a “Auto Detallado” con el inspector de objetos en la propiedad Caption.

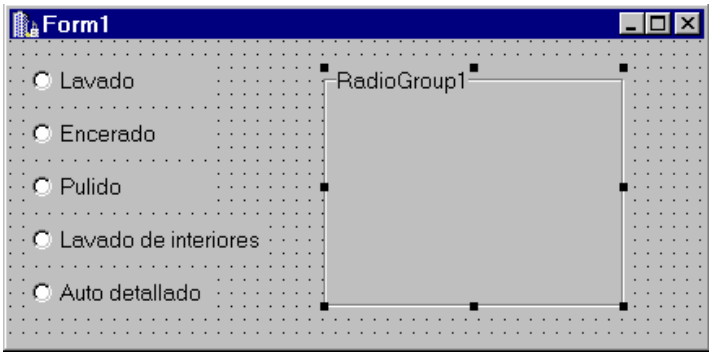


FIG. 3.3.9.2 Acomodo del componente RadioGroup.

5.- En el inspector de objetos para este componente, seleccione la propiedad Items y haga doble clic, en ese momento aparecerá una ventana de edición; en ella teclee los mismos nombres que en los RadioButtons como se muestra en la Figura.

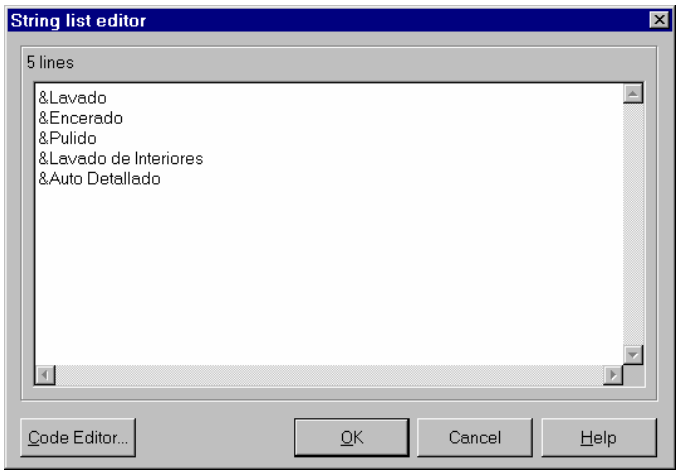


FIG. 3.3.9.3 Editor para el componente RadioGroup.

6.- Seleccione el botón Ok. Para cerrar esta ventana de edición y obtendrá el siguiente resultado:

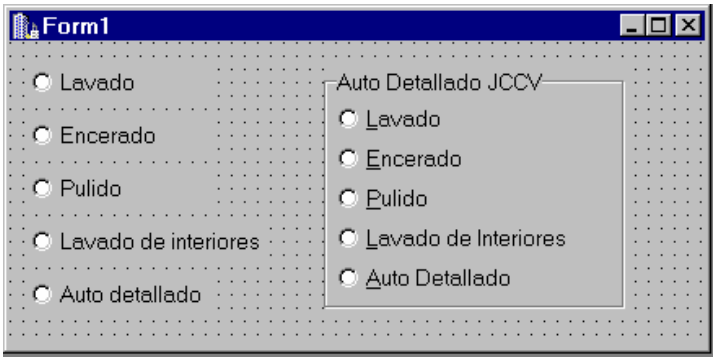


FIG. 3.3.9.4 Acomodo de componentes RadioButton y RadioGroup.

7.- Presione la tecla de función F9 para ejecutar su aplicación y observe los resultados.

3.3.10 USO DE MASCARAS EN LA EDICIÓN (MASKEDIT).

Este componente es usado para evitar que el usuario introduzca datos distintos a lo validos, ya que solo el tipo de dato y en la forma en que se indique, es lo que el usuario podrá introducir en este componente.

Para observar mejor su uso, siga los siguientes pasos:

1.- Elija del menú principal File ⇒ New Application.

2.- En la hoja Additional de la paleta de componentes, seleccione el objeto MaskEdit  (), y colóquelo en la forma.

3.- Con el inspector de objetos, seleccione la propiedad EditMask haciendo doble clic, en ese instante aparecerá un editor de tipos en el que existe una variedad de tipos de datos definidos de los que puede seleccionar posteriormente..

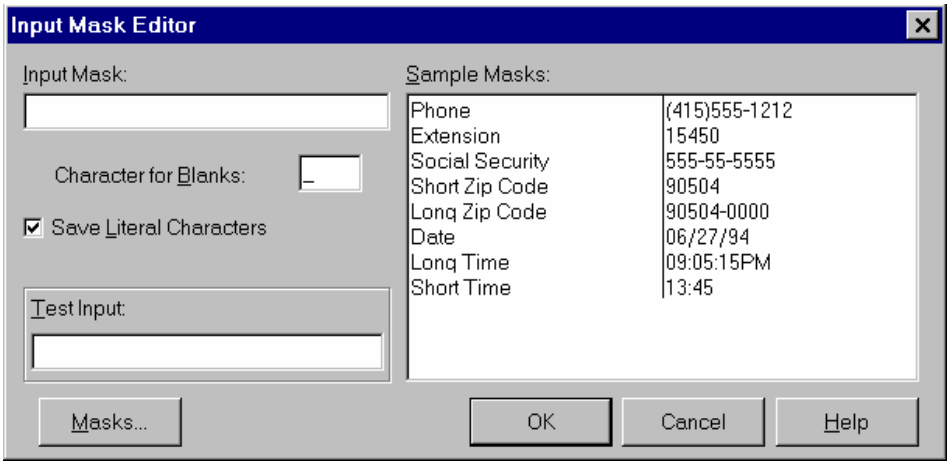


FIG. 3.3.10.1 Editor de mascaras.

4.- En la caja de dialogo(Input Mask) introduzca el siguiente código:
LLLLL00000 y seleccione el botón OK.

5.- Ejecute su programa y trate de introducir texto y números, observe que pasa.

Usted puede elaborar su propia mascara o secuencia de caracteres siguiendo las indicaciones de la siguiente tabla:

Caractér	Usado para enmascarar...
A	Permite introducir letras o números (ejemplo A ⇒ Z, a ⇒ z, 0 ⇒ 9)
L	Permite introducir letras (ejemplo A ⇒ Z, a ⇒ z)
0	Permite caracteres numéricos solamente
#	Permite introducir caracteres numéricos signados (+) y (-)
_(Guión Bajo)	Inserta un espacio en esa posición.
>	Convierte todos los caracteres a Mayúscula hasta que encuentra el signo <.
<	Convierte todos los caracteres a Minúscula hasta que encuentra el signo >.
!	Remueve los caracteres iniciales en blanco (espacios) que se encuentre en la mascara. Ejm. “ 12345”. Remueve los espacios iniciales y queda así “12345”.

3.3.11 USO DE COMPONENTE DE APARIENCIA (SHAPE).

El componente Shape por si solo no es muy útil, pero combinado con otros componentes y mucho ingenio, se pueden lograr resultados excelentes.
Este componente se encuentra en la hoja Additional de la paleta de componentes.

Vea el uso de sus propiedades con el inspector de objetos de este componente.

3.3.11.1 LA PROPIEDAD SHAPE.

Esta propiedad determina cual será la configuración o forma del objeto, usted tiene para elegir entre 6 formas que puede tomar su objeto.

Seleccione de la paleta de componentes Additional un componente Shape (

) y colóquelo en su forma para observar como funciona esta propiedad.

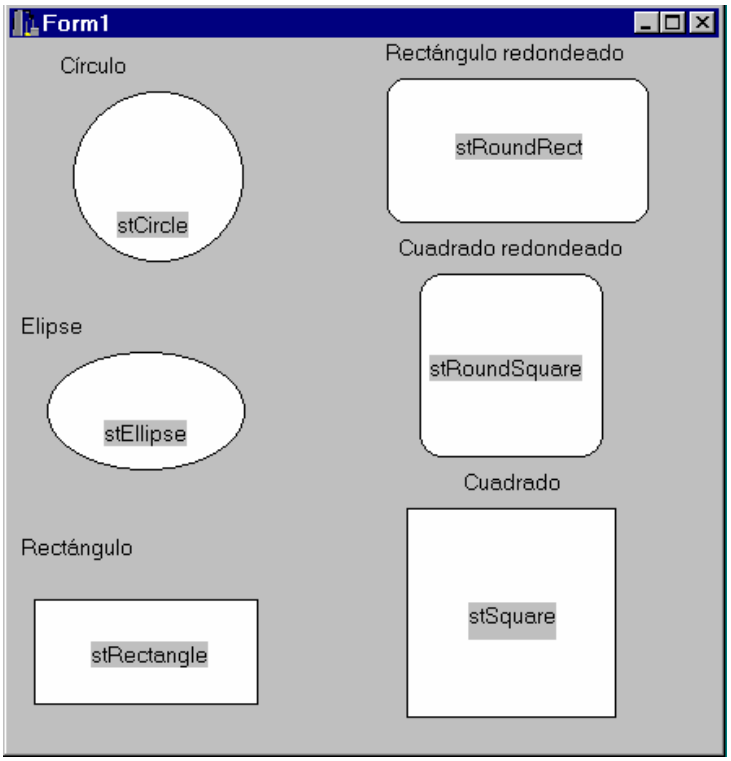


FIG. 3.3.11.1 Objeto Shape en sus seis distintas formas.

3.3.11.2 LAS PROPIEDADES HEIGHT Y WIDTH.

Estas propiedades sirven para ajustar el tamaño exacto de las figuras, o simplemente jale con su mouse la caja de contorno de su figura para ajustarla a su preferencia.

3.3.11.3 LAS PROPIEDADES PEN Y BRUSH.

Con estas propiedades usted determina la forma en que estarán dibujados los objetos shape que ha seleccionado.
Con la propiedad Pen determina la línea exterior del objeto.
Width $\Rightarrow$ determina la densidad de la línea exterior.
Color $\Rightarrow$ determina el color de la línea.
Style $\Rightarrow$ incluye un rango de valores para inicializar, línea sólida, punteada etc.

Con la propiedad Brush determina el interior del área del objeto.
Style $\Rightarrow$ indica como será el arrea interior, si sólida o achurada en alguna forma.
Color $\Rightarrow$ Será el color que lleve el relleno.

3.3.12 USO DE COMPONENTE IMAGE.

Donde quiera desplegar una imagen o un gráfico en su forma, lo puede hacer por medio de el componente Image.

Para ver como funciona este componente siga los siguientes pasos:

1.- De la paleta de componentes en la hoja Additional, seleccione el componente Image 

y colóquelo en su forma.

2.- Seleccione la propiedad ***Picture*** en el inspector de objetos de la Imagen haciendo doble clic para obtener la caja de diálogo del componente imagen.

3.- Seleccione el botón LOAD para buscar una imagen de algún directorio.

4.- Seleccione la imagen deseada y oprima el botón Ok.

Si selecciona del inspector de objetos la propiedad stretch = true, la imagen se adaptará al tamaño que tiene su objeto Image ya sea mayor o menor.

3.3.13 USO DEL COMPONENTE STRINGGRID.

El componente StringGrid al igual que DrawGrid, se utilizan para mostrar artículos del mismo tipo en un formato compacto de renglones y columnas.
StringGrid es como poner varios DrawGrid juntos, por lo que StringGrid es mucho mas potente, por lo que mostraré el uso de este componente y pocas son las variaciones que hacer para utilizar DrawGrid.

Siga los siguientes pasos para conocer su funcionamiento.

1.- Abra una nueva aplicación.

2.- Coloque un objeto panel de la paleta de componentes Standard en su forma, con el inspector de objetos, en la propiedad *Align*, seleccione alLeft y remueva el título Panel1 de la propiedad *Caption*..

3.- Con el objeto panel seleccionado elija de la paleta de componentes Standard un objeto Botón y colóquelo sobre el panel; modifique el título de Button a Cargar.

4.- En la paleta de componentes System seleccione un objeto DirectoryListBox  y colóquelo en el panel.

5.- En la paleta de componentes Additional seleccione un objeto StringGrid  y colóquelo en la forma principal, con el inspector de objetos cambie la propiedad *align* por alClient.
Y las siguientes propiedades según se muestra en la siguiente tabla.

PROPIEDAD	VALOR
ColCount	10
DefaultColWidth	80
DefaultRowHeight	40
FixedCols	0
FixedRows	0
RowCount	10

Con el inspector de objetos en la hoja de eventos haga doble clic en el evento *OnDrawCell*
E introduzca el siguiente código.

```
if( StringGrid1 -> Objects [ Col ] [ Row ] != 0 )
{
 StringGrid1 -> Canvas -> Draw (Rect.Left + 10, Rect.Top + 20, ( Graphics :: TGraphic* )
StringGrid1 -> Objects [ Col ] [ Row ] );
}
```

6.- Haga doble clic en el objeto botón y coloque el siguiente código:

```
int Col, Row;
ffblk fbFileData;
TPicture *picThis;

String strPath = DirectoryListBox1->Directory + "\\";
String strWildCard = "*.bmp";
String strTemp = strPath + strWildCard;
findfirst(strTemp.c_str(),&fbFileData, 0);

strTemp = strPath;
strTemp += fbFileData.ff_name;

for(Row=0; Row<16; Row++)
 {
  for(Col=0;Col<16; Col++)
```

```
  {
  StringGrid1->Cells[Col][Row] = fbFileData.ff_name;
  picThis = new TPicture;
  picThis->LoadFromFile(strTemp);
  StringGrid1->Objects[Col][Row] = (TObject*) picThis->Graphic;

  findnext(&fbFileData);
  strTemp = strPath;
  strTemp = fbFileData.ff_name;
  }
}
```

7.- Incluya las siguientes librerías de cabecera antes de #pragma.

```
#include <dir.h>
#include <dos.h>
```

8.- Ejecute su aplicación y verá el siguiente resultado.

FIG. 3.3.13.1 Uso del componente stringgrid.

Esta forma muestra cada una de los archivos de imagen que se pueden usar en los BtnBits antes visto.

3.3.14 USO DEL COMPONENTE MAIN MENU.

Los menús proveen de una forma rápida y estética de hacer cosas complicadas, evitando elaborar botones para ejecutar algunas cosas.
Los menús, se sitúan en la barra de título de la forma que generalmente hemos visto en la mayoría de los paquetes en que trabajamos.
Siga los siguientes pasos para ver como trabaja el Main Menú.

1.- Cree una nueva aplicación.

2.- En la paleta de componentes Standard seleccione un objeto Main Menú  y colóquelo en la forma.

3.- Haga doble clic en este objeto y se abrirá el diseñador de menús que tiene la siguiente apariencia.

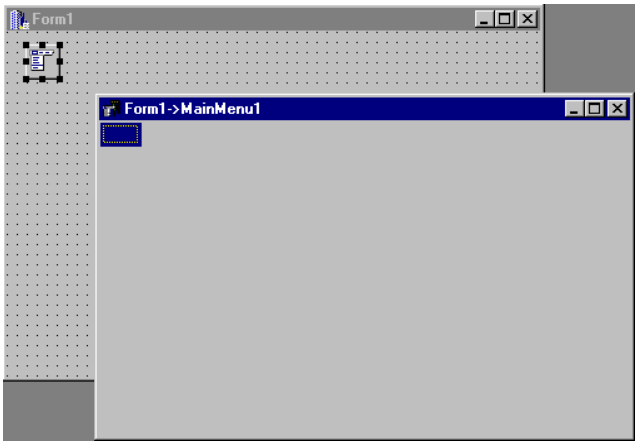


FIG. 3.3.14.1 llamada al diseñador de menús.

4.- Con el inspector de objetos ponga los títulos del menú modificando la propiedad caption de la siguiente manera.

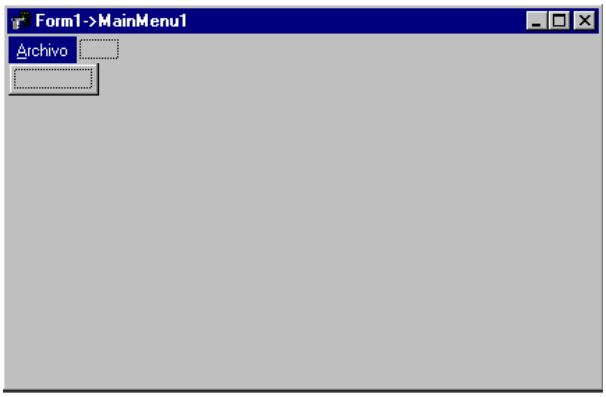


FIG. 3.3.14.2 Diseñador de menús añadiendo identificadores.

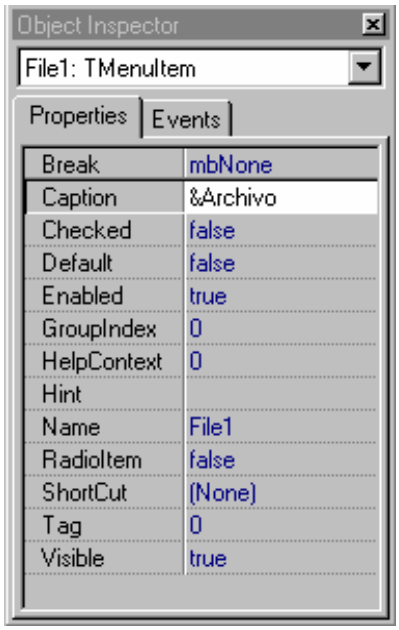


FIG. 3.3.14.3 Inspector de objetos en la creación de menús.

Y al hacer un clic bajo esta opción de menú que colocamos, podemos anexar mas opciones, ya sea como otra opción del menú (haciendo clic a la derecha) o como submenú (haciendo el clic hacia abajo).

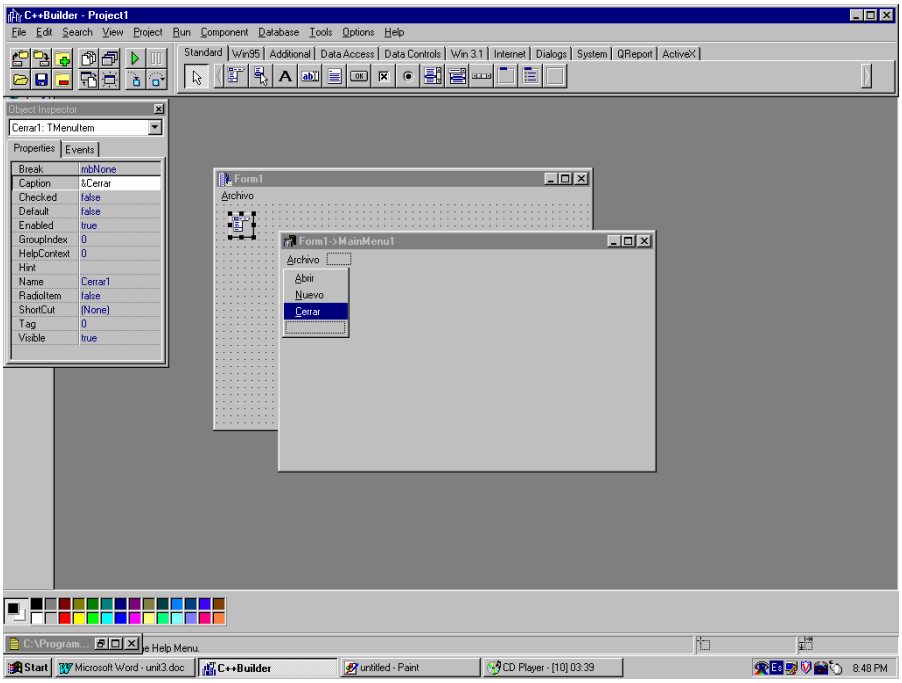


FIG. 3.3.14.4 Creación de menús.

5.- Cuando ya no necesite mas opciones en su menú, cierre el diseñador de menús.

3.3.14.1 AÑADIENDO ACCION A LOS MENUS.

Para que alguna opción del menú tenga su efecto necesita introducir el código de lo que debe realizar al seleccionar esa opción.

Haga doble clic en la opción, para tener acceso a su función OnClick, lo que inmediatamente llamará al editor de código y podrá teclear el código que tenga para esa acción.

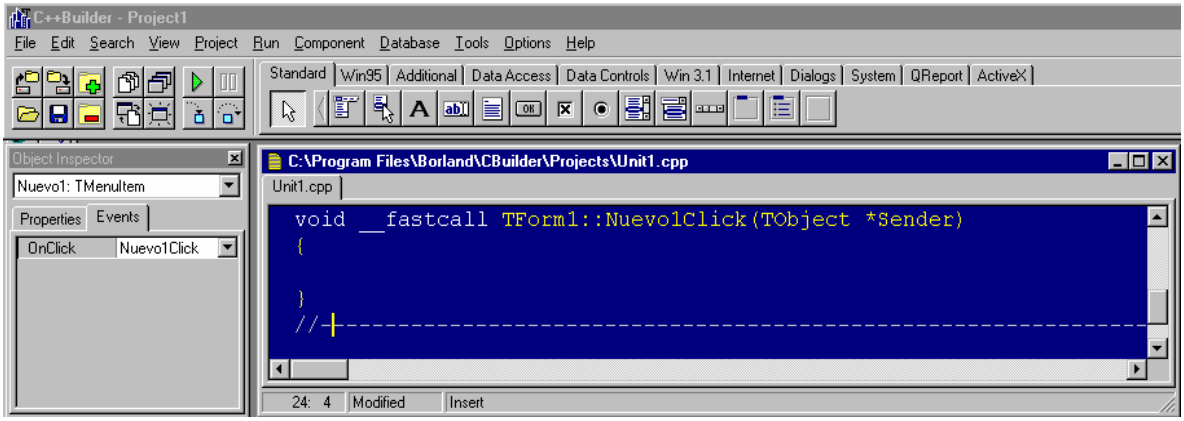


FIG. 3.3.14.1 Añadiendo acción a los menús.

3.3.14.2 AÑADIENDO TECLAS RAPIDAS Y ACCESOS DIRECTOS.

Las teclas rápidas se implantan de manera sencilla en un menú principal, solo hay que añadir un ampersand (&) antes de la palabra que será el identificador de la acción (esto en la propiedad caption del inspector de objetos).
Al hacer esto, tenemos implantada una llamada rápida a esa acción solo oprimiendo la tecla que esta subrayada.

Para implantar atajos (ShortCuts) es igualmente sencillo, si quisiera añadir un shortcut a la acción cerrar del menú anterior, solo tengo que seleccionar la acción cerrar, y con el inspector de objetos en la propiedad ShortCut escoger la combinación de teclas que mas convenga para la acción, en este caso podría poner la combinación Ctrl + E.

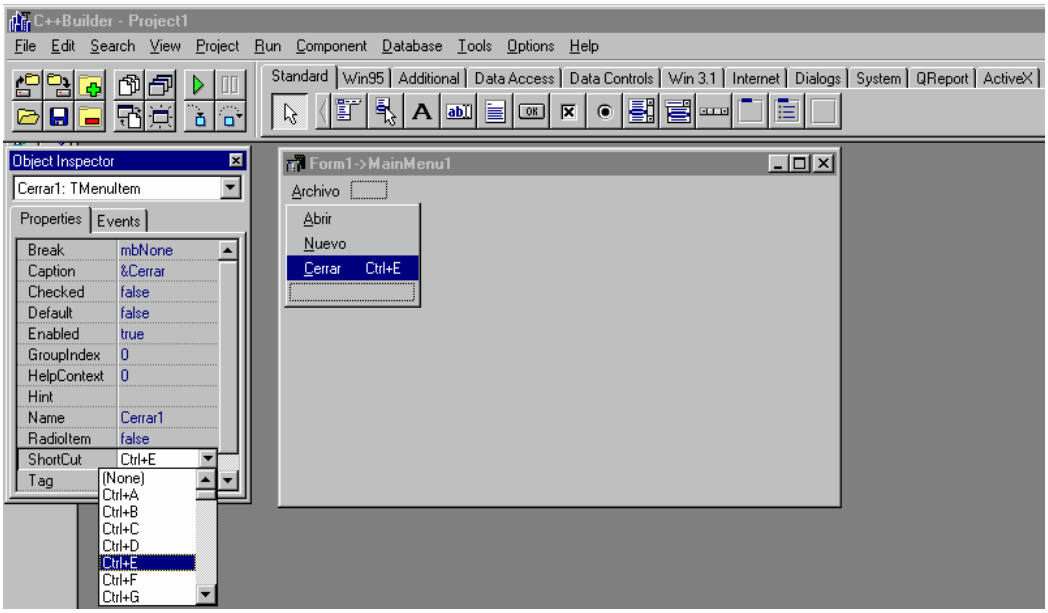


FIG. 3.3.14.2 Añadiendo teclas rápidas y Accesos directos.

3.3.14.3 CREANDO MENUS CON FORMATOS PREDEFINIDOS.

El formato predefinido, es una forma de estandarizar los menús, y es una forma muy rápida de tener menús completos en un instante, siga los pasos que se muestran para observar su acción.

1.- Cree una nueva aplicación.

2.- En la paleta de componentes Standard seleccione un objeto Main Menú  y colóquelo en la forma.

3.- Haga doble clic en este objeto y se abrirá el diseñador de menús, haga clic derecho en el diseñador y verá un menú de opciones,

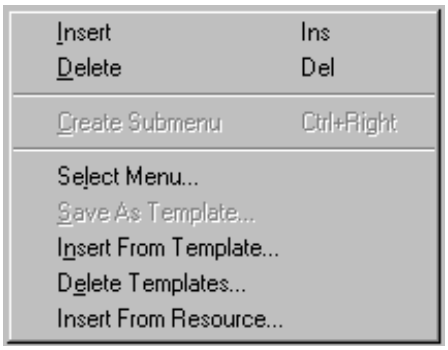


FIG. 3.3.14.3.1 Menú de opciones del diseñador de menús.

seleccione Insert From Template y seleccione la opción que sea útil para su aplicación.

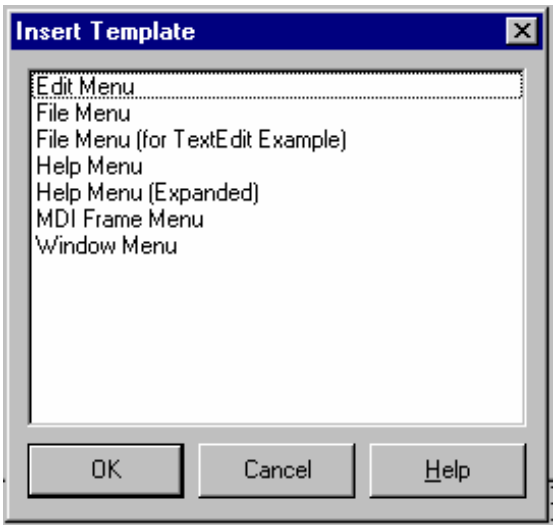


FIG. 3.3.14.3.2 Menú de formatos predeterminados.

la forma estandard para la opción File Menú es:

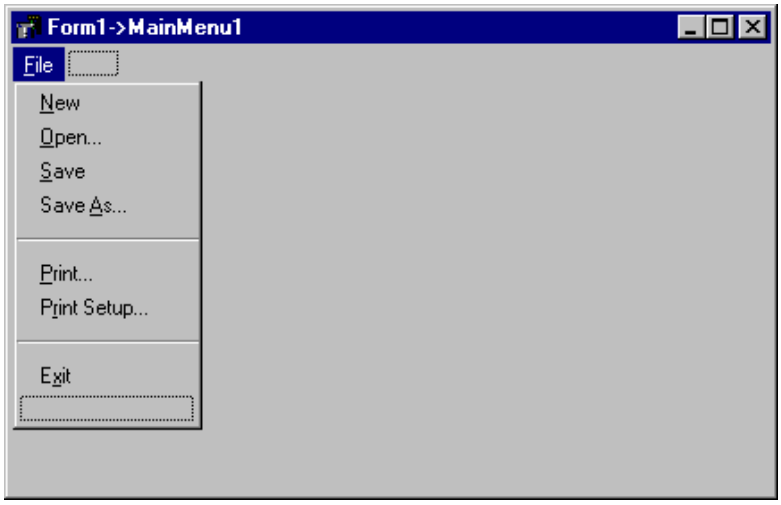


FIG. 3.3.14.3.3 Menú File de formatos predeterminados.

Para crear un menú para formato predefinido, primero diseñe su menú con las características que usted quiera, cuando halla concluido, haga clic con el botón derecho del mouse y seleccione Save As Template introduzca el nuevo nombre, y tendrá listo un modelo para sus menús.

Como ejemplo mostraré el anterior pero con las palabras en español.

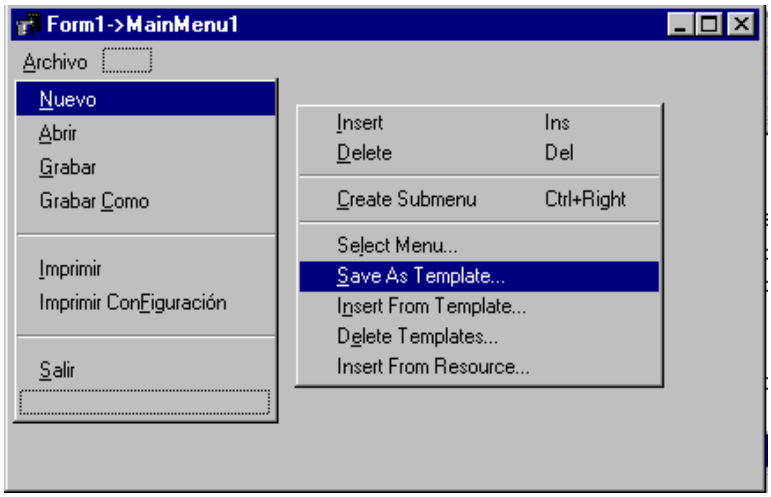


FIG. 3.3.14.3.4 Creación de formatos predefinidos.

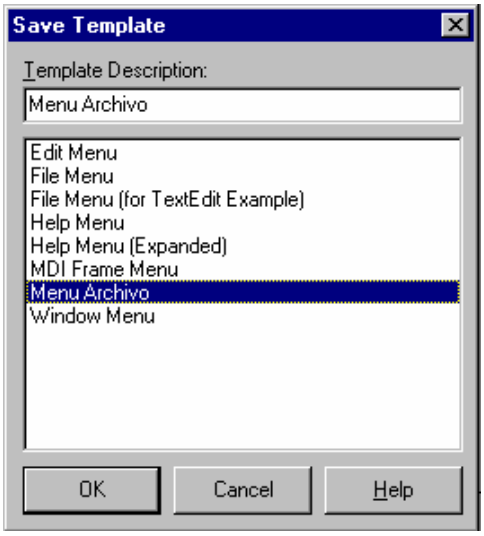


FIG. 3.3.14.3.5 Creación de formatos predefinidos.

3.3.15 USO DE BARRAS DE PROGRESO EN SUS APLICACIONES. (PROGRESSBAR)

El uso de este componente es muy importante, ya que muestra que efectivamente se está realizando una acción. Muchas veces el usuario al correr un programa no sabe si esta ejecutando o si ha ocurrido algún error y su equipo se encuentra bloqueado; con las barras de progreso, esta duda desaparece, ya que las barras de progreso muestran un avance determinado en la ejecución de alguna parte del programa. Esto lo mostramos sin mayor explicación en el ejemplo de la sección III.3.2.
Generalmente este componente se utiliza para: Cargar archivos, Grabar archivos, ejecutar cálculos, o procesos largos que no muestren resultados continuos.
Como otro ejemplo de este componente, siga los siguientes pasos.

1.- Cree una nueva aplicación.

2.- En la paleta de componentes Win95 seleccione un objeto ProgressBar  y colóquelo en su forma.

3.- En la paleta de componentes Standard seleccione un objeto Label y colóquelo en la forma.

4.- En la paleta de componentes Standard seleccione un objeto Button y colóquelo en la forma, con el inspector de objetos modifique la propiedad Caption a *AVANCE*,

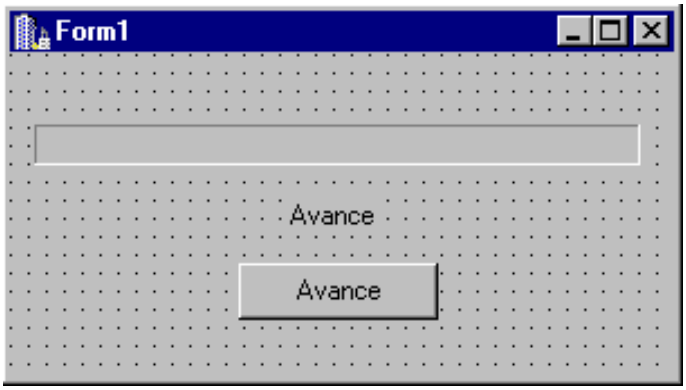


FIG. 3.3.15.1 Uso de barras de progreso en sus aplicaciones.

en la hoja de eventos haga doble clic en el evento OnClick e introduzca el siguiente código.

```
for(ProgressBar1->Position=0; ProgressBar1->Position=0< ProgressBar1->Max;
                                    ProgressBar1->Position++)
{
  Label1->Caption=string(ProgressBar1->Position) + “%”;
  Application->ProcessMessages();
  Sleep(50);
  }
Sleep(250);
ProgressBar1->Position=0;
Label1->Caption=String(ProgressBar1->Position) + "%";
```

5.- Presione la tecla de función F9 para ejecutar su programa, presione el botón avance y tendrá el siguiente resultado.

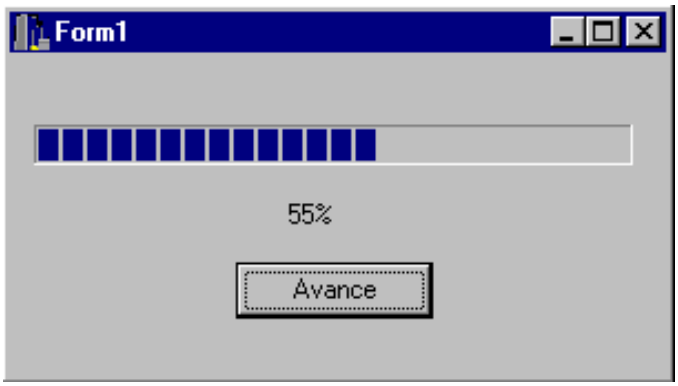


FIG. 3.3.15.2 Uso de barras de progreso en sus aplicaciones.

3.3.16 USO DE BARRAS DE ESTADO EN SUS APLICACIONES. (STATUSBAR)

Las barras de estado proveen al usuario de información de la aplicación que se está ejecutando. Usted puede crear mensajes separados y repartirlos en una misma barra de estado.

Complemente el programa anterior con los siguientes pasos:

1.- En la paleta de componentes Win95 seleccione un objeto StatusBar  y colóquelo en la forma, con el inspector de objetos haga doble clic en la propiedad *Panels* y automáticamente obtendrá la caja de dialogo *editor de panels* que tiene la siguiente apariencia.

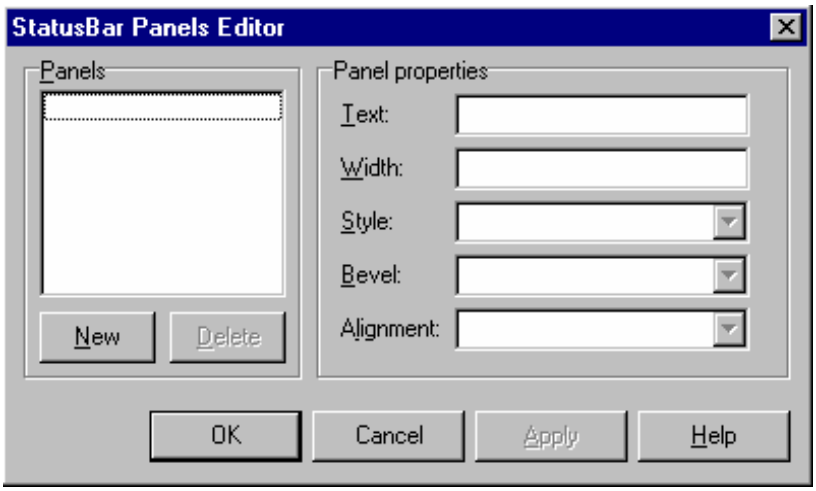


FIG. 3.3.16.1 Editor de panels de la barra de estado.

2.- Seleccione haciendo clic en el botón New para añadir un panel a la barra de estado.

3.- Escriba el siguiente texto.

QUE TAL HOY ES : en la ventana Text.
100 en la ventana Width.
Lowered en la ventana Bevel.
oprima el botón Apply.

4.- Seleccione haciendo clic en el botón New para añadir otro panel a la barra de estado.

5.- Escriba el siguiente texto.

MM/DD/YY en la ventana Text.
65 en la ventana Width.
Raised en la ventana Bevel.
Y oprima el botón Apply.

6.- Seleccione haciendo clic en el botón New para añadir otro panel a la barra de estado.

7.- Escriba el siguiente texto.

SON LAS : en la ventana Text.
100 en la ventana Width.
Lowered en la ventana Bevel.
Y oprima el botón Apply.

8.- Seleccione haciendo clic en el botón New para añadir otro panel a la barra de estado.

9.- Escriba el siguiente texto.

HH:MM:SS en la ventana Text.
65 en la ventana Width.
Raised en la ventana Bevel.
Oprima el botón Apply y el botón Ok.

10.- En la paleta de componentes System seleccione un objeto Timer  , con el inspector de objetos y tendrá su forma de la siguiente manera.

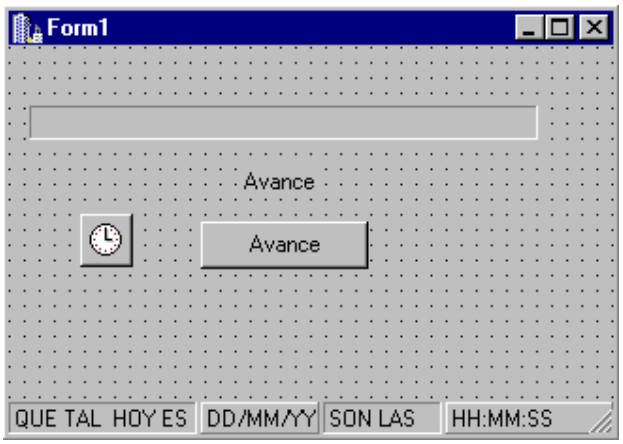


FIG. 3.3.16.2 Textos seleccionados y distribuidos en la barra de estado.

seleccione haciendo doble clic el evento OnTimer e introduzca el siguiente código.

```
StatusBar1->Panels->Items[3]->Text = Now().TimeString();
StatusBar1->Panels->Items[1]->Text = Now().DateString();
```

11.- Presione la Tecla F9 para ejecutar su programa.

Y obtendrá el siguiente resultado

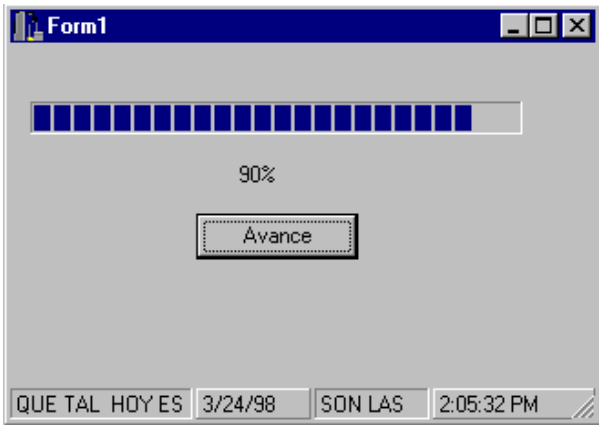


FIG. 3.3.16.3 Ejecución del programa de ejemplo de barras de progreso y estado..

3.3.17 USO DEL COMPONENTE RICHEDIT.

Con este componente se da al usuario acceso a todos los archivos de texto y en ellos podrá realizar modificaciones en el estilo del texto, además de poderlo imprimir.
Con los siguientes pasos podrá dotar su aplicación de un procesador de texto.

1.- Seleccione del menú principal File ⇒ New, y obtendrá una caja de diálogo, seleccione en la hoja de proyectos (projects) el icono de aplicación SDI y seleccione con un clic el botón Ok.

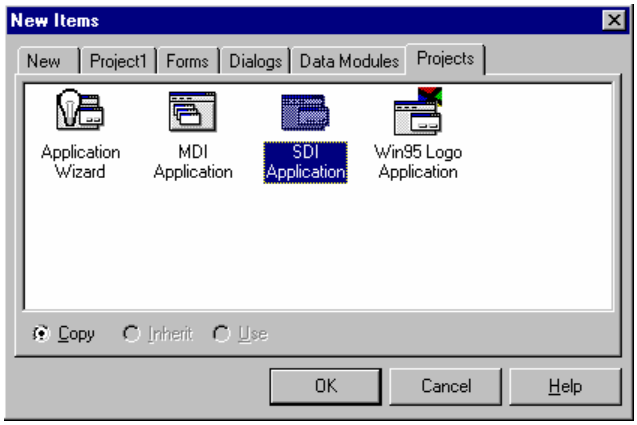


FIG. 3.3.17.1 Nuevos componentes.

2.- Haga clic en el botón Ok para dejar el directorio que marca por omisión, o seleccione un directorio en el que desee que guarde los archivos que generará este proyecto.

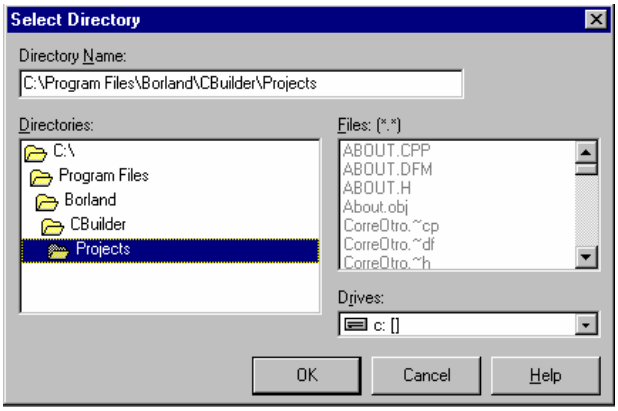


FIG. 3.3.17.2 Caja de dialogo para tomar un directorio destino.

3.- Después de esto obtendrá la siguiente forma:

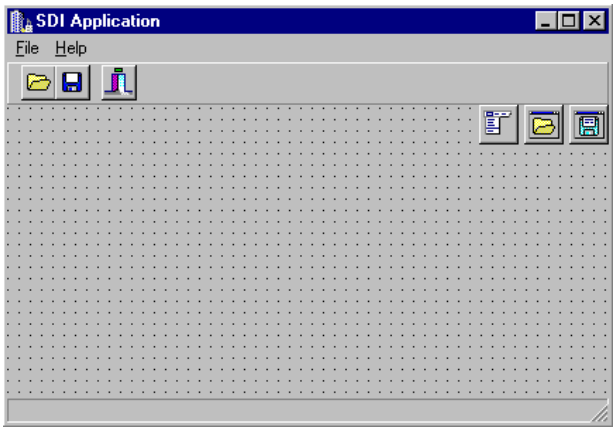


FIG. 3.3.17.3 Aplicación SDI, componentes ya integrados en este tipo de aplicación.

4.- De la paleta de componentes Win95 seleccione el objeto RichEdit  y colóquelo en la forma, con el inspector de objetos modifique la propiedad ***Align,*** ponga en ella **alClient.**

5.- De la paleta de componentes Additional seleccione un objeto SpeedButton y colóquelo en el panel en la parte superior de la forma, repita esta acción seis veces mas.

6.- Usando el inspector de objetos coloque los siguientes valores en las respectivas propiedades en cada uno de los 7 botones.

Propiedad **Name**	**GroupIndex**	**Caption**	**AllowAllUp settig**
BtnBold	1	N	true
BtnUnderline	2	S	true
BtnItalic	3	*I*	true
BtnLeftJust	4	|<	false
BtnCenter	4	=	false
BtnRightJust	4	>|	false
BtnPrint	0	Imprime	false

7.- Seleccione los tres primeros Botones (btnBold , btnUnderline, btnItalic), presionando la tecla Shift y haciendo clic en cada uno de ellos mientras se presiona la tecla; con el inspector de objetos haga doble clic en el evento OnClick e introduzca el siguiente código.

```
TFontStyles fsThis;
if(btnBold->Down)
fsThis<<fsBold;
if(btnUnderline->Down)
fsThis<<fsUnderline;
if(btnItalic->Down)
fsThis<<fsItalic;
RichEdit1->SelAttributes->Style = TFontStyles(fsThis);
```

8.- Seleccione los siguientes tres botones (btnLeftJust, btnCenter, btnRightJust) de igual manera que los anteriores, con el inspector de objetos, en la hoja de eventos haga doble clic en el evento OnClick e introduzca el siguiente código.

```
TAlignment taThis;

if(btnLeftJust->Down)
taThis = taLeftJustify;
if(btnCenter->Down)
taThis = taCenter;
if(btnRightJust->Down)
taThis = taRightJustify;
RichEdit1->Paragraph->Alignment = TAlignment(taThis);
```

9.- Seleccione el componente RichEdit y con el inspector de objetos en la hoja de eventos seleccione el evento OnSelectionChange e introduzca el siguiente código.

```
btnBold->Down = RichEdit1->SelAttributes->Style.Contains(fsBold);
btnUnderline->Down = RichEdit1->SelAttributes->Style.Contains(fsUnderline);
btnItalic->Down = RichEdit1->SelAttributes->Style.Contains(fsItalic);
switch(RichEdit1->Paragraph->Alignment)
{
case taLeftJustify :
 btnLeftJust->Down = true;break;
case taCenter :
 btnCenter->Down = true;break;
case taRightJustify :
 btnRightJust->Down = true;break;
}
```

10.- Seleccione el último botón *imprime* haciendo doble clic en él e introduce el siguiente código.

```
RichEdit1->Print(Caption);
```

11.- Del menú principal de la forma, seleccione con doble clic Archivo ⇒ Abrir e introduzca el siguiente código:

```
OpenDialog->Execute();
if(OpenDialog->FileName != "")
RichEdit1->Lines->LoadFromFile(OpenDialog->FileName);
```

12.- Del menú principal de la forma, seleccione con doble clic Archivo ⇒ Grabar e introduzca el siguiente código:

```
SaveDialog->Execute();
   if(SaveDialog->FileName != "")
  RichEdit1->Lines->SaveToFile(SaveDialog->FileName);
```

13.- Presione F9 para correr su programa.

3.3.18 USO DEL COMPONENTE IMAGELIST.

Como su nombre lo indica, el componente ImageList agrupa una colección de imágenes todas ellas del mismo tamaño, este componente no es visual, lo que quiere decir que usted no podrá ver las imágenes una vez agrupadas (Por lo menos no directamente con este componente).

Con los siguientes pasos podrá utilizar este componente.

1.- De la paleta de componentes en la hoja Win95 seleccione un objeto ImageList  y colóquelo en su forma.

2.- Haga clic con el botón derecho del mouse y obtendrá un menú de opciones elija la opción ImageList Edito y aparecerá la caja de dialogo llamada ImegeList Editor.

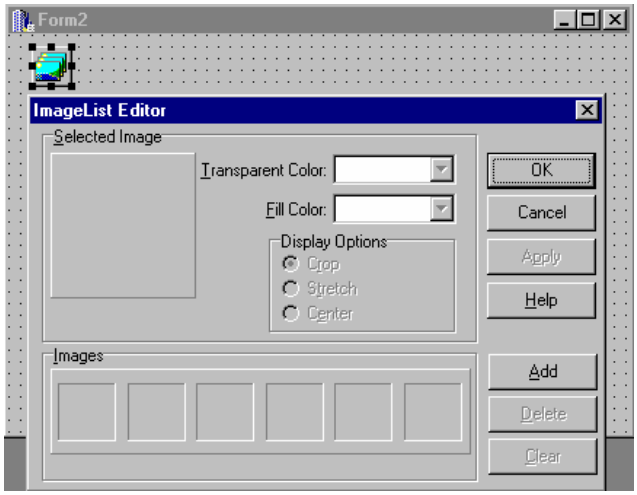


FIG. 3.3.18.1 Uso del componente imagelist.

3.- Seleccione el botón Add para añadir una imagen a la lista de imágenes.
Obtendrá una caja de dialogo para buscar el directorio donde están almacenados los archivos de las imágenes.

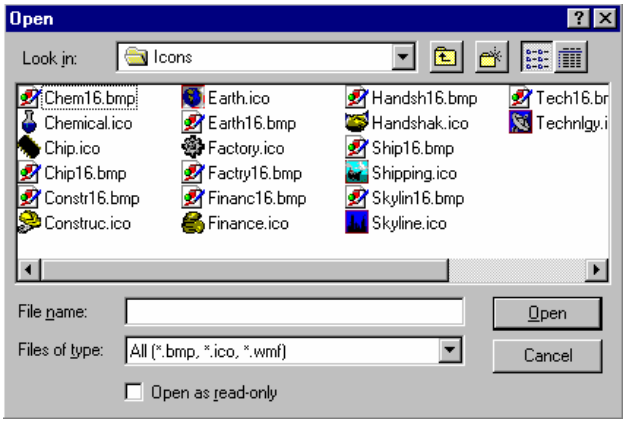


FIG. 3.3.18.2 Imágenes a seleccionar para tener en un arreglo de imágenes.

4.- Seleccione la imagen y oprima el botón Open; puede seleccionar cualquiera de las tres opciones que presenta para cada imagen. Crop (Cortar), Stretch (Ajustar), Center (Centrar),

5.- Puede repetir los pasos 3 y 4 cuantas veces quiera para tener una lista de imágenes disponibles en su ImageList.

3.3.19 USO DEL COMPONENTE LISTVIEW.

Este componente le da al usuario una manera de mostrar artículos del mismo tipo; como ejemplo, puede mostrar el grupo de imágenes que anteriormente guardó usando el componente ImageList y presentándolas con este componente.

Con los siguientes pasos podrá utilizar este componente.

1.- Inicie un nuevo proyecto.

2.- Utilice los pasos anteriores para tener una lista de imágenes.

3.- De la paleta de componentes Win95 seleccione un objeto ListView  y colóquelo en su forma, Con el inspector de objetos modifique la propiedad ***SmallImages*** colocando en ella **ImageList1**, También modifique la propiedad ***ViewStyle*** escogiendo de la lista disponible **vsList**.

4.- De la paleta de componentes Standard seleccione un objeto Panel y colóquelo en su forma, con el inspector de objetos modifique la propiedad ***Align*** escogiendo de la lista disponible **alBotton**.

5.- Seleccione el objeto ListView y con el inspector de objetos haga doble clic en la propiedad ***Items*** para abrir la caja de dialogo **Items Editor**.

6.- oprima el botón New, y escriba el texto que quiera que acompañe a la imagen guardada, oprima el numero de imagen a la se le asignará ese nombre.

7.- Repita los pasos 5 y 6 para cada una de las imágenes que halla almacenado.

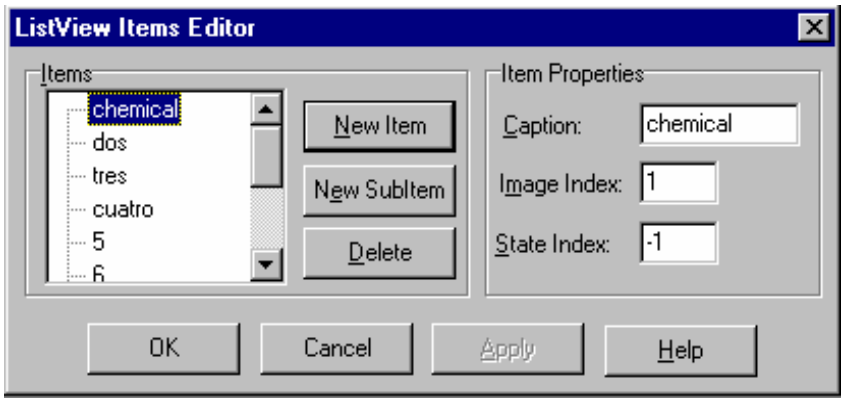


FIG. 3.3.19.1 Editor de artículos del objeto ListView.

8.- Seleccione el objeto ListView y oprima las teclas Ctrl + C, para hacer una copia de este objeto.

9.- Haga clic en cualquier parte de la forma y oprima las teclas Ctrl + V para pegar la copia hecha al objeto ListView, Repita esta acción una vez mas de manera que tenga en su forma 3 objetos ListView; acomódelos a lo largo de la forma.

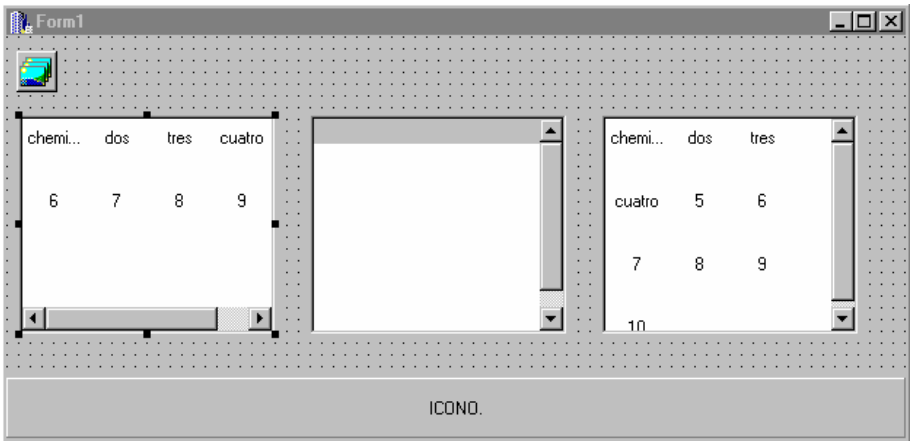


FIG. 3.3.19.2 Tres objetos ListView con diferentes parámetros en sus propiedades.

10.- Seleccione el componente que se encuentra en la parte central y con el inspector de objetos modifique la propiedad ***ViewStyle*** por la opción **vsReport,** en la propiedad ***columns*** haga doble clic para que aparezca el editor de columnas del componente ListView; oprima el botón New y escriba en la caja ***Caption*** la palabra **Artículos,** seleccione también el botón circular de nombre Header Widt, para colocar el título en la parte superior de la caja; Oprima el botón Ok para cerrar esta caja de dialogo.

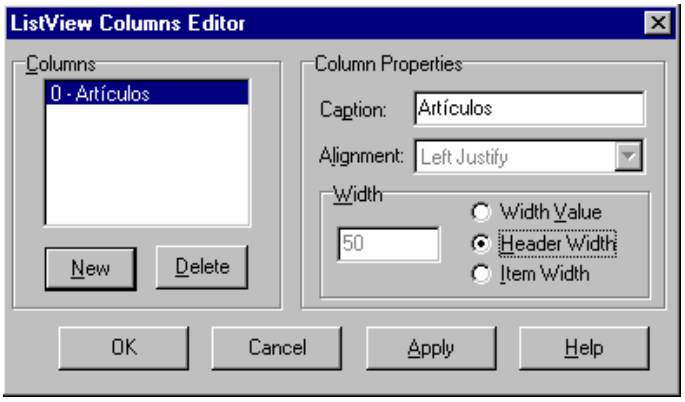


FIG. 3.3.19.3 Editor de columnas del objeto ListView.

11.- Seleccione el objeto ListView de la derecha y con el inspector de objetos modifique la propiedad ***ViewStyle*** por la opción **vsIcon,** También modifique la propiedad LargeImages y coloque en ella el texto ImageList1.

12.- Seleccione los tres componentes ListView oprimiendo la tecla Shift y haciendo clic en cada uno de ellos, con el inspector de objetos en la hoja de eventos haga doble clic en el evento OnChange y coloque el siguiente código.

```
Panel1->Caption = Item->Caption;
```

13.- Presione la tecla F9 para ejecutar su programa, obtendrá un resultado parecido al siguiente:

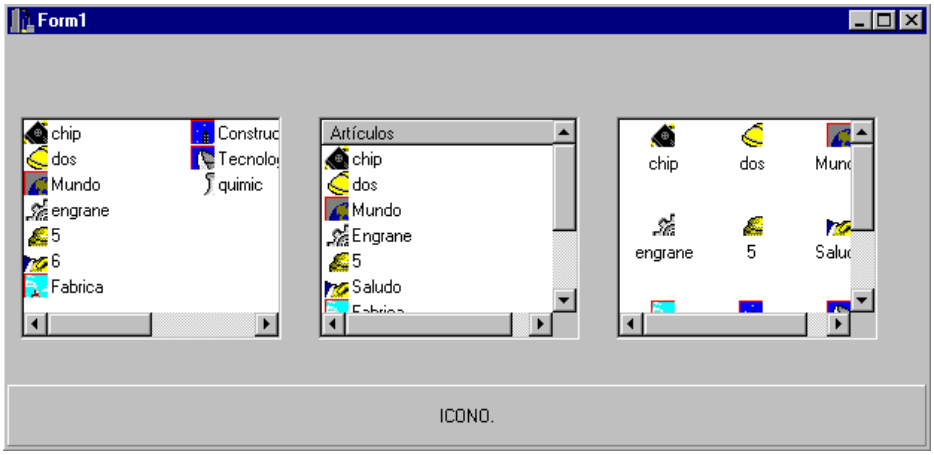


FIG. 3.3.19.4 Resultado de la ejecución del ejemplo del uso del componente ListView.

3.3.20 USO DEL COMPONENTE PAGECONTROL.

Este componente proporciona de una manera de tener múltiples hojas de trabajo y puede cambiar de hoja haciendo un simple clic en la cejilla de la hoja a la que se quiera accesar. Con los siguientes pasos podrá ver un sencillo ejemplo del uso de PageControl.

1.- Abra una nueva aplicación.

2.- En la paleta de componentes Win95 seleccione un objeto PageControl  ; al colocarlo en la forma lucirá como un panel; con el inspector de objetos modifique la propiedad ***Align*** por la opción **alClient**.

3.- Haga clic con el botón derecho del mouse y seleccione la opción New Page y una cejilla con la leyenda TabSheet1 aparecerá; con el inspector de objetos cambie la propiedad ***Caption*** por el texto **Hoja 1**, y repita la acción 2 veces.

4.- Seleccione la Hoja 1 y coloque en ella un objeto Label de la paleta de componentes Standard y colóquelo en la hoja, cambie la propiedad Caption por el texto “En la hoja 1”.

5.- Coloque un Objeto Button de la paleta de componentes Standard en la hoja Cambie la porpiedad Caption por el texto “Boton 1” ; haga doble clic en el botón e introduzca el siguiente código:

Label1->Caption = “Ya se oprimió el botón en la primer hoja .”;

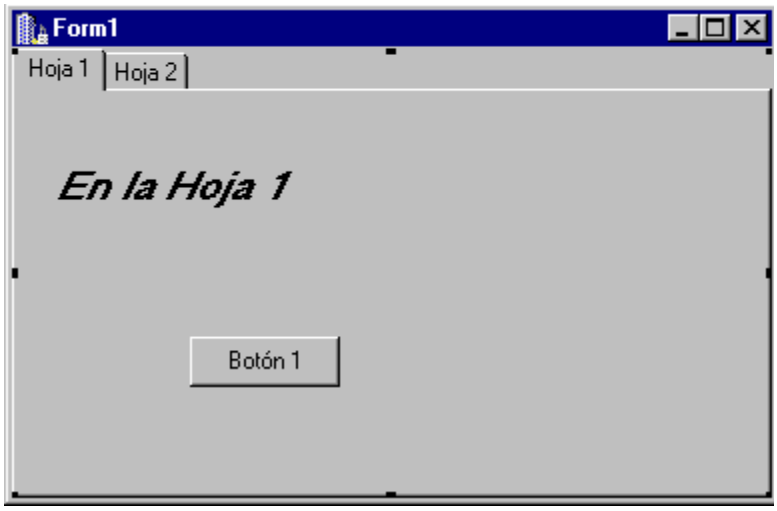


FIG. 3.3.20.1 Uso del componente pagecontrol.

6.- Seleccione la Hoja 2 y coloque en ella un objeto Editl de la paleta de componentes Standard y colóquelo en la hoja, cambie la propiedad text por el texto “Línea de copia”.

7.- Coloque un Objeto Button de la paleta de componentes Standard en la hoja Cambie la porpiedad Caption por el texto “Boton 2” ; haga doble clic en el botón e introduzca el siguiente código:

```
Edit1->Text = Label1->Caption;
```

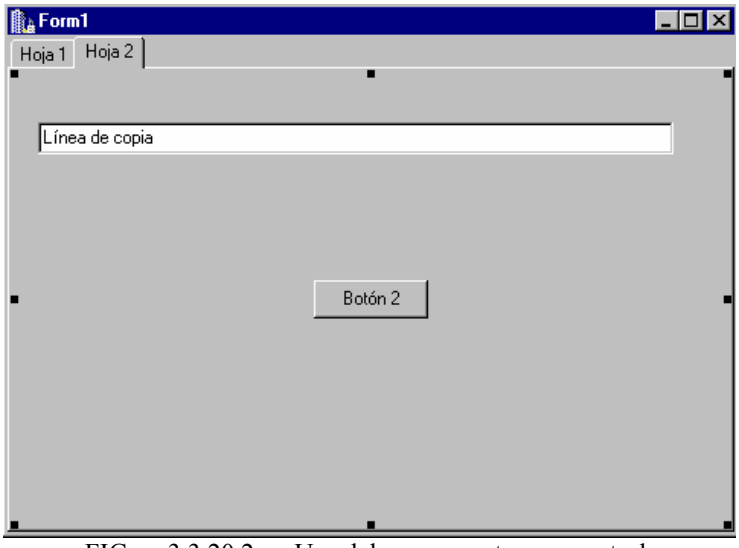


FIG. 3.3.20.2 Uso del componente pagecontrol.

8.- Presione la tecla F9 para ejecutar su programa.

3.3.21 USO DEL COMPONENTE OPENDIALOG.

La mayoría de la aplicaciones windows usan un estandar para abrir archivos, C++ Builder le ofrece una manera sencilla de apegarse a estos estandares.

Con los siguientes pasos podrá añadir una forma sencilla de abrir archivos.

1.- Seleccione un objeto Panel de la paleta de componentes Standard y elimine el texto que tiene en Caption para eliminar el titulo.

2.- Seleccione un objeto OpenDialog  de la paleta de componentes Dialogs y colóquelo en el panel, Modifique la propiedad ***DefaultExt*** a la opción **txt**, en la propiedad ***Filter***, ponga.

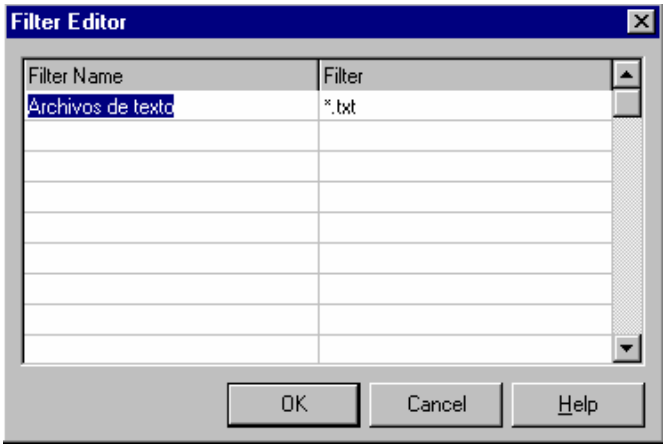


FIG. 3.3.21.1 Editor de archivos del componente OpenDialog**.**

3.- De la paleta de componentes Standard seleccione un objeto Memo y colóquelo en la forma, modifique la propiedad Align y seleccione alClient.

4.- De la paleta de componentes Standard seleccione un objeto Button y colóquelo en el panel, modifique la propiedad Caption poniendo el texto “Abrir”, en la hoja de eventos haga doble clic en el evento OnClick e introduzca el siguiente código.

```
if(OpenDialog1->Execute())
Memo1->Lines->LoadFromFile(OpenDialog1->FileName);
```

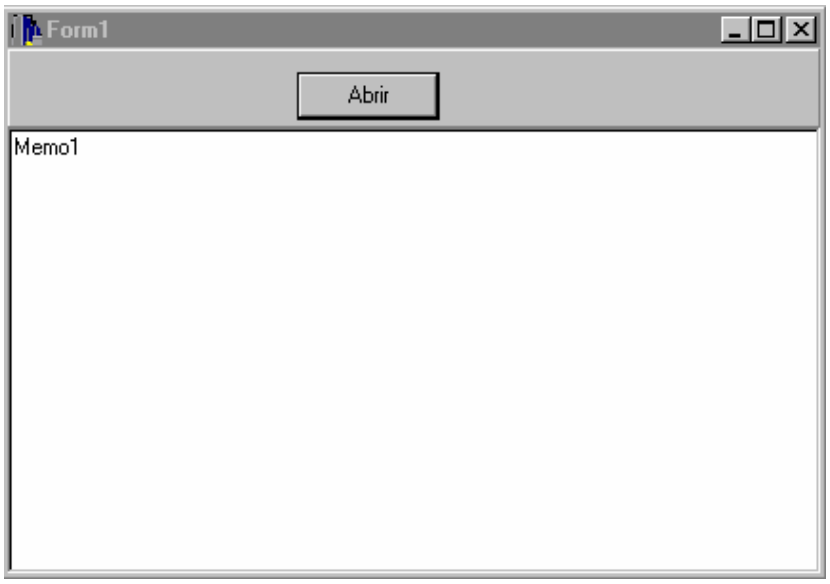


FIG. 3.3.21.2 Aplicación OpenDialog.

5.- Presione la tecla F9 para ejecutar su aplicación, presione el botón abrir para ver la acción de este componente.

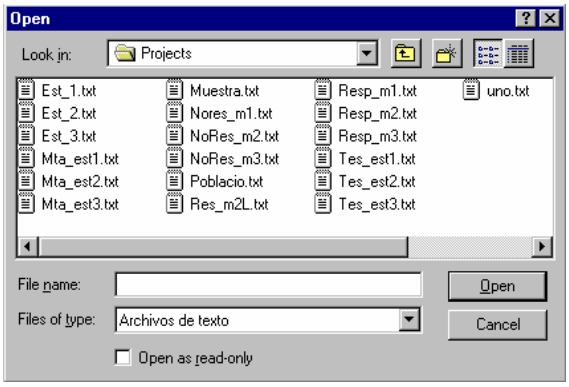


FIG. 3.3.21.3 Busqueda de archivos *.txt.

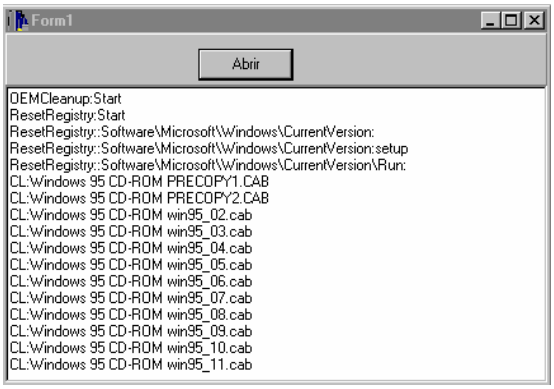


FIG. 3.3.21.4 Muestra del archivo seleccionado.

3.3.22 USO DEL COMPONENTE SAVEDIALOG.

El componente SaveDialog casi siempre es el compañero del componente anterior, esto quiere decir que la mayor parte de las veces que se abra un archivo, se querrán guardar o salvar los cambios hechos en ese archivo.

Con los siguientes pasos usted podrá complementar el ejemplo anterior en el que se abre un archivo de texto; ahora, se podrá guardar con otro nombre, con el mismo, en el mismo directorio, o en otro que se seleccione.

1.- De la paleta de componentes en la hoja Dialogs, seleccione el componente SaveDialog  , colóquelo en el panel del ejemplo anterior; usando el inspector de objetos cambie las propiedades de la siguiente forma:

Propiedad	**Setting**
DefaultExt	Txt
Filter	Text Files, *.txt
Options ofOverwrite Prompt	True
Title	Grabar archivo.

2.- De la paleta de componentes Standard; seleccione un objeto Button, y colóquelo en el panel, con el inspector de objetos cambie la propiedad Caption por el texto “Grabar”. En la hoja de eventos seleccione con doble clic el evento OnClick e introduzca el siguiente código.

```
If(SaveDialog->Execute())
Memo1->Lines->SaveToFile(SaveDialog1->Filename);
```

FIG. 3.3.22.1 Uso del componente SaveDialog.

3.- Presione F9 para ejecutar su programa; tendrá el resultado anterior, podrá abrir cualquier archivo de texto de cualquier directorio y además podrá grabar el archivo abierto con el mismo nombre, en el mismo directorio o con diferente nombre y en diferente directorio.

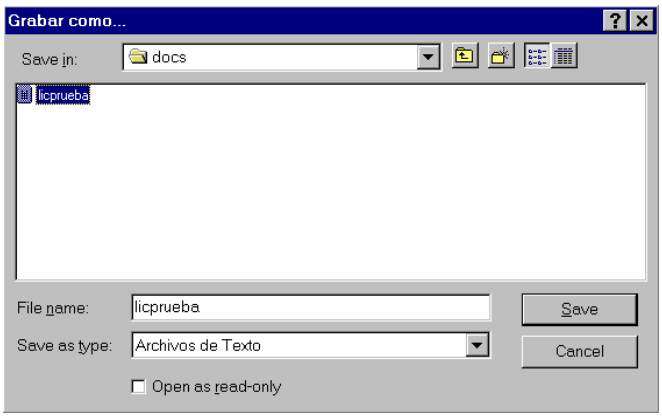


FIG. 3.3.22.1 Aplicación del componente SaveDialog.

3.3.23 USO DEL COMPONENTE FONTDIALOG.

Otro componente que vendrá a enriquecer nuestras aplicaciónes es FontDialog, este componente da al usuario la facilidad de escoger entre las fuentes (Tipos de Letras) disponibles en su particular instalación de Windows.
Use los siguientes pasos para ver el uso de este componente.

Abra una nueva aplicación.

1.- De la paleta de componentes Dialogs, seleccione un objeto FontDialog, y colóquelo en la forma, Cambie la propiedad ***Options*** haciendo doble clic en ella y cambiando a **true** las opciones ***fdApply*** y ***fdEffects***, el la hoja de eventos haga doble clic en el evento ***OnApply*** e introduzca el siguiente código:

```
Panel1->Font = FontDialog->Font;
```

2.- De la paleta de componentes Standard, seleccione un objeto Panel, y colóquelo en la forma, cambie la propiedad ***Caption*** por el texto “Prueba de cambio de letra”, modifique la propiedad ***Align*** por la opcion **alTop**.

3.- De la paleta de componentes Standard, seleccione un objeto Button, y colóquelo en la forma, Cambie la propiedad ***Caption*** por el texto “Tipo de letra” y el la hoja de eventos, seleccione haciendo doble clic el evento ***OnClick*** e introduzca el siguiente código:

```
if(FontDialog1->Execute())
Panel1->Font = FontDialog1->Font;
```

4.- Presione la tecla F9 para ejecutar su aplicación, que tendrá los siguientes efectos:

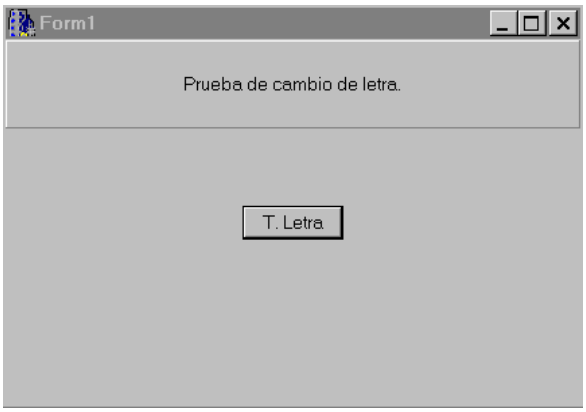


FIG. 3.3.23.1 Uso del componente *fontdialog*.

y al seleccionar el botón T. Aparecerá la siguiente caja de dialogo:

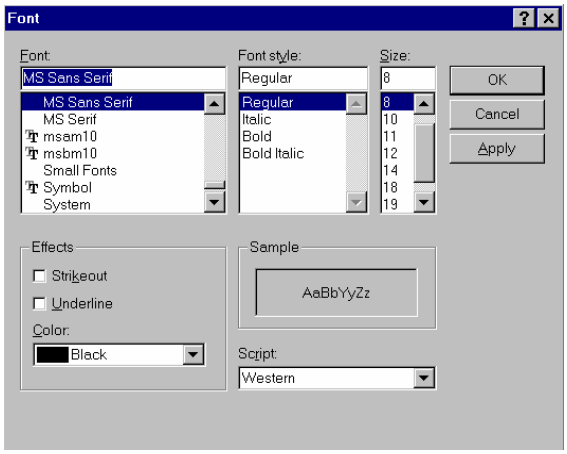


FIG. 3.3.23.2 Caja de dialogo para seleccionar tipos de letras.

y podrá seleccionar el tipo de letra que quiera.

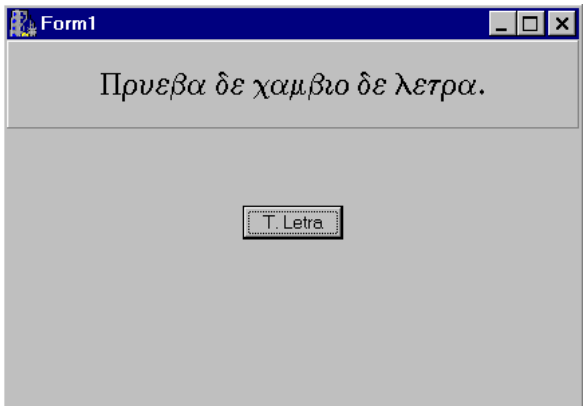


FIG. 3.3.23.3 Uso del componente *fontdialog*

3.3.24 USO DEL COMPONENTE COLORDIALOG.

El uso de este componente le da visibilidad, presentación y estilo a sus aplicaciones, ya que puede cambiar la característica del color mediante este componente.

El componente ColorDialog no tiene eventos, únicamente propiedades a modificar mediante el inspector de objetos.
La propiedad Color tiene los colores principales.
La propiedad Options tiene las siguientes subpropiedades.

Subpropiedad	Descripción
CdFullOpen	Da acceso a la paleta de colores extendida.
CdPreventFullOpen	Restringe el uso de colores a la paleta de colores actual.
CdShowHelp	Añade un botón de ayuda al componente ColorDialog.

Pero con los siguientes pasos podrá conocer su uso:

1.- Abra una nueva aplicación.

2.- De la paleta de componentes standard seleccione un objeto Panel, un objeto Memo y un objeto Button colocándolos en la forma.

3.- Seleccione el objeto Memo; con el inspector de objetos cambie la propiedad ***ScrollBars*** a la opción **ssVertical.**

4.- Seleccione de la paleta de componentes Dialogs un objeto ColorDialog  y colóquelo en la forma, con el inspector de objetos en la propiedad ***Options*** seleccione la surpopiedad **cdFullOpen** en true.

5.- Seleccione el objeto Panel y modifique la propiedad ***Align*** a la opción **alTop,** y borre el titulo en la propiedad ***Caption***.

6.- Seleccione el objeto Memo y modifique la propiedad ***Align*** a la opción **alTop**.

7.- Seleccione el objeto Button cambiando la propiedad ***Caption*** al titulo **Inicializar color**, en la hoja de eventos haga doble clic en el evento OnClick, en el editor de código introduzca el siguiente código:

```
if(ColorDialog1->Execute())
   {
      Panel1->Color=ColorDialog1->Color;
      Memo1->Lines->Assign(ColorDialog1->CustomColors);
   }
```

8.- Presione la tecla F9 para ejecutar su programa. Verá que al escoger el botón inicializar color, podrá seleccionar cualquier color y su codificación hexadecimal se editará en el objeto Memo.

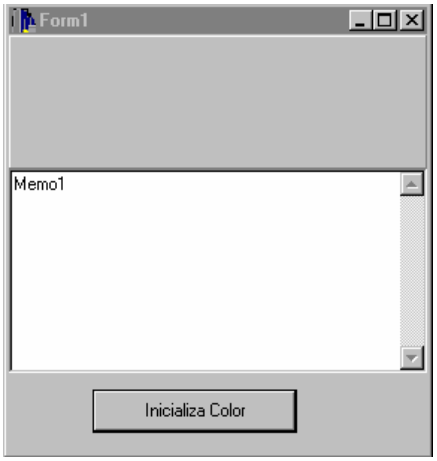


FIG. 3.3.24.1 Uso del componente colordialog.

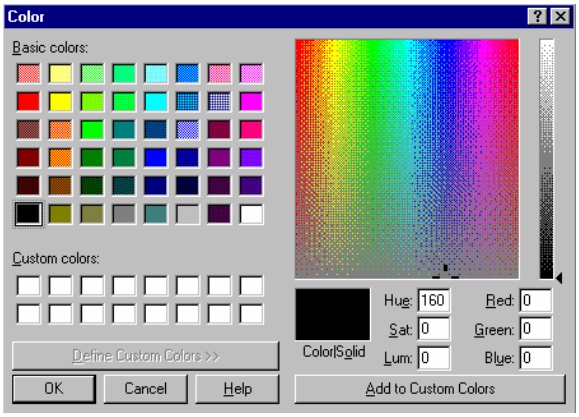


FIG. 3.3.24.2 Componente colordialog en colores verdaderos.

3.3.25 USO DEL COMPONENTE PRINTDIALOG.

El componente PrintDialog, complementa todos los componentes vistos anteriormente, ya que después de dejar la edición con las características deseadas, podemos presentarlos en Papel.

Veamos los siguientes pasos para conocer mejor su uso.

1.- Seleccione una nueva aplicación.

2.-De la paleta de componentes Standard seleccione un objeto Panel y colóquelo en la forma.

3.-De la paleta de componentes Additional seleccione un objeto SpeedButton y colóquelo en la forma dos veces.

4.- De la paleta de componentes Dialogs seleccione un objeto PrinterSetupDialog  y colóquelo en la forma.

5.- De la paleta de componentes Dialogs seleccione un objeto PrintDialog  y colóquelo en la forma.

6.- Seleccione un Botón acelerador haciendo doble clic e introduzca el siguiente código.

```
PrinterSetupDialog1->Execute();
```

7.- Seleccione el otro Botón acelerador haciendo doble clic e introduzca el siguiente código:

```
PrintDialog1->Execute();
```

Puede poner imagenes en los botones aceleradores usando la propiedad Glyph.

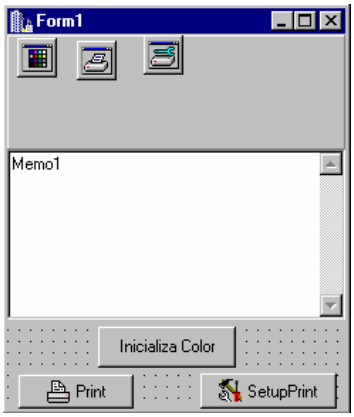


FIG. 3.3.25.1 Uso de Componentes PrinterSetupDialog y PrintDialog.

8.- Presione la tecla F9 para ejecutar su aplicación.

FIG. 3.3.25.2 Ejemplo de uso de Componentes PrinterSetupDialog y PrintDialog.

3.3.26 USANDO EL EXPLORADOR DE BASES DE DATOS

Para explorar de una forma sencilla cualquier base de datos, podemos usar el DataBase Explorer, complete los siguientes pasos:

1.- En la barra de menú principal en la sección DataBase elija la opción Explorer, obtendrá una ventana como la siguiente:

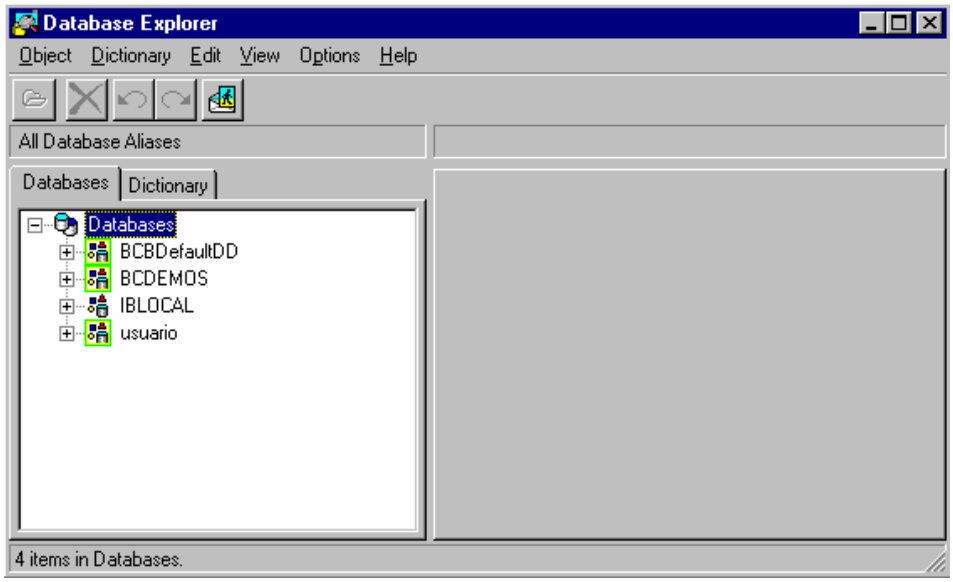


FIG. 3.3.26.1 Explorador de Bases de Datos.

2.- Seleccione las bases de datos de ejemplo (BCDEMOS) y seleccione la tabla BIOLIFE.DB en la parte derecha del explorador, podrá observar los detalles de dicha tabla.

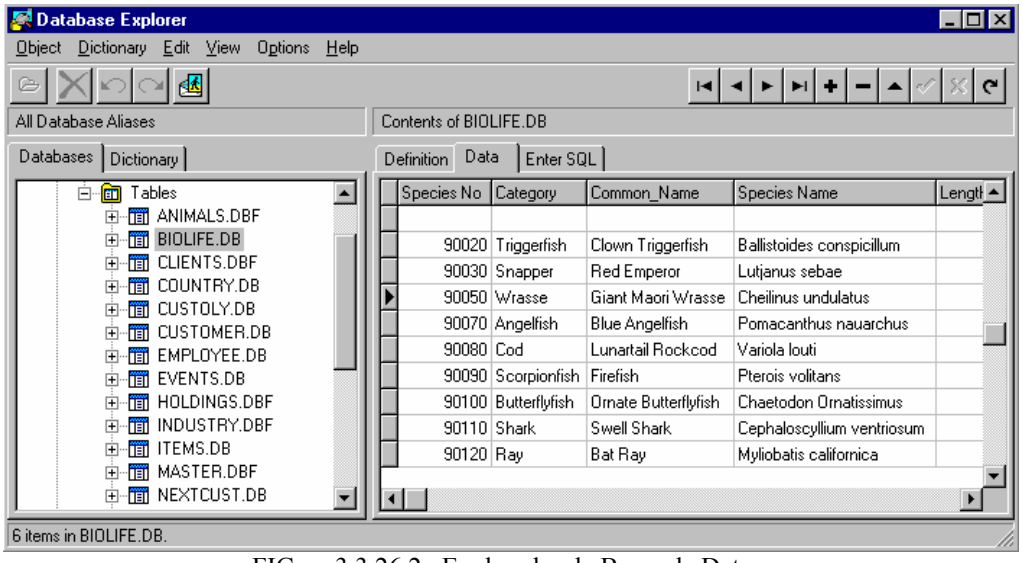


FIG. 3.3.26.2 Explorador de Bases de Datos.

3.- Si algún campo es Memo o Gráfico, se puede conocer su contenido haciendo doble clic en ese renglón.

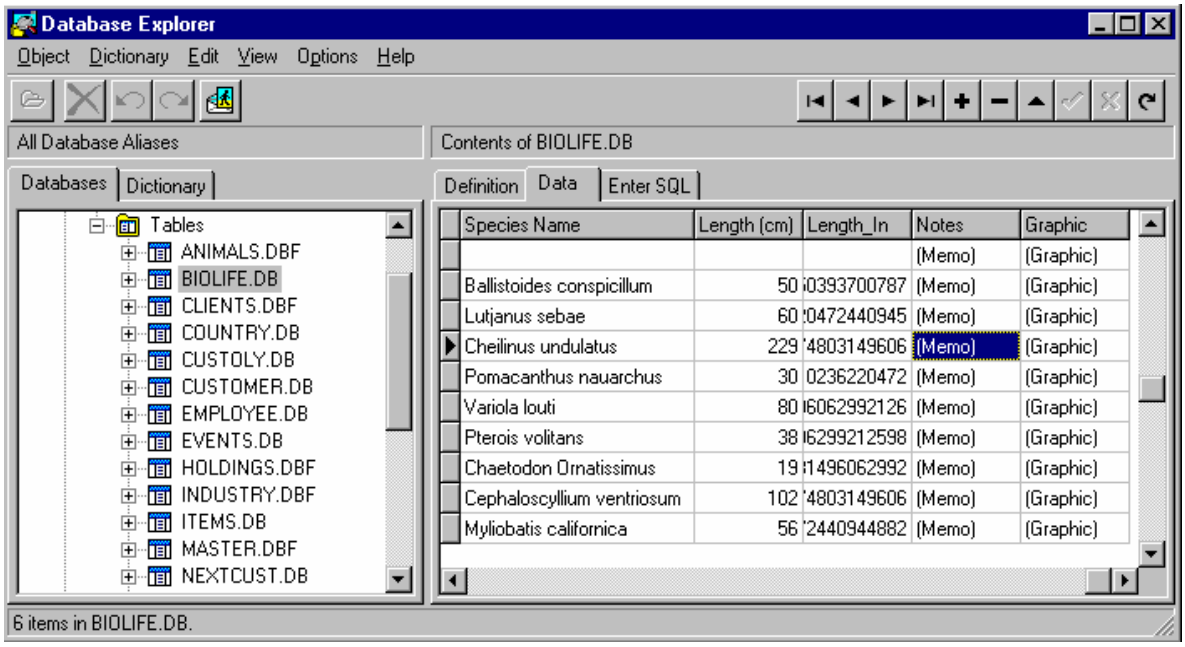


FIG. 3.3.26.3 Explorador de Bases de Datos.

4.- Al hacer doble clic aparecerá otra ventana con la información de ese campo.

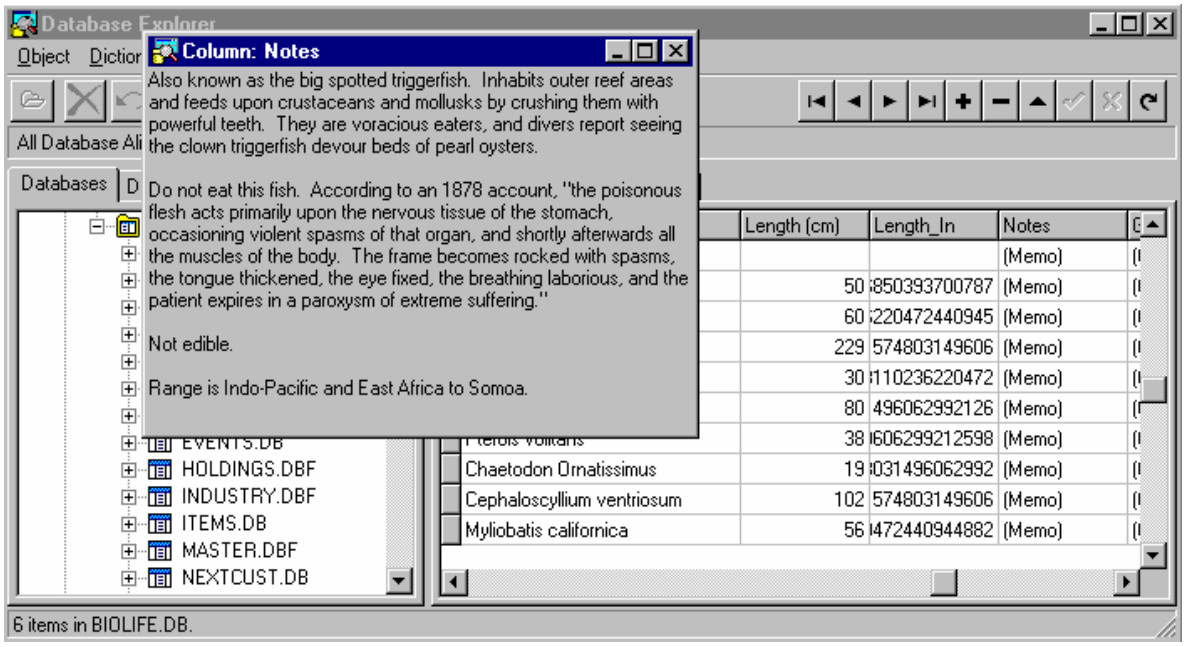


FIG. 3.3.26.4 Explorador de Bases de Datos.

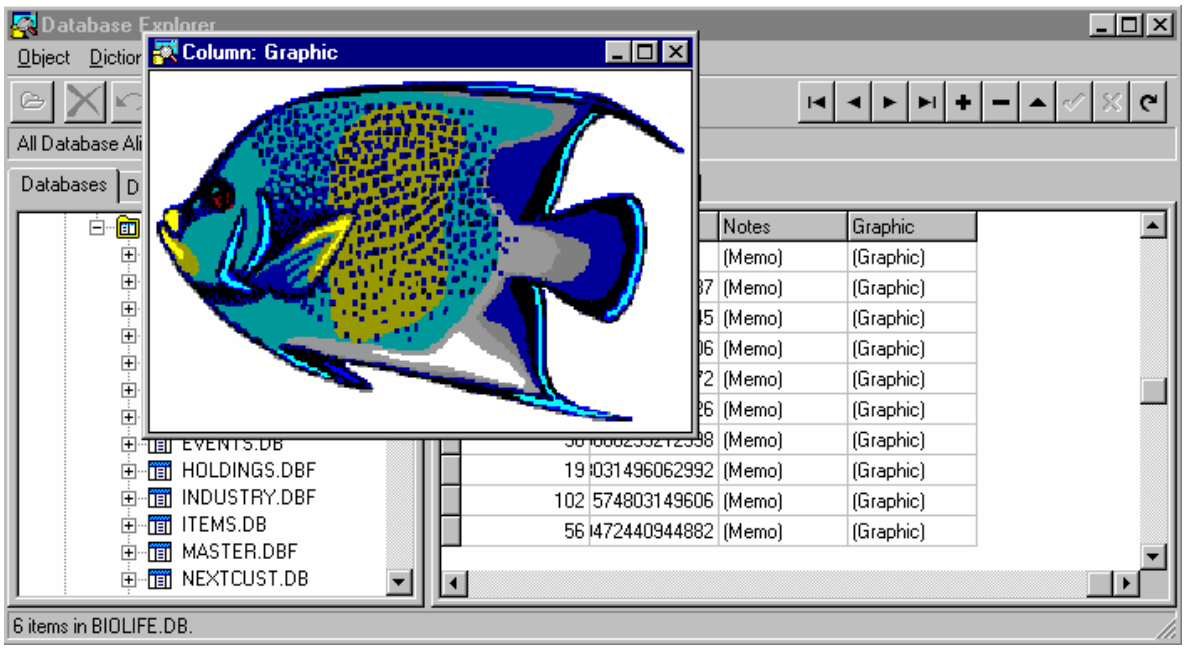


FIG. 3.3.26.5 Explorador de Bases de Datos.

3.3.27 USANDO EL DATABASE FORM WIZARD

C++ Builder viene equipado con una herramienta llamada Wizards, que facilita la creación de bases de datos, y nos ahorra en la edición de mascaras para la presentación de ellos.

Con los siguientes pasos podrá conocer un poco de su uso, ya que se combina con otros componentes para hacerlos más poderosos.

1.- Abra una nueva aplicación.

2.- Del menú principal elija DataBase $\Rightarrow$ Form Wizard y aparecerá una ventana como la siguiente.

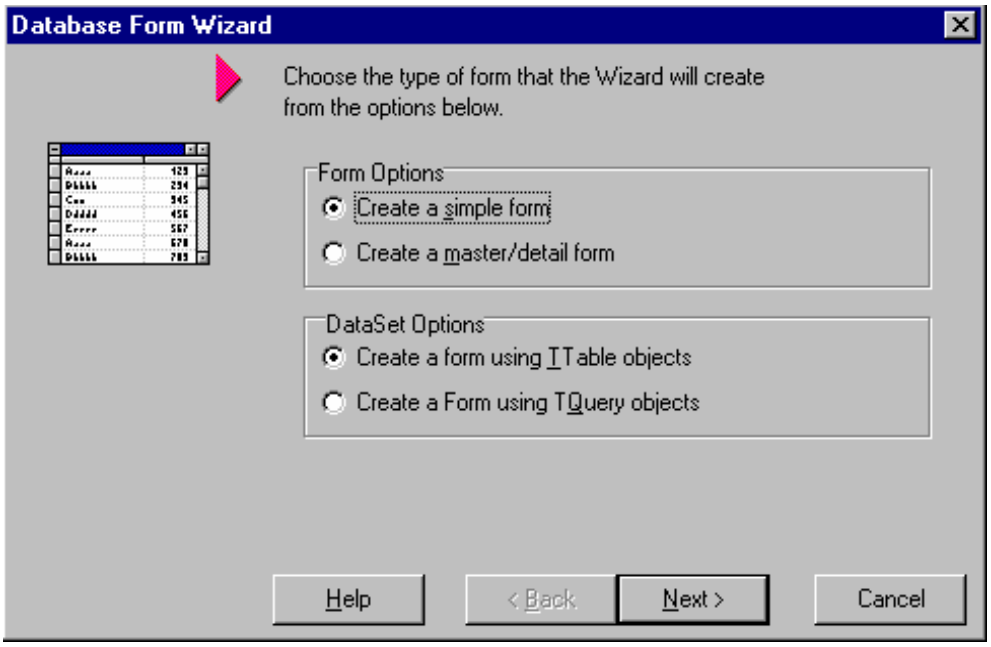


FIG. 3.3.27.1 Usando el DataBase Form Wizard.

3.- Seleccione el botón Next para continuar.

4.- Seleccione en el subdirectorio de Ejemplos ⇒ Data, seleccione la tabla de datos llamada BIOLIFE.DB como lo muestra la figura.

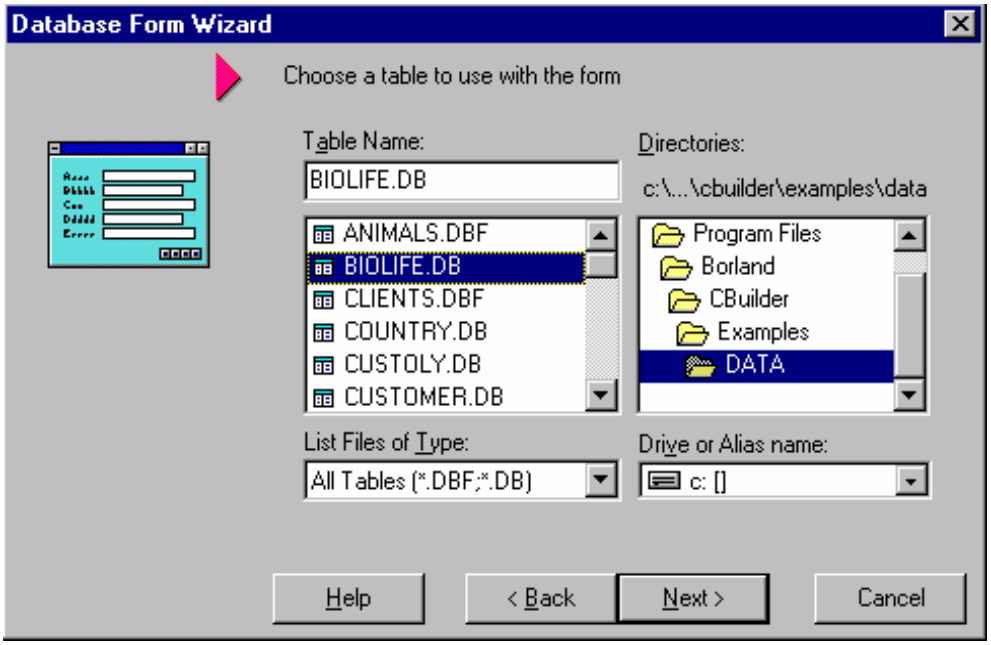


FIG. 3.3.27.2 Cargando una tabla de datos.

5.- Oprima el botón Next.

6.- Seleccione el botón >> para añadir todas los campos de la tabla a visualización.

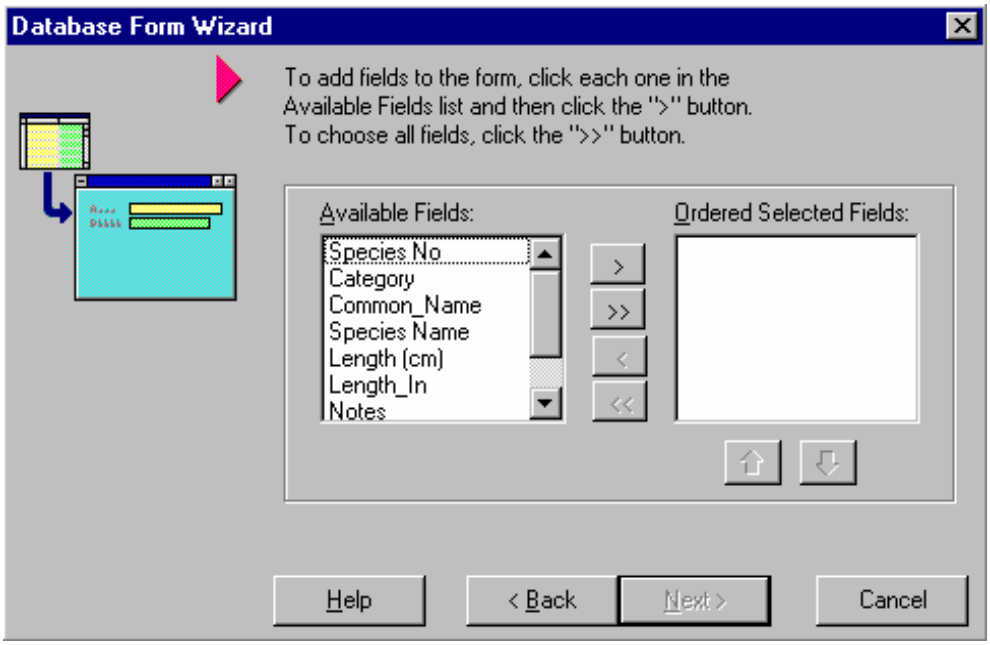


FIG. 3.3.27.3 Anexando campos para visualización.

7.- Seleccione el botón Next, y luego otra vez Next.

8.- Seleccione el botón Finish.

9.- Del menú principal elija Project Remove From Project y seleccione Form1 y haga clic en el botón Ok.

10.- Presione F9 para ejecutar la aplicación.

Se ha creado la primera aplicación de Bases de Datos.

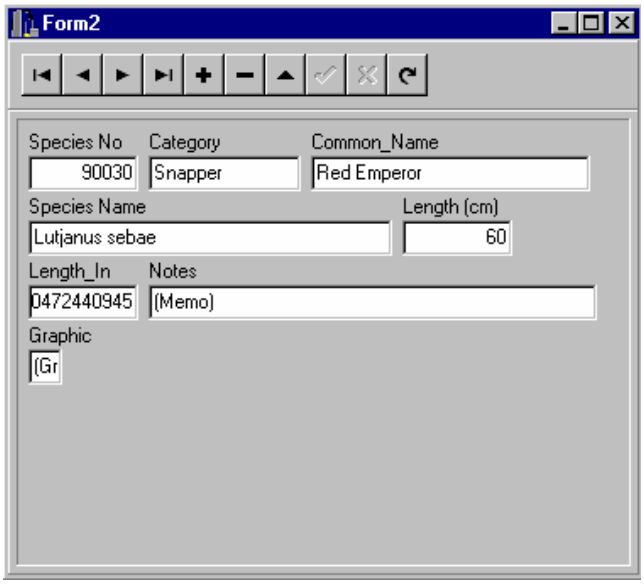


FIG. 3.3.27.4 Presentación en pantalla de los datos seleccionados.

3.3.28 USANDO EL COMPONENTE TABLE

Este componente se encuentra en la hoja Data Access de la paleta de componentes.
El componente Tabla representa una simple tabla de datos con la que podrá visualizar los campos de la base de datos seleccionada.

Con los siguientes pasos puede ver la forma en que se maneja el componente tabla.

1.- Abra una nueva aplicación.

2.- De la hoja de componentes Data Access seleccione el componente Table  y colóquelo en la forma, con el inspector de objetos modifique la propiedad ***DatabaseName*** y coloque en ella el valor BCDEMOS, la propiedad ***TableName*** modifiquela por el valor CUSTOMER.DB y la propiedad ***Active*** modifiquela a true, para activar la tabla seleccionada, mientras este valor se encuentre en fase, la tabla estará desactivada.

3.- De la hoja de componentes Standard seleccione el componente Memo y colóquelo en la forma.

4.- De la hoja de componentes Standard seleccione el componente Button y colóquelo en la forma con tres copias mas, alineandolos en la parte superior de su forma, modifique con el ispector de objetos la propiedad Caption de la siguiente manera.

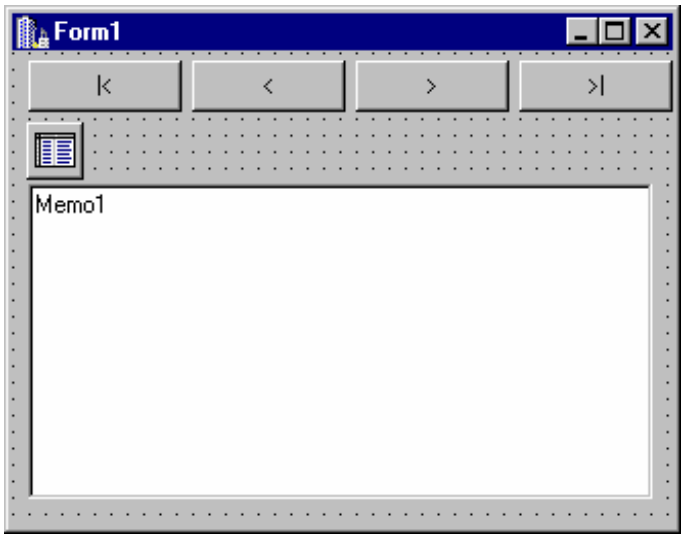


FIG. 3.3.28.1 Acomodo de componentes.

5.- Haga doble clic en el botón |< e introduzca el siguiente código:

```
Table1->First();
LoadMemo();
```

6.- Haga doble clic en el botón < e introduzca el siguiente código:

```
Table1->Prior();
if(Table1->Bof)
  ShowMessage("Es el inicio de la tabla ");
  LoadMemo();
```

7.- Haga doble clic en el botón > e introduzca el siguiente código:

```
Table1->Next();
if(Table1->Eof)
 ShowMessage("Final de la tabla ");
 LoadMemo();
```

8.- Haga doble clic en el botón > e introduzca el siguiente código:

```
Table1->Last();
LoadMemo();
```

9.- Al final en el editor de código anexe la función siguiete:

```
void TForm1::LoadMemo()
{
 Memo1->Clear();
 for(int i=0;i<Table1->FieldCount-1;i++)
   Memo1->Lines->Add(Table1->Fields[i]->FieldName + " : " + Table1->Fields[i]-
>Text);
}
```

10.- Con el botón derecho del Mouse haga clic en el editor de código y seleccione la opción Open Source/Header File y anexe la siguiente línea en posición.

```
private:        // User declarations
void LoadMemo();
public:         // User declarations
```

11.- Presione F9 para correr su programa y obtendrá el siguiente resultado:

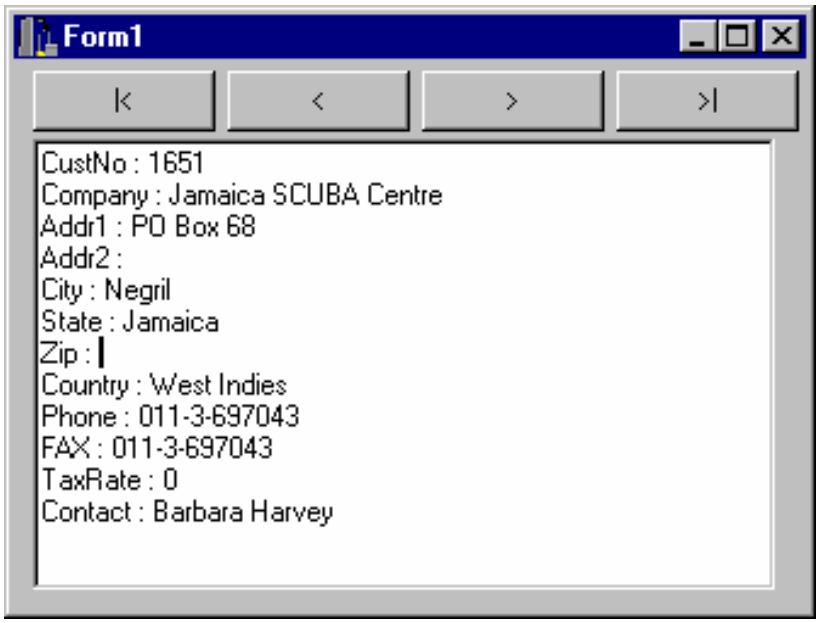


FIG. 3.3.28.2 Uso del componente Table.

3.3.29 USANDO EL COMPONENTE DBGRIDY DATA SOURCE

El componente DataSource es un componente no visual, la función que realiza es enlazar la base de datos a los objetos visuales.
El componente DBGrid nos proporciona de una poderosa herramienta para capturar y observar los datos de una tabla, veamos como funciona.

1.- Cree una nueva aplicación.

2.- De la hoja de componentes Data Access seleccione el componente Table  y colóquelo en la forma, con el inspector de objetos modifique la propiedad ***DatabaseName*** y coloque en ella el valor BCDEMOS, la propiedad ***TableName*** modifíquela por el valor CUSTOMER.DB y la propiedad ***Active*** modifíquela a true, para activar la tabla seleccionada, mientras este valor se encuentre en fase, la tabla estará desactivada.

3.- De la hoja de componentes Data Access seleccione el componente DataSource  y colóquelo en la forma, con el inspector de objetos modifique la propiedad ***DataSet*** colocando en ella **Tabla1**.

4.- De la hoja de componentes Data Controls seleccione el componente DBGrid  y colóquelo en la forma, modifique la propiedad DataSource a la opción DataSource1, si la propiedad ***Auto Edit*** esta en true podrá editar los datos de la tabla.

5.- De la hoja de componentes Data Access seleccione el componente DataSource  y colóquelo en la forma, con el inspector de objetos modifique la propiedad ***DataSet*** colocando en ella **Tabla1**.

6.- Presione F9 para ejecutar su programa.

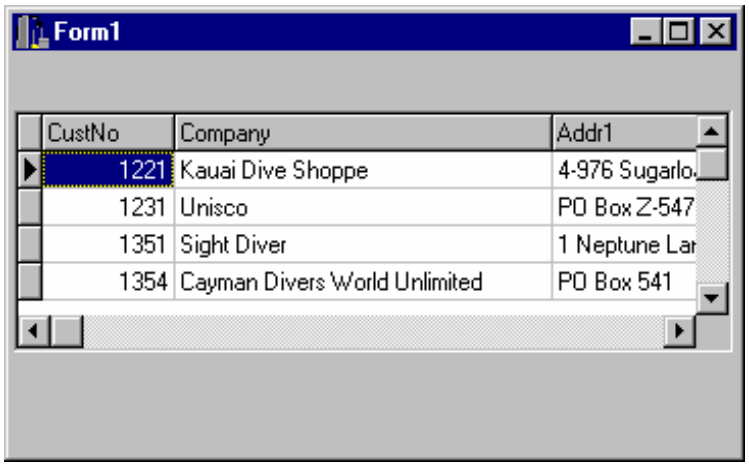


FIG. 3.3.28.1 Usando el componente DBGrid.

Si usted quisiera solamente visualizar algunas columnas de datos y ocultar las demás,

Puede utilizar los siguientes pasos:

1.- Seleccione el componente DBGrid que ya tiene en su forma y en la propiedad Columns haga doble clic y aparecerá una caja de dialogo como la siguiente:

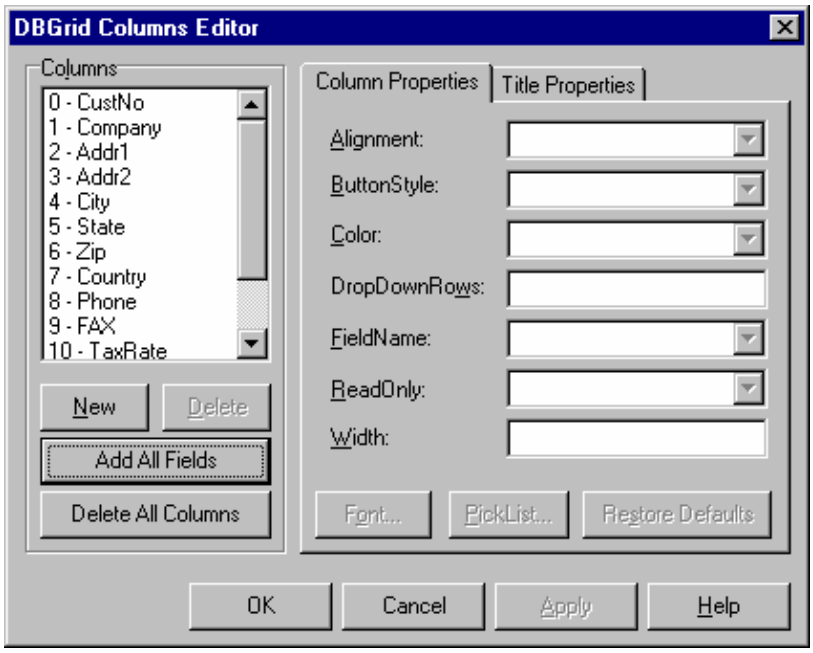


FIG. 3.3.28.2 Editor de Columnas DBGrid.

2.- Seleccione el botón Add All Fields para tener todos los campos de la tabla seleccionada.

3.- Con los campos en la ventana de edición puede borrar los campos que no desea que sean mostrados, y al finalizar de borrarlos de la edición; seleccione el botón Ok.

4.- Presione F9 y vea sus datos seleccionados.

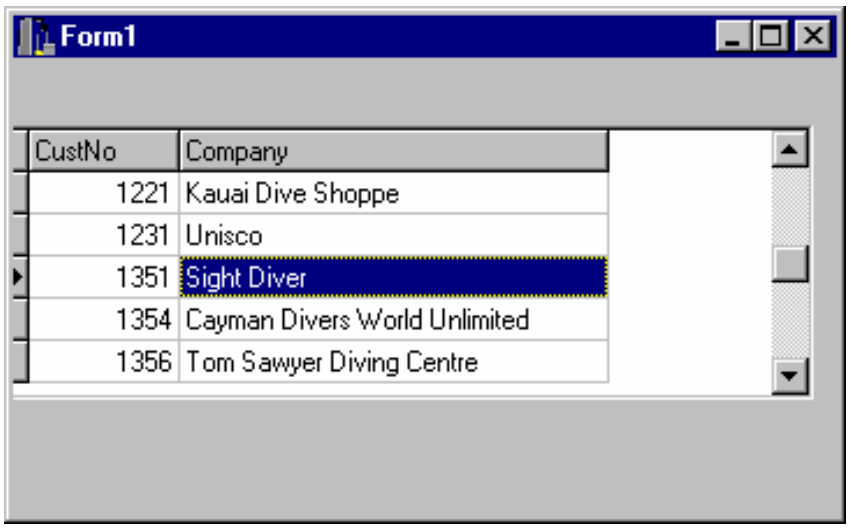


FIG. 3.3.28.3 Columnas seleccionadas para visualización.

3.3.30 USANDO EL COMPONENTE DBTEXT

El componente DBText es utilizado como un campo variante, que puede tomar los valores de un campo de la tabla que se active, para conectar este objeto a la tabla deseada solo se tiene que poner en la propiedad ***DataSource*** cual es la tabla que se le asociará.

1.- Cree una nueva aplicación.

2.- De la hoja de componentes Data Access seleccione el componente Table  y colóquelo en la forma, con el inspector de objetos modifique la propiedad ***DatabaseName*** y coloque en ella el valor BCDEMOS, la propiedad ***TableName*** modifíquela por el valor CUSTOMER.DB y la propiedad ***Active*** modifíquela a true, para activar la tabla seleccionada, mientras este valor se encuentre en fase, la tabla estará desactivada.

3.- De la hoja de componentes Data Access seleccione el componente DataSource  y colóquelo en la forma, con el inspector de objetos modifique la propiedad ***DataSet*** colocando en ella **Tabla1**.

4.- De la hoja de componentes Data Controls seleccione el componente DBGrid  y colóquelo en la forma, modifique la propiedad DataSource a la opción DataSource1, si la propiedad ***Auto Edit*** esta en true podrá editar los datos de la tabla.

5.- De la hoja de componentes Data Controls seleccione el componente DBText  y colóquelo en la forma, y modifique la propiedad ***DataSource*** a la opción **DataSource1**, y seleccione cual es el nombre del campo que quiere que se muestre en el componente DBText (Como por ejemplo Addr1).

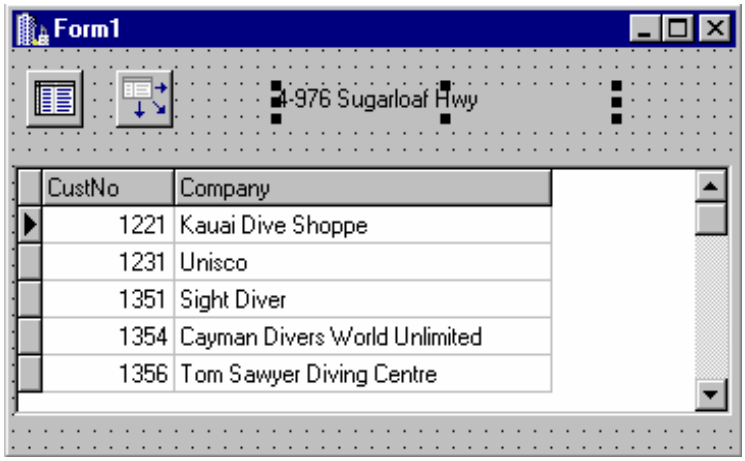


FIG. 3.3.29.1 Usando el componente DBText.

6.- Presione F9 para ejecutar su programa y cada vez que recorra un campo de la tabla aparecerá la dirección del campo activo.

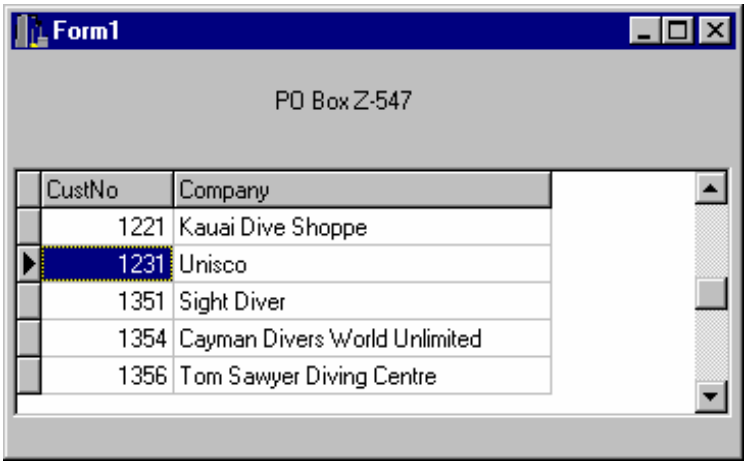


FIG. 3.3.29.2 Usando el componente DBText en al aplicación.

3.3.31 USANDO EL COMPONENTE DBEDIT

El uso de este componente es igual que el componente Edit de la paleta de componentes Standard, pero con la ventaja de poderla asociar a una tabla de base de datos.

Al ejercicio anterior anéxele los siguientes pasos:

1.- De la hoja de componentes Data Controls seleccione un objeto DBEdit  y colóquelo en la forma, con el inspector de objetos modifique la propiedad ***DataSource*** al valor **DataSource1**, y en la propiedad ***DataField*** seleccione otro campo que quiera ser visualizado al igual que en DBText.

2.- Presione F9 para ver el resultado de la aplicación.

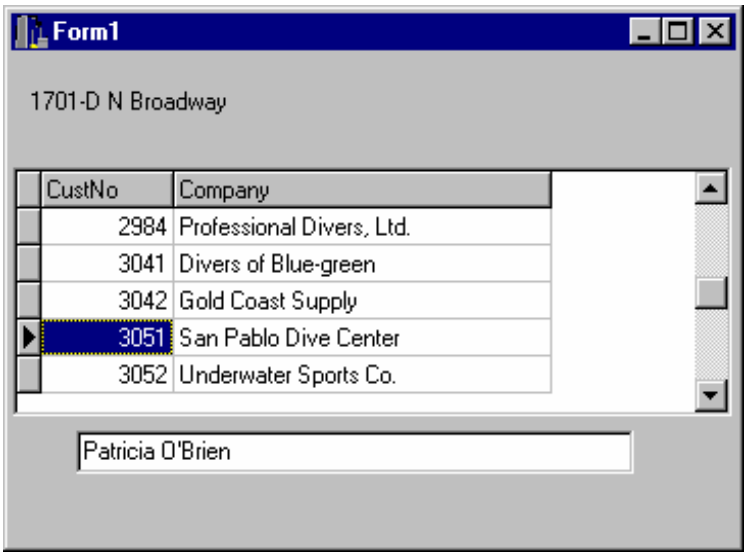


FIG. 3.3.30.1 Usando el componente DBEdit

3.3.32 USANDO EL COMPONENTE DBMEMO Y DBIMAGE

Como el componente DBEdit no puede presentar mas de una línea a la vez, se provee de este campo DBMemo, para presentar campos que contengan más de una línea.
El campo DBImage presenta la imagen almacenada en un campo de la base de datos.

Para ver el uso de este componente siga las instrucciones:

1.- Seleccione Nueva Aplicación.

2.- De la hoja de componentes Data Access seleccione el componente Table  y colóquelo en la forma, con el inspector de objetos modifique la propiedad ***DatabaseName*** y coloque en ella el valor BCDEMOS, la propiedad ***TableName*** modifíquela por el valor **BIOLIFE.DB** y la propiedad ***Active*** modifíquela a **true**, para activar la tabla seleccionada, mientras este valor se encuentre en fase, la tabla estará desactivada.

3.- De la hoja de componentes Data Access seleccione el componente DataSource  y colóquelo en la forma, con el inspector de objetos modifique la propiedad ***DataSet*** colocando en ella **Tabla1**.

4.- De la hoja de componentes Data Controls seleccione el componente DBGrid  y colóquelo en la forma, modifique la propiedad DataSource a la opción DataSource1, si la propiedad ***Auto Edit*** esta en **true** podrá editar los datos de la tabla.

5.- De la hoja de componentes Data Controls seleccione el componente DBMemo  y colóquelo en la forma, modifique la propiedad ***DataSource*** a la opción **DataSource1**, y la propiedad ***DataField*** en la opción **Notes**.

6.- De la hoja de componentes Data Controls seleccione el componente DBImage  y colóquelo en la forma, modifique la propiedad ***DataSource*** a la opción **DataSource1**, y la propiedad ***DataField*** en la opción **Image**.

7.- Presione la tecla F9 para observar los resultados.

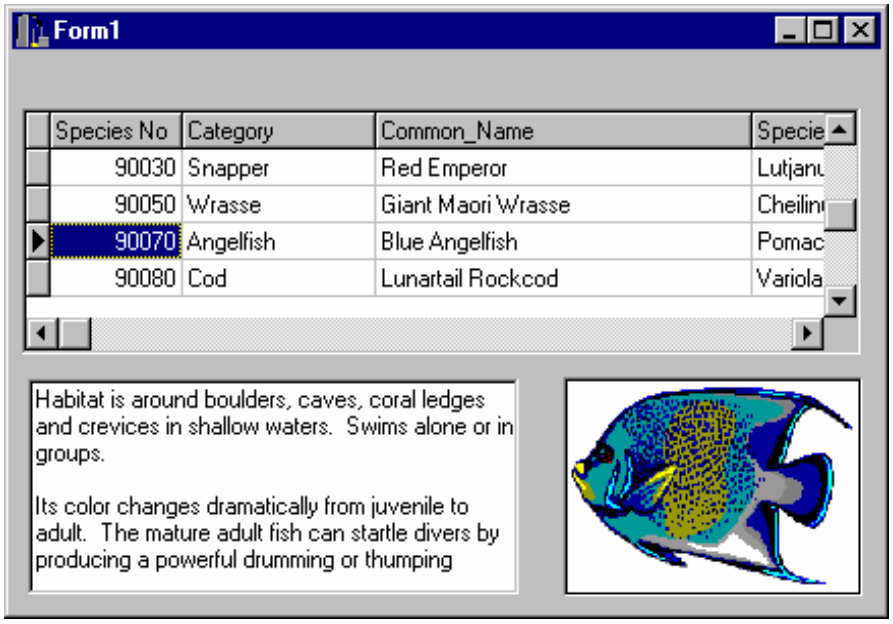


FIG. 3.3.31.1 Usando el componente DBMemo y DBImage

3.3.33 USANDO EL COMPONENTE DBNAVIGATOR

se usan tablas para navegar a través de los registros de la base de datos, el componente DBNavigator le proporciona una forma de controlar y editar los campos por medio de botones, vea el siguiente ejemplo para conocer su funcionamiento.

Utilice el programa anterior y anexe los siguientes pasos:

1.- De la hoja de Componentes Data Controls seleccione un objeto DBNavigetor y colóquelo en la forma, en la propiedad ***DataSource*** seleccione la opción **DataSource1**, y la propiedad ***ShowHint*** a **true**.

Y obtendrá el siguiente resultado.

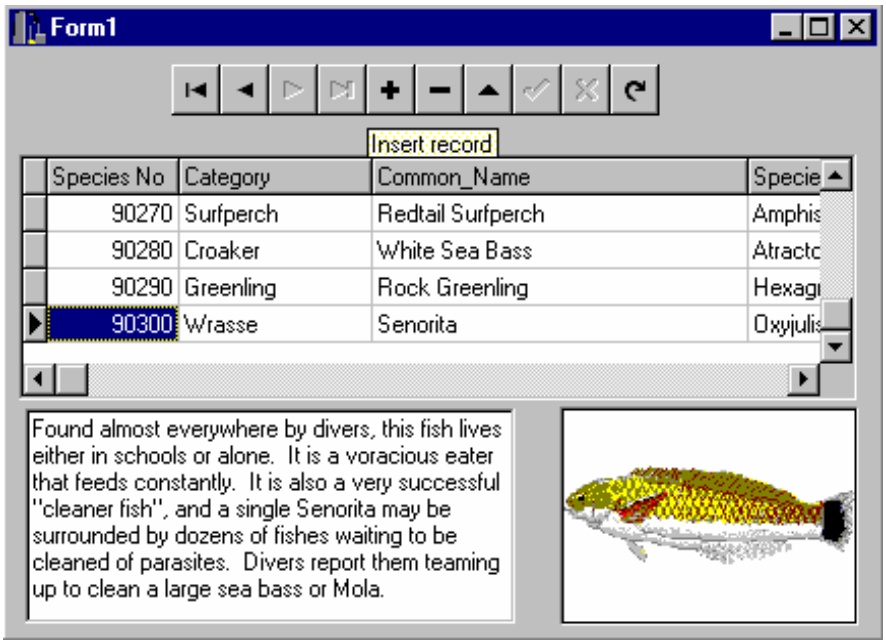


FIG. 3.3.32.1 Usando el componente DBNavigator

3.3.34 USANDO EL COMPONENTE QUERY

El componente Query, sirve para filtrar datos según el valor que se desee de algún campo. Siga el ejemplo para ver su uso.

1.- Seleccione nuevo proyecto.

2.- En la paleta de componentes Data Access seleccione el objeto Query 

y colóquelo en la forma, en la propiedad ***DataBaseName*** seleccione **BCDEMOS**, haga doble clic en la porpiedad SQL para que introduzca el siguiente código en editor de líneas:

```
SELECT  * FROM EVENTS WHERE VENUENO = 5;
```

Y haga clic en el botón Ok. Modifique la propiedad ***Active*** a **true**.

3.- En la paleta de componentes Data Access seleccione el objeto DataSource  y colóquelo en la forma, en la propiedad ***DataSet*** elija la opción **Query1**.

4.- En la paleta de componentes Data Controls seleccione el objeto DBGrid  y colóquelo en la forma, en la propiedad ***DataSource*** elija la opción **DataSource1** y la propiedad ***Align*** en **alBottom**.

5.- Presione F9 para observar el resultado.

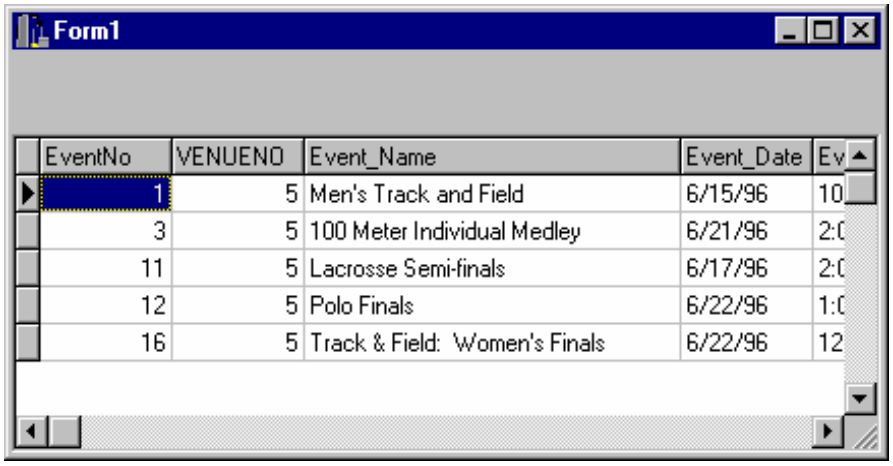


FIG. 3.3.33.1 Usando el componente *query*

La idea para la utilización del resto de los componentes es la misma, se han presentado solo algunos de los más comunes en la programación.

Capítulo IV

Desarrollo de un Laboratorio Virtual

Al comenzar a trabajar con el DR. José Luis Marroquín Zaleta comentamos sobre la posibilidad de realizar una aplicación matemática enfocada a la enseñanza de un tema particular de la geometría analítica a nivel bachillerato que es: la recta. Se penso que fuera de una manera sencilla pero que cubriera la mayoría de los puntos vistos en la preparatoria sobre el tema y que integrara un módulo de ayuda sensible al contexto (Hipertexto).

Después de varias recomendaciones, el proyecto se diseñó de la siguiente manera:

Comenzará con una pantalla de selección en la que se tendrán los temas por ver; en esa pantalla estarán los logotipos de las instituciones participantes en el proyecto:

El Centro de Investigación en Matemáticas (CIMAT)
representado por el DR. José Luis Marroquín Zaleta, como asesor del proyecto.
El Instituto Tecnológico de Morelia (ITM)
representado por el MC. Cristobal Villegas Santoyo, como asesor del proyecto y Juan Carlos Cruz Vega como realizador del proyecto.

Esta pantalla estará controlada por un menú de opciones, en este menú de opciones se encontrará el módulo de consulta de Hipertexto.

Al seleccionar cualquiera de las opciones que presenta la pantalla inicial se llamará a otro módulo que contiene la aplicación, cada una de ellas tiene su funcionalidad particular y también tienen su propio módulo de consulta de Hipertexto todo esto controlado también por su menú particular, la funcionalidad y manejo de cada módulo se describe a continuación:

4.1 PRIMER MODULO INTERACTIVO
(Programa Punto.Cpp)

Objetivo: Presentación gráfica de coordenadas en el sistema coordenado cartesiano, habilidad para reconocer posiciones en un sistema coordenado.

Estrategia: Elaboración de un módulo en el cual el usuario mediante el mouse coloque puntos aleatorios en el sistema coordenado, y el programa muestre según el punto marcado la coordenada de ese punto.

Presentación del texto relacionado al tema mediante un Hipertexto asociado a la aplicación final.

Para la elaboración del primer módulo se diseña una máscara que presenta:

1.- Una ventana de diálogo que tenga las instrucciones de manejo del programa.
2.- Una ventana de diálogo que contenga información del módulo.
3.- Un menú de opciones que contenga:

a) Un Campo Opciones.
   - &Salir ⇒ Para finalizar el programa.
   - &Limpiar ⇒ Para limpiar la ventana de trabajo.

b) Un Campo Ayuda
   - &Uso ⇒ Información del manejo del módulo.
   - &Acerca de… ⇒ Información de la elaboración del módulo.

c) Un Campo Base Teórica. Hipertexto de lo esencial del tema.

Diseño:

La construcción del módulo se realizó mediante los siguientes pasos:

1.- Seleccionamos del menú principal File ⇒ New Application, para tener una forma sin inicializar.

2.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	" Sistema Coordenado ".
b) La propiedad ClientHeight	⇒	721
c) La propiedad ClientWidth	⇒	1024
d) La propiedad Color	⇒	" clBack "
e) La propiedad Icon	⇒	BildCim.ico
f) La propiedad Left	⇒	0

g) La propiedad Top ⇒ 0

Y lo que veremos será la ventana en la cual tendremos el control del programa:

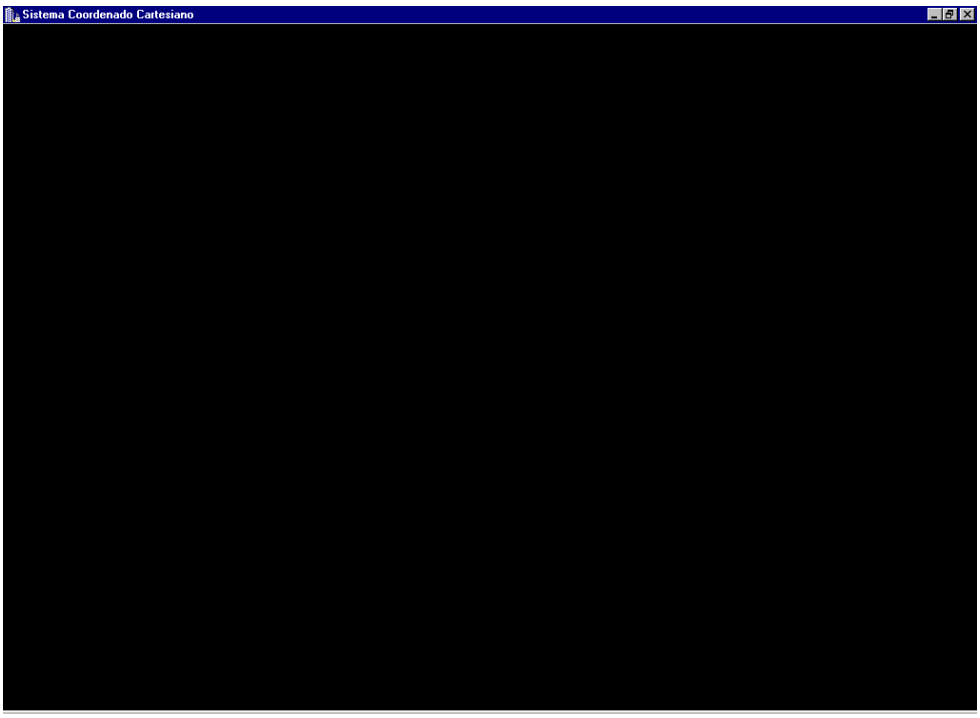


FIG. 4.1.1 Ventana de trabajo módulo No. 1.

3.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

4.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	265
c) La propiedad Width	⇒	553
d) La propiedad Color	⇒	“ clBtnFace “
e) La propiedad Left	⇒	208
f) La propiedad Top	⇒	144

5.- De la paleta de componentes Standard seleccione un objeto Button  y colóquelo en el Panel.

6.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Continuar.
b) La propiedad Height	⇒	89
c) La propiedad Width	⇒	129
d) La propiedad Left	⇒	392
e) La propiedad Top	⇒	96

Haga doble clic en el botón Continuar e introduzca el siguiente código.

```
Panel1->Visible=false;
```

7.- De la paleta de componentes Standard seleccione un objeto Memo  y colóquelo en el Panel.

8.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	241
b) La propiedad Width	⇒	377
c) La propiedad Left	⇒	8
d) La propiedad Top	⇒	16
e) La propiedad Lines	⇒	Doble clic

Aparecerá el editor de Líneas, en el se tecleará el siguiente Texto.

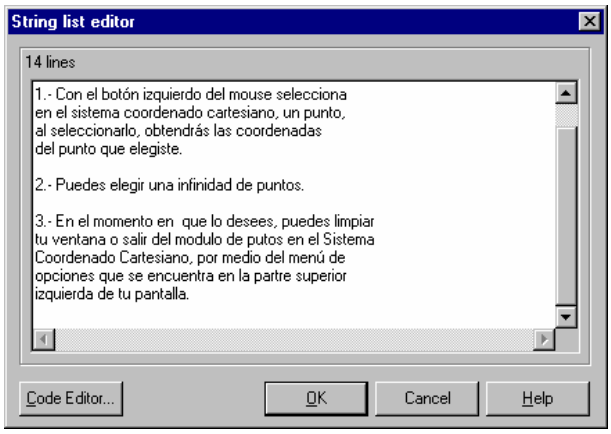


FIG. 4.1.2 Editor de Líneas

En el se tecleará el siguiente Texto.

INDICACIONES
a).- Con el botón izquierdo del mouse selecciona en el sistema coordenado cartesiano, un punto, al seleccionarlo, obtendrás las coordenadas del punto que elegiste.

b).- Puedes elegir una infinidad de puntos.

c).- En el momento en que lo desees, puedes limpiar tu ventana o salir del modulo de puntos en el Sistema Coordenado Cartesiano, por medio del menú de opciones que se encuentra en la parte superior izquierda de tu pantalla.

9.- Selecciona el botón Ok para cerrar la ventana de dialogo.

10.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

11.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	169
c) La propiedad Width	⇒	153
d) La propiedad Color	⇒	“ clTeal “
e) La propiedad Left	⇒	840
f) La propiedad Top	⇒	16

12.- De la paleta de componentes Additional seleccione un objeto Image  y colóquelo en el Panel2.

13.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	73
b) La propiedad Width	⇒	57
c) La propiedad Left	⇒	48
f) La propiedad Top	⇒	8
e) La propiedad Stretch	⇒	True
f) La propiedad Picture	⇒	Doble clic

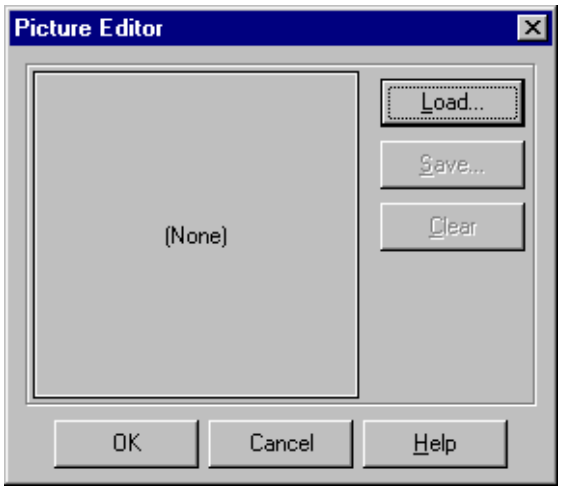


FIG. 4.1.3 Presentador de Imágenes.

Presione el botón Load y obtendrá una caja de dialogo para seleccionar el icono deseado o podrá buscarlo en caso de no encontrarse en el directorio mostrado.

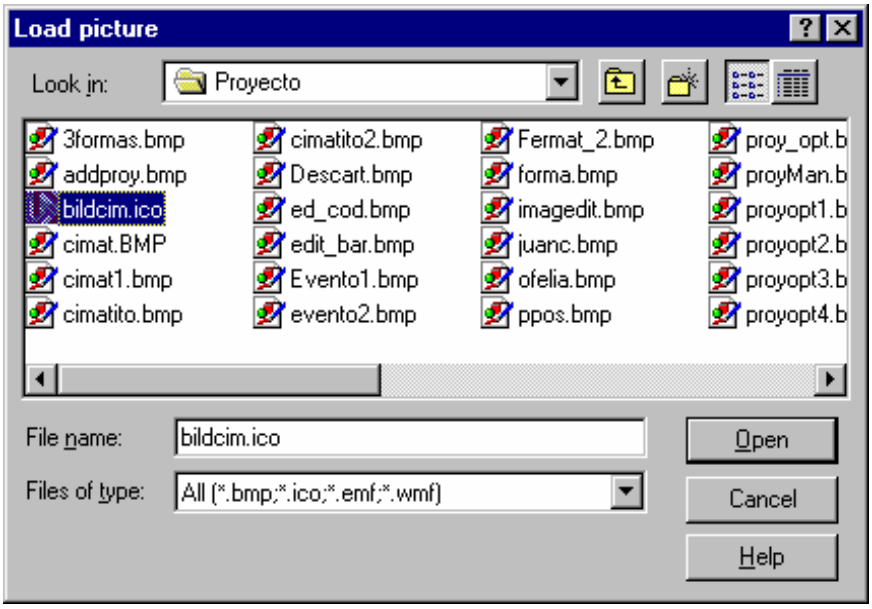


FIG. 4.1.4 Selección de Imágenes.

Seleccione la imagen deseada y oprima el botón Open.

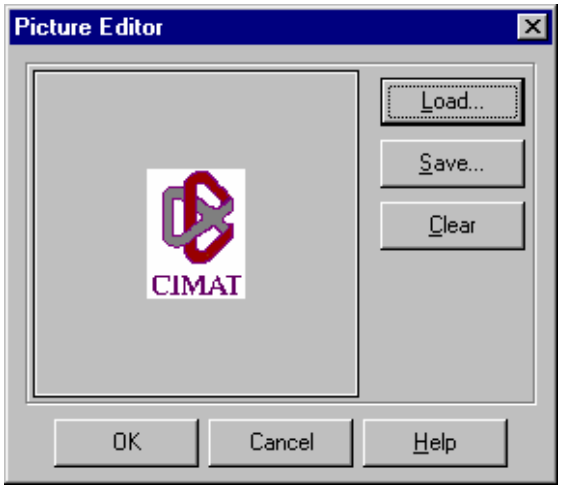


FIG. 4.1.5 Selección de Imágenes.

Oprima el botón Ok.

14.- De la paleta de componentes Standard seleccione un objeto Button 

y colóquelo en el Panel2.

15.- Modifiquemos los siguientes atributos:

Propiedad	Valor o Texto

a) La propiedad Caption ⇒ &Continuar
b) El evento OnClick ⇒ Panel2->Visible=false;

16.- De la paleta de componentes Standard seleccione un Label  y colóquelo en el Panel2 con otras dos copias.

17.- Modifiquemos los siguientes atributos:

Label1.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Geometría Parte 1.
b) La propiedad Height	⇒	24
c) La propiedad Width	⇒	179
d) La propiedad Color	⇒	" clGray "
e) La propiedad Left	⇒	16
f) La propiedad Top	⇒	120
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	16

Label2.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	JUAN CARLOS CRUZ VEGA
b) La propiedad Height	⇒	13
c) La propiedad Width	⇒	167
d) La propiedad Color	⇒	" clGray "
e) La propiedad Left	⇒	16
f) La propiedad Top	⇒	144
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	9

Label3.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	FEBRERO DE 1998
b) La propiedad Height	⇒	13
c) La propiedad Width	⇒	118
d) La propiedad Color	⇒	" clGray "
e) La propiedad Left	⇒	32

f) La propiedad Top ⇒ 160
g) La propiedad Font Style ⇒ Bold Italic
h) La propiedad Font Size ⇒ 9

18.- De la paleta de componentes Standard seleccione un objeto Main Menu  y colóquelo en la forma.

19.- haga doble clic en este componente y obtendrá el diseñador de menus:

20.- Con el inspector de objetos cambie las propiedades:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Opciones + Enter
b) La propiedad Caption	⇒	&Ayuda + Enter
c) La propiedad Caption	⇒	&Base teórica + Enter

21.- Seleccione la hoja de eventos y haga doble clic en el evento OnClick e introduzca el siguiente código.

Componente	**Código**
Opción **&Base Teórica**;	Application->HelpCommand(HELP_CONTENTS,0);
Evento ***OnClick***	Invalidate();

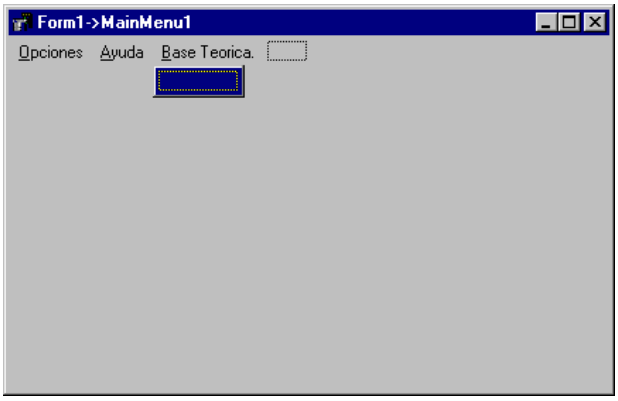


FIG. 4.1.6 Diseñador de menús.

22.- En el sub menu Opciones anexar 2 campos mas.

Propiedad	Valor o Texto

a) La propiedad Caption	⇒	&Limpiar
a) La propiedad Caption	⇒	&Salir

De manera que tenga el siguiente resultado:

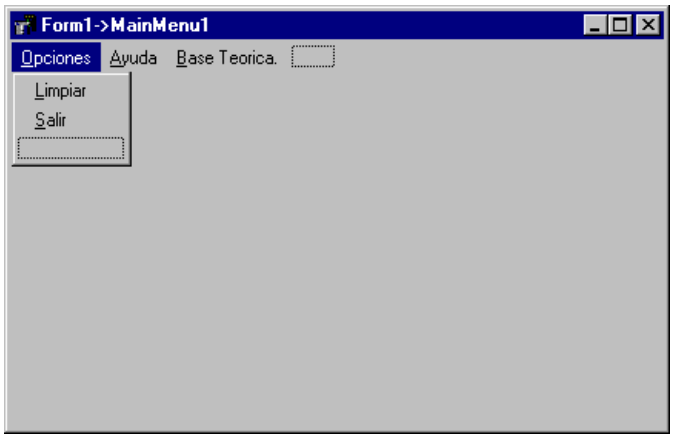


FIG. 4.1.7 Opciones de Sub menú.

23.- Seleccione la hoja de eventos y haga doble clic en el evento OnClick e introduzca el siguiente código.

	Componente	**Código**
Opción	**&Limpiar**;	Invalidate();
Evento	***OnClick***	
Opción	**&Salir**;	Close();
Evento	***OnClick***	Exit(1);

24.- En el sub menú Ayuda anexar 2 campos mas.

	Propiedad		Valor o Texto
a) La propiedad Caption		⇒	&Uso
a) La propiedad Caption		⇒	a&Cerca de…

De manera que se vea de la siguiente forma:

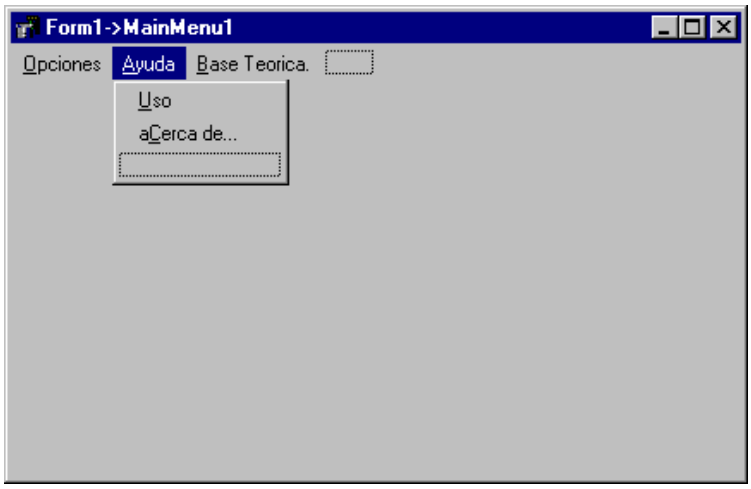


FIG. 4.1.8 Opciones de Sub menú.

25.- Seleccione la hoja de eventos y haga doble clic en el evento OnClick e introduzca el siguiente código.

	Componente	**Código**
Opción	**&Uso**;	Panel1->Visible=true;
Evento	***OnClick***	Invalidate();
Opción	**a&Cerca de…**	Panel2->Visible=true;
Evento	***OnClick***	

26.- Seleccione la forma haciendo un clic en ella, con el inspector de objetos en la hoja de eventos haga doble clic en el evento que se marca e introduzca el código asociado.

Componente	**Código**	**Acción**

Form1
 	
Evento
OnMouseDown
	Canvas->MoveTo(X,Y);
	Canvas->Pen->Color=clAqua;
	Canvas->Ellipse(X,Y,X,Y);
	Canvas->Ellipse(X,Y,X+1,Y+1);
	Canvas->Ellipse(X,Y,X+2,Y+2);
	Canvas->Ellipse(X,Y,X+3,Y+3);
	Canvas->Ellipse(X,Y,X+4,Y+4);
	old_x=X;
	old_y=Y;
	Pinta_Coordenada(old_x,old_y);
		Se selecciona un punto en la pantalla y se remarca de manera que no se vea un solo pixel, sino un grupo de pixeles.
		
		Se hace una llamada a la función que pintará los valores numéricos de la coordenada.

Form1 Evento	Float X1,Y1; X1=(float(X-(ClientWidth/2))/30.0); Y1=(float((ClientHeight/2)-Y)/30.0); Sprintf(text," ",X1,Y1);	Se obtienen los valores de la posición del mouse mientras este se esta moviendo. Se realizan los cálculos en base

OnMouseMove	Canvas->TextOut(5,5,text); Canvas->TextOut(0,0,text); Canvas->TextOut(5,10,text); Canvas->Font->Color=clAqua; Sprintf(text,"X= %2.1f Y= %2.1f ",X1,Y1); Canvas->TextOut(5,5,text); Canvas->Font->Color=clWhite;	al valor en pixeles de la pantalla visual en la que se esta trabajando, se cambia de valor de pixeles a valor de unidades de medida. Los valores obtenidos en las operaciones anteriores son desplegados en la pantalla visual.
Form1 Evento OnPaint	Ejes();	Se realiza una llamada a la función que marca las graduaciones de las coordenadas.
Form1 Evento OnReSize	Invalidate();	Borra lo que se tenia para el ajuste de tamaño.

Hasta ahora tenemos integrado nuestro modulo con la siguiente presentación:

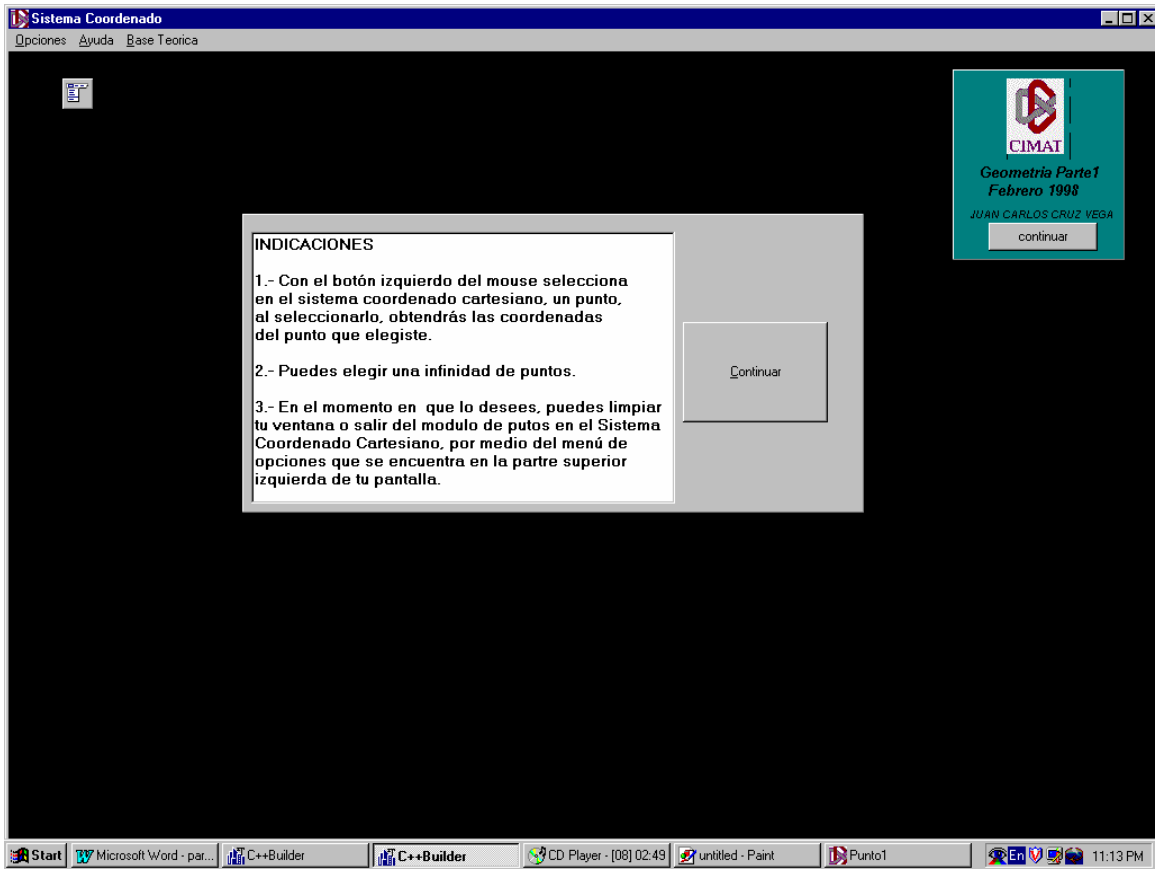


FIG. 4.1.9 Conformación del módulo No. 1.

Pero aun falta por anexar a nuestro módulo para que funcione como lo planeamos, por lo que lo complementaremos con lo siguiente:

Nota: La elaboración y detalles del Hipertexto asociado a cada módulo se presentará mas adelante en este capítulo.

27.- Seleccione el editor de código y al final de la funciones ya capturadas introduzca las siguientes funciones con las que complementará el modulo de programa.

```
//---------------------------------------------------------------------------------------------------------
//Función:       Ejes();
//Parámetros:    Ninguno.
//Objetivo:      Pintar los ejes coordenados y sus graduaciones en la ventana visual.
//---------------------------------------------------------------------------------------------------------
void __fastcall TForm1::Ejes()
{
 int i,j,k;
 int Ex=30;
 Canvas->Pen->Style=psSolid;
 Canvas->Pen->Color=clSilver;
 LINEA(ClientWidth/2,0,ClientWidth/2,ClientHeight);
 LINEA(0,ClientHeight/2,ClientWidth,ClientHeight/2);
 k=0;
 for(i=1,j=0;i<=13,j<17;i++,j++)
   {
LINEA((ClientWidth/2)+1,(ClientHeight/2)+k,(ClientWidth/2)+5,(ClientHeight/2)+k);
LINEA((ClientWidth/2)+1,(ClientHeight/2)-k,(ClientWidth/2)+5,(ClientHeight/2)-k);
LINEA((ClientWidth/2)+k,(ClientHeight/2)+1,(ClientWidth/2)+k,(ClientHeight/2)+5);
LINEA((ClientWidth/2)-k,(ClientHeight/2)+1,(ClientWidth/2)-k,(ClientHeight/2)+5);
 Canvas->Font->Size=7;
 Canvas->Font->Pitch=fpDefault;
 Canvas->Font->Color=clWhite;
 Canvas->TextOut((ClientWidth/2)+5,((ClientHeight/2)-30)-k,i);
 Canvas->TextOut((ClientWidth/2)+5,((ClientHeight/2)+30)+k,-i);   // Numeración en Y
 Canvas->TextOut((ClientWidth/2)+k,(ClientHeight/2)+5,j);         // Numeración en X
 Canvas->TextOut((ClientWidth/2)-k,(ClientHeight/2)+5,-j);
    k+=Ex;
  }
  for(i=2;i<ClientWidth;i+=30)
    for(j=0;j<ClientHeight;j+=30)
   {
     Canvas->Pen->Color=clTeal;
     LINEA(i,j,i,j);
     LINEA(i,j,i+1,j+1);
     LINEA(i,j,i-1,j-1);
   }
}
//---------------------------------------------------------------------------------------------------------
```

```
//---------------------------------------------------------------------------------------------------------
//Función:        Pinta_Coordenada(ina A, int B);
//Parámetros:   (int A, int B) Valor en pixeles de la dirección en que se marcó el punto.
//Objetivo:       Pintar los valores numéricos de la coordenada en la ventana visual.
//---------------------------------------------------------------------------------------------------------
void __fastcall TForm1::Pinta_Coordenada(int A,int B)
{
 int W=15;
 float X1,Y1;
 X1=(float(A-(ClientWidth/2))/30.0);
 Y1=(float((ClientHeight/2)-B)/30.0);
 Canvas->Pen->Style=psDashDot;//InsideFrame;//
 Canvas->Pen->Color=clFuchsia;
 LINEA(A,B,A,(ClientHeight/2));
 LINEA(A,B,(ClientWidth/2),B);
 Canvas->Pen->Color=clWhite;
 sprintf(text,"(%.1f ,%.1f )",X1,Y1);
 sign =(B>(ClientHeight/2)+1 ? 1 : -1 );
 Canvas->TextOut(A+(W*sign),B+(W*sign),text);
}
/*---------------------------------------------------------------------------------------------------------
Función:        LINEA(int P1,int P2,int P3,int P4);
Parámetros:     (int P1,int P2,int P3,int P4)
                Valor en pixeles de la dirección en que se marcó el punto.
Objetivo:       Pintar un pixel o una serie de ellos en la ventana visual según los
                parámetros recibidos.
---------------------------------------------------------------------------------------------------------*/
void __fastcall TForm1::LINEA(int P1,int P2,int P3,int P4)
{
  Canvas->MoveTo(P1,P2);
  Canvas->LineTo(P3,P4);
}
//---------------------------------------------------------------------------
```

28.- Seleccione el editor de código con el botón derecho del mouse para obtener el menú de utilidades, seleccione la opción Open Source/Header File para obtener la hoja *.h de nuestra aplicación, en ella anexaremos las líneas siguientes:

```
private:          // User declarations
      void __fastcall Ejes();
      void __fastcall Pinta_Coordenada(int X,int Y);
      void __fastcall LINEA(int P1,int P2,int P3,int P4);
      char text[70];
      int old_x,old_y,sign;
public:          // User declarations
```

29.- Seleccione File ⇒ Save Project As… y obtendrá una ventana de dialogo en la que podrá darle nombre a la aplicación.

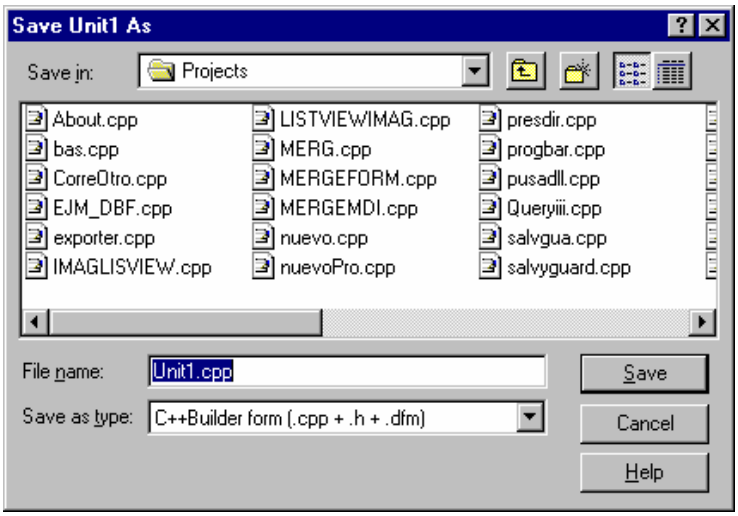


FIG. 4.1.10 Almacenamiento en disco de un módulo.

En esta parte se almacenan los módulos .cpp .h y .dmf

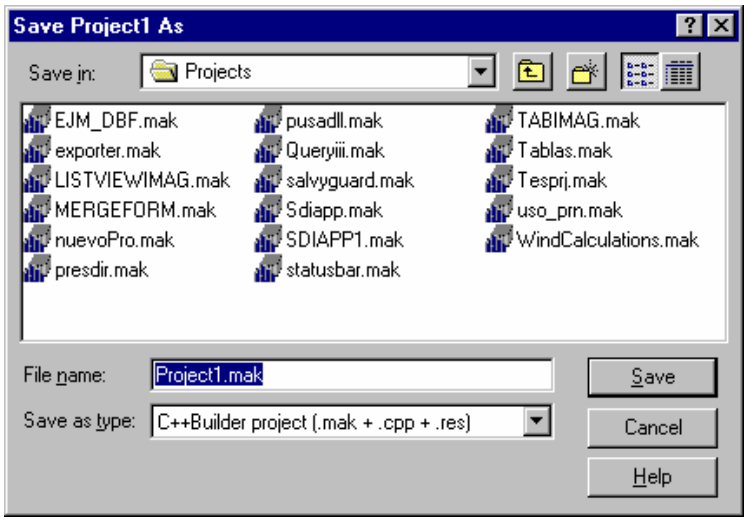


FIG. 4.1.11 Almacenamiento en disco de un proyecto.

En esta parte se almacenan los módulos .mak, .cpp del project manager, y .res

30.- Ahora puede presionar la tecla de función F9 para ejecutar la aplicación.

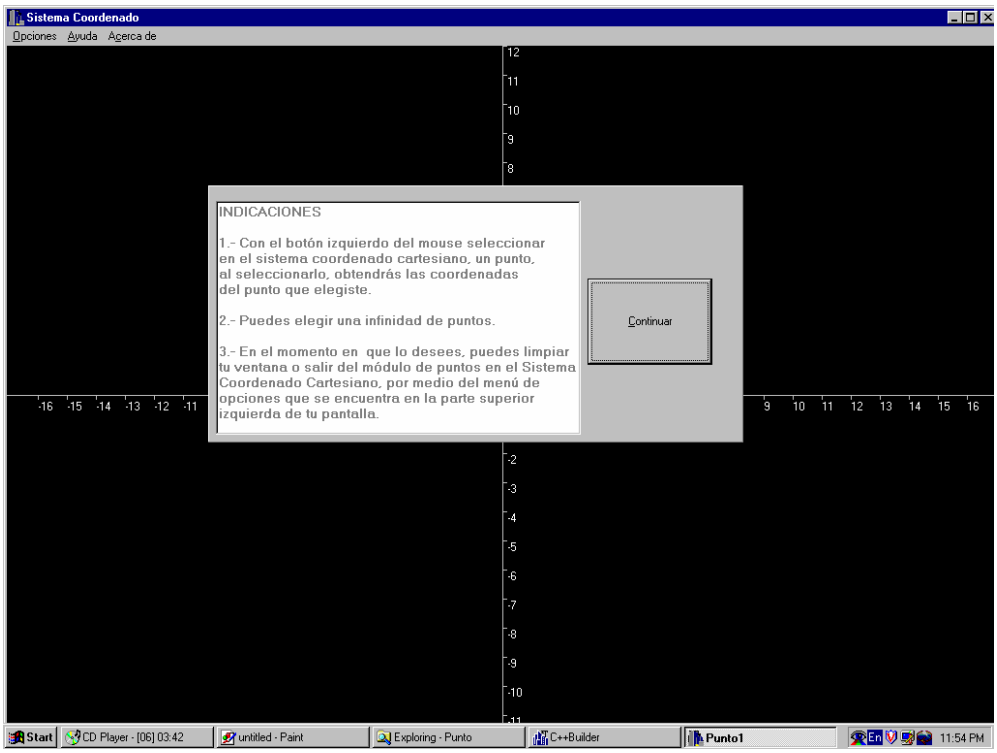


FIG. 4.1.12 Ejecución del módulo No. 1.

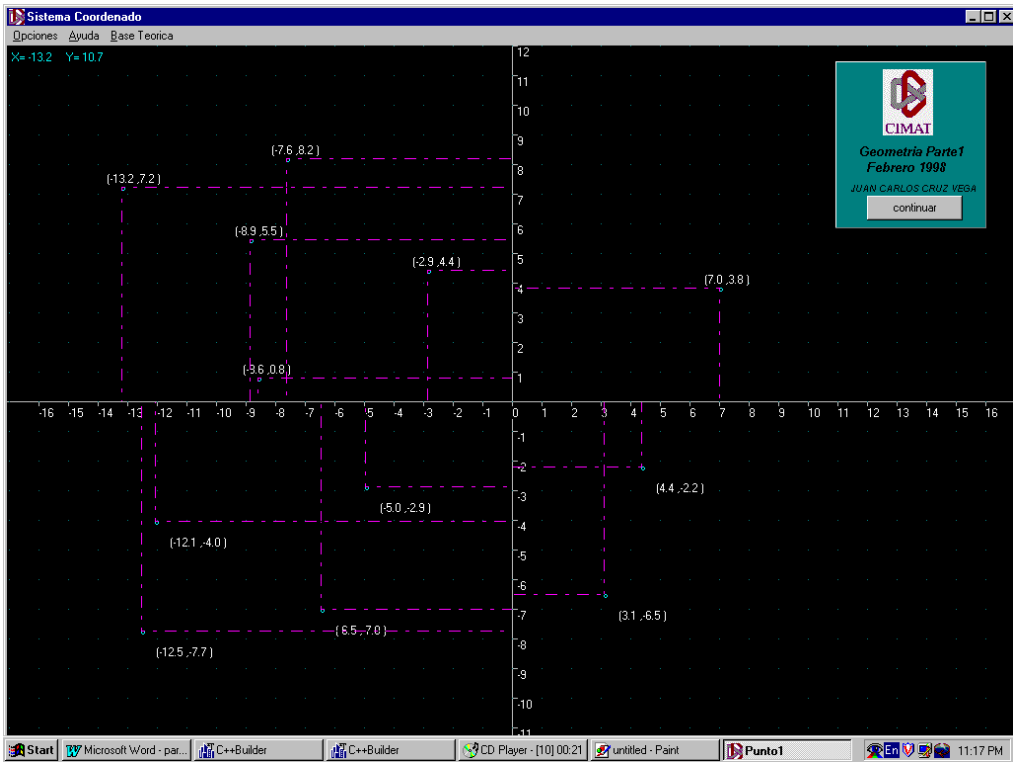


FIG. 4.1.13 Ejecución del módulo No. 1.

4.2 SEGUNDO MODULO INTERACTIVO (Programa Xotra.Cpp)

Objetivo: Conociendo el uso de los puntos en las coordenadas cartesianas, el paso siguiente es conocer la distancia que existe entre dos puntos; así que me apoyare en ese conocimiento para este módulo en el que presentaré la distancia existente entre dos puntos.

Estrategia: Elaboración de un módulo en el cual el usuario mediante el mouse coloque 2 puntos aleatorios en el sistema coordenado, y el programa muestre según los puntos marcados las coordenadas de los puntos así como la distancia que existe entre ellos.

Presentación del texto relacionado al tema mediante un Hipertexto asociado a la aplicación final.

Para la elaboración del segundo módulo se diseña una máscara que presenta:

1.- Una ventana de diálogo que tenga las instrucciones de manejo del programa.
2.- Una ventana de diálogo que contenga información del modulo.
3.- Un menú de opciones que contenga:

a) Un Campo Opciones.

&Salir ⇒ Para finalizar el programa.
&Limpiar ⇒ Para limpiar la ventana de trabajo.

b) Un Campo
&Acerca de… ⇒ Información de la elaboración del módulo.

c) Un Campo Base Teórica. Hipertexto de lo esencial del tema.

Diseño:

La construcción del módulo se realizó mediante los siguientes pasos:

1.- Seleccionamos del menú principal File ⇒ New Application, para tener una forma sin inicializar.

2.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	" Ejes Coordenados ".
b) La propiedad ClientHeight	⇒	721
c) La propiedad ClientWidth	⇒	1024
d) La propiedad Color	⇒	" clBack "

e) La propiedad Icon ⇒ BildCim.ico
f) La propiedad Left ⇒ 0
g) La propiedad Top ⇒ 0

Y lo que veremos será la ventana en la cual tendremos el control del programa:

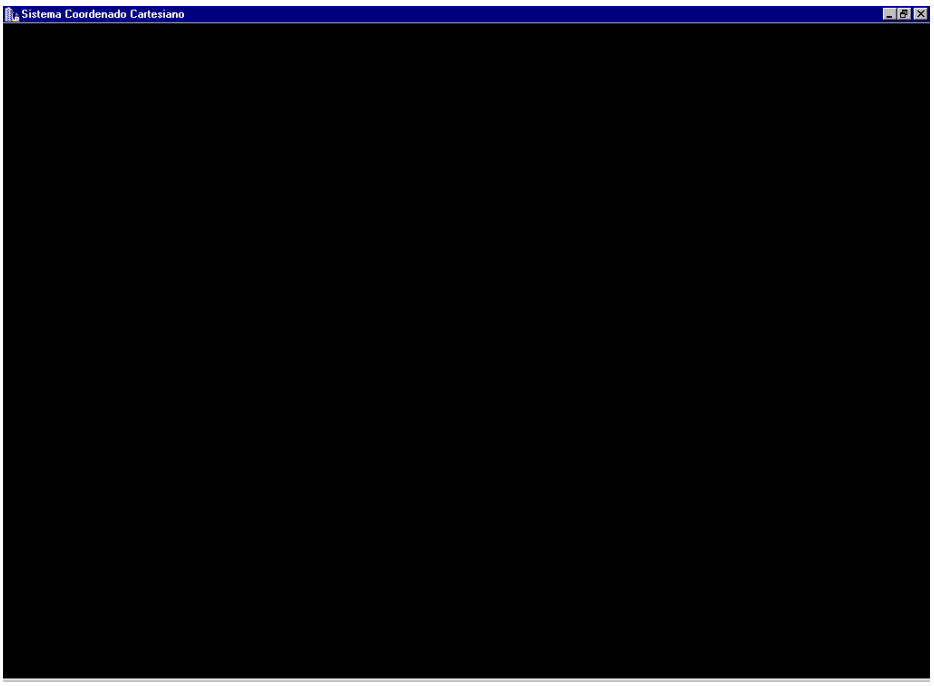


FIG. 4.2.1 Ventana de trabajo modulo No. 1.

3.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

4.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	161
c) La propiedad Width	⇒	417
d) La propiedad Color	⇒	“ clBtnFace “
e) La propiedad Left	⇒	264
f) La propiedad Top	⇒	8

5.- De la paleta de componentes Standard seleccione un objeto Button 

y colóquelo en el Panel.

6.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Continuar.
b) La propiedad Height	⇒	57
c) La propiedad Width	⇒	59
d) La propiedad Left	⇒	352
e) La propiedad Top	⇒	48

Haga doble clic en el botón Continuar e introduzca el siguiente código.

```
Panel1->Visible=false;
```

7.- De la paleta de componentes Standard seleccione un objeto Memo  y colóquelo en el panel.

8.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	145
b) La propiedad Width	⇒	337
c) La propiedad Left	⇒	8
d) La propiedad Top	⇒	8
e) La propiedad Lines	⇒	Doble clic

Aparecerá el editor de Líneas, en el se tecleará el siguiente Texto.

Distancia

Con el botón izquierdo del mouse puedes seleccionar
arbitrariamente dos puntos. de los cuales se calculará
su distancia siguiendo la fórmula
d = Raiz Cuadrada [$(Y2 - Y1)^2 + (X2 - X1)^2$]

Podrás trazar el número de rectas que desees.

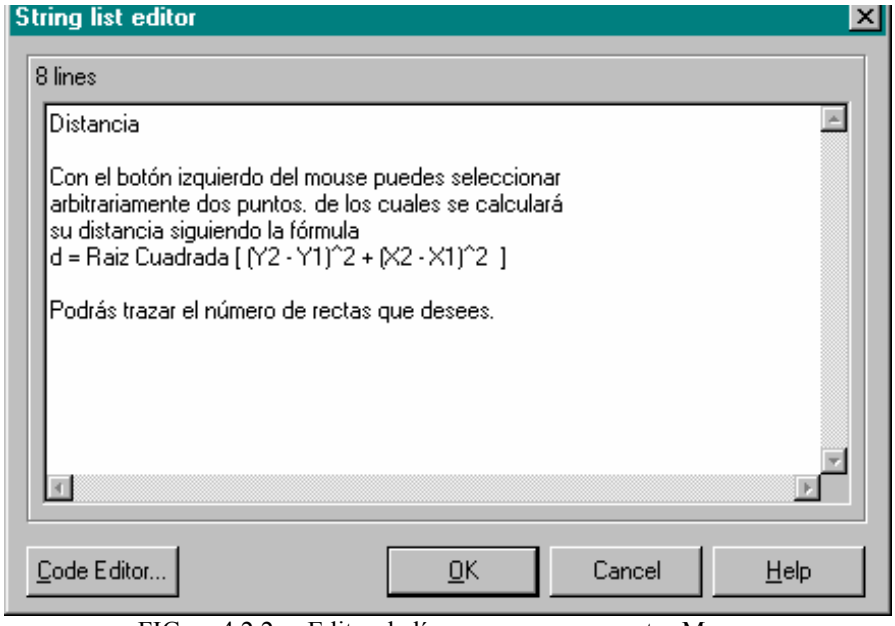


FIG. 4.2.2 Editor de líneas para componentes Memo.

9.- Selecciona el botón Ok para cerrar la ventana de dialogo.

10.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

11.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	169
c) La propiedad Width	⇒	153
d) La propiedad Color	⇒	“ clTeal “
e) La propiedad Left	⇒	840
f) La propiedad Top	⇒	16

12.- De la paleta de componentes Additional seleccione un objeto Image  y colóquelo en el Panel2.

13.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	73
b) La propiedad Width	⇒	57
c) La propiedad Left	⇒	48
f) La propiedad Top	⇒	8
e) La propiedad Stretch	⇒	True
f) La propiedad Picture	⇒	Doble clic

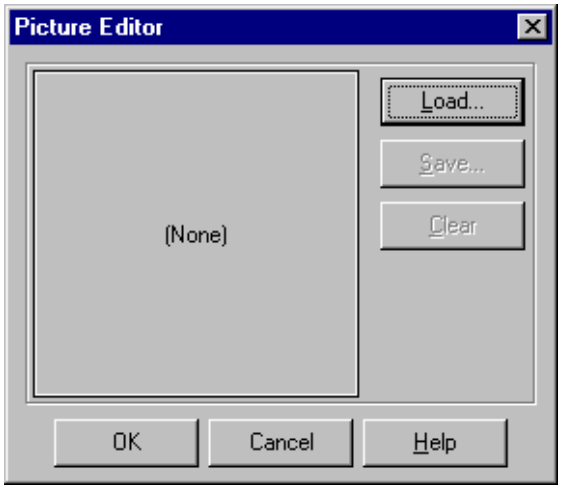


FIG. 4.2.3 Presentador de Imágenes.

Presione el botón Load y obtendrá una caja de dialogo para seleccionar el icono deseado o podrá buscarlo en caso de no encontrarse en el directorio mostrado.

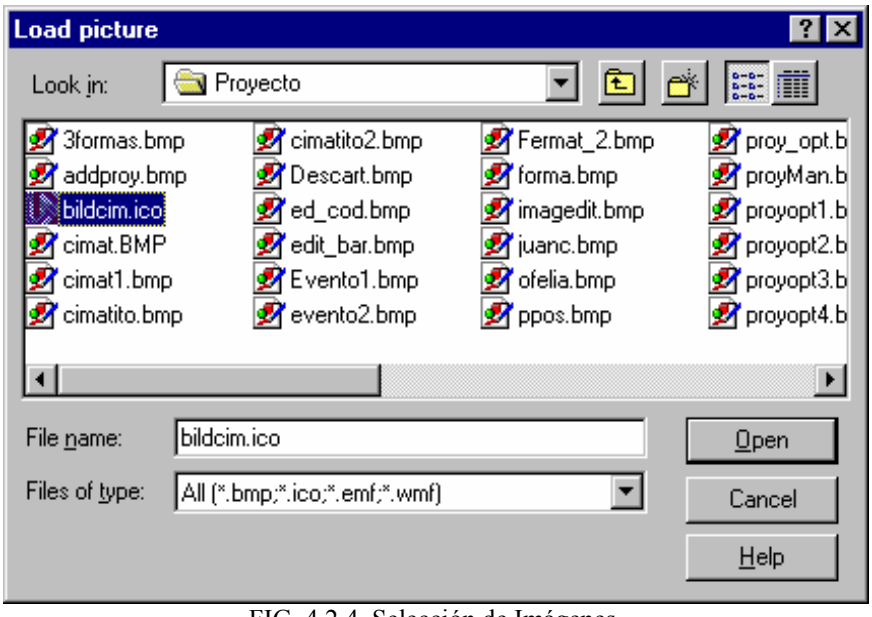


FIG. 4.2.4 Selección de Imágenes.

Seleccione la imagen deseada y oprima el botón Open.

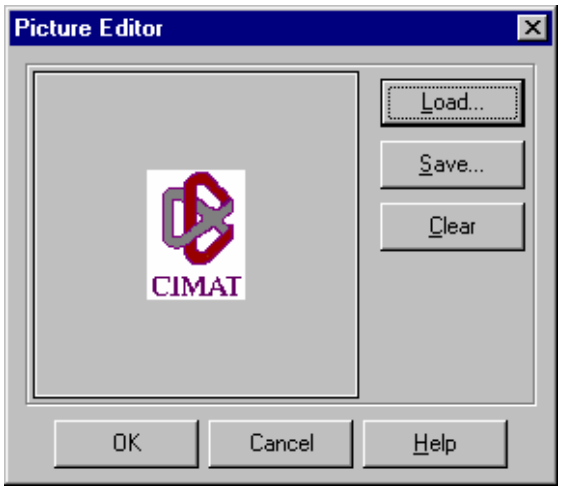


FIG. 4.2.5 Selección de Imágenes.

Oprima el botón Ok.

14.- De la paleta de componentes Standard seleccione un objeto Button 

y colóquelo en el Panel2.

15.- Modifiquemos los siguientes atributos:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Continuar
b) El evento OnClick	⇒	Panel2->Visible=false;

16.- De la paleta de componentes Standard seleccione un Label 

y colóquelo en el Panel2 con otras dos copias.

17.- Modifiquemos los siguientes atributos:

Label1.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Geometría Parte 1.
b) La propiedad Height	⇒	16
c) La propiedad Width	⇒	110
d) La propiedad Color	⇒	“ clTeal “
e) La propiedad Left	⇒	32

f) La propiedad Top ⇒ 80
g) La propiedad Font Style ⇒ Bold Italic
h) La propiedad Font Size ⇒ 10

Label2.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	FEBRERO DE 1998
b) La propiedad Height	⇒	16
c) La propiedad Width	⇒	80
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	40
f) La propiedad Top	⇒	96
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	10

Label3.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	JUAN CARLOS CRUZ VEGA
b) La propiedad Height	⇒	11
c) La propiedad Width	⇒	128
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	24
f) La propiedad Top	⇒	120
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	7

18.- De la paleta de componentes Standard seleccione un objeto Main Menu  y colóquelo en la forma.

19.- haga doble clic en este componente y obtendrá el diseñador de menús:

20.- Con el inspector de objetos cambie las propiedades:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Opciones + Enter
b) La propiedad Caption	⇒	&Acerca de… + Enter

c) La propiedad Caption ⇒ &Base teórica + Enter

21.- Seleccione la hoja de eventos y haga doble clic en el evento OnClick e introduzca el siguiente código.

Componente	**Código**
Opción **&Base Teórica**;	Application->HelpCommand(HELP_CONTENTS,0);
Evento ***OnClick***	Invalidate();

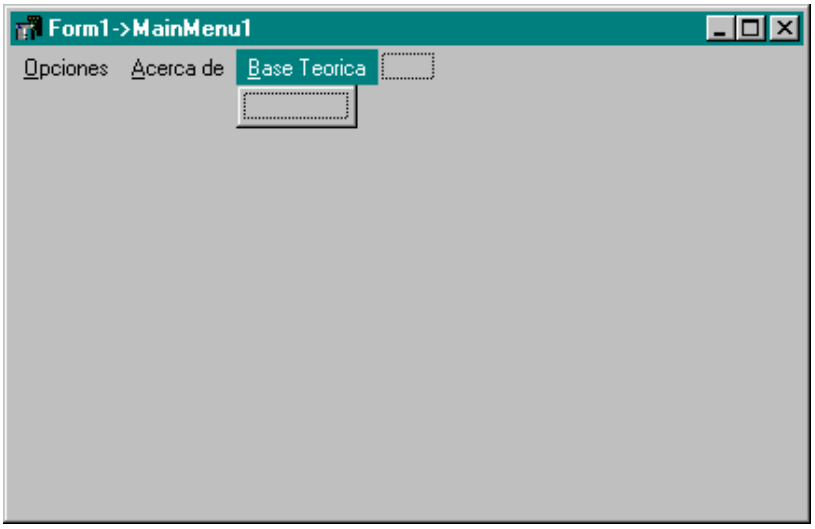


FIG. 4.2.6 Diseñador de menús.

22.- En el sub menú Opciones anexar 2 campos más.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Limpiar
a) La propiedad Caption	⇒	&Salir

De manera que tenga el siguiente resultado:

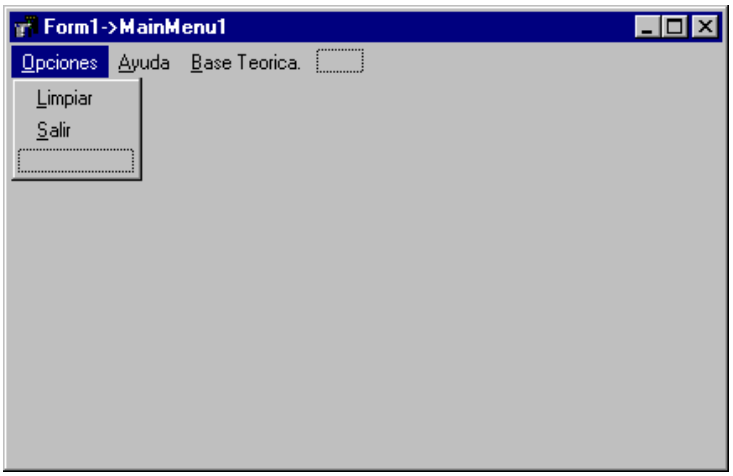


FIG. 4.2.7 Opciones de Sub menú.

23.- Seleccione la hoja de eventos y haga doble clic en el evento OnClick e introduzca el siguiente código.

Componente	**Código**
Opción **&Limpiar**;	Invalidate();
Evento ***OnClick***	
Opción **&Salir**;	Close();
Evento ***OnClick***	Exit(1);

24.- En el sub menú Acerca de… colocar el siguiente código en el evento **OnClick**:

Propiedad		Valor o Texto
a) Evento ***OnClick***	⇒	Panel2->Visible=true;

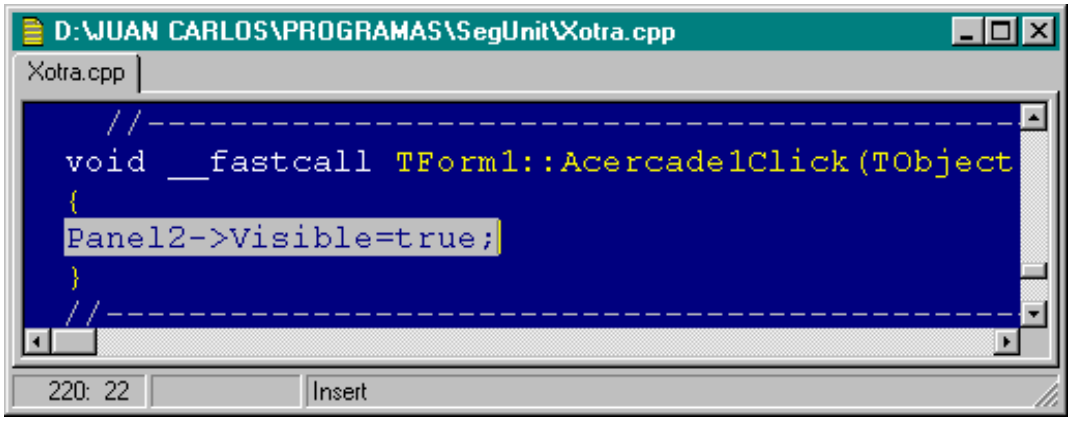


FIG. 4.2.8 Eventos asociados a los componentes.

25.- Seleccione la forma haciendo un clic en ella, con el inspector de objetos en la hoja de eventos haga doble clic en el evento que se marca e introduzca el código asociado.

Componente	**Código**	**Acción**
Form1 Evento ***OnMouseDown***	Obten_Puntos(X,Y); Panel1->Visible=false; Panel2->Visible=false;	Se realiza una llamada a una función que obtiene los puntos seleccionados para el cálculo de la distancia. Me aseguro que las ventanas de mensaje creadas no estén visibles en este momento.
Form1 Evento ***OnMouseMove***	Float X1,Y1; X1=(float(X-(ClientWidth/2))/30.0); Y1=(float((ClientHeight/2)-Y)/30.0); Sprintf(text," ",X1,Y1); Canvas->TextOut(ClientWidth-130,ClientHeight-40,text); Canvas->TextOut(ClientWidth-130,ClientHeight-50,text); Canvas->TextOut(ClientWidth-130,ClientHeight-60,text); Canvas->Font->Color=clAqua; Canvas->Font->Size=12; Sprintf(text,"X= %2.1f Y= %2.1f ",X1,Y1); Canvas->TextOut(ClientWidth-150,ClientHeight-50,text); Canvas->Font->Color=clWhite;	Se obtienen los valores de la posición del mouse mientras este se esta moviendo por la pantalla. Se realizan los cálculos en base al valor en pixeles de la pantalla visual en la que se esta trabajando, se cambia de valor de pixeles a valor de unidades de medida. Los valores obtenidos en las operaciones anteriores son desplegados en la pantalla visual.
Form1 Evento OnPaint	Ejes();	Se redibujan los ejes y sus graduaciones.
Form1 Evento OnReSize	Invalidate();	Borra lo que se tenía para el ajuste de tamaño.

Hasta ahora tenemos integrado nuestro módulo con la siguiente presentación:

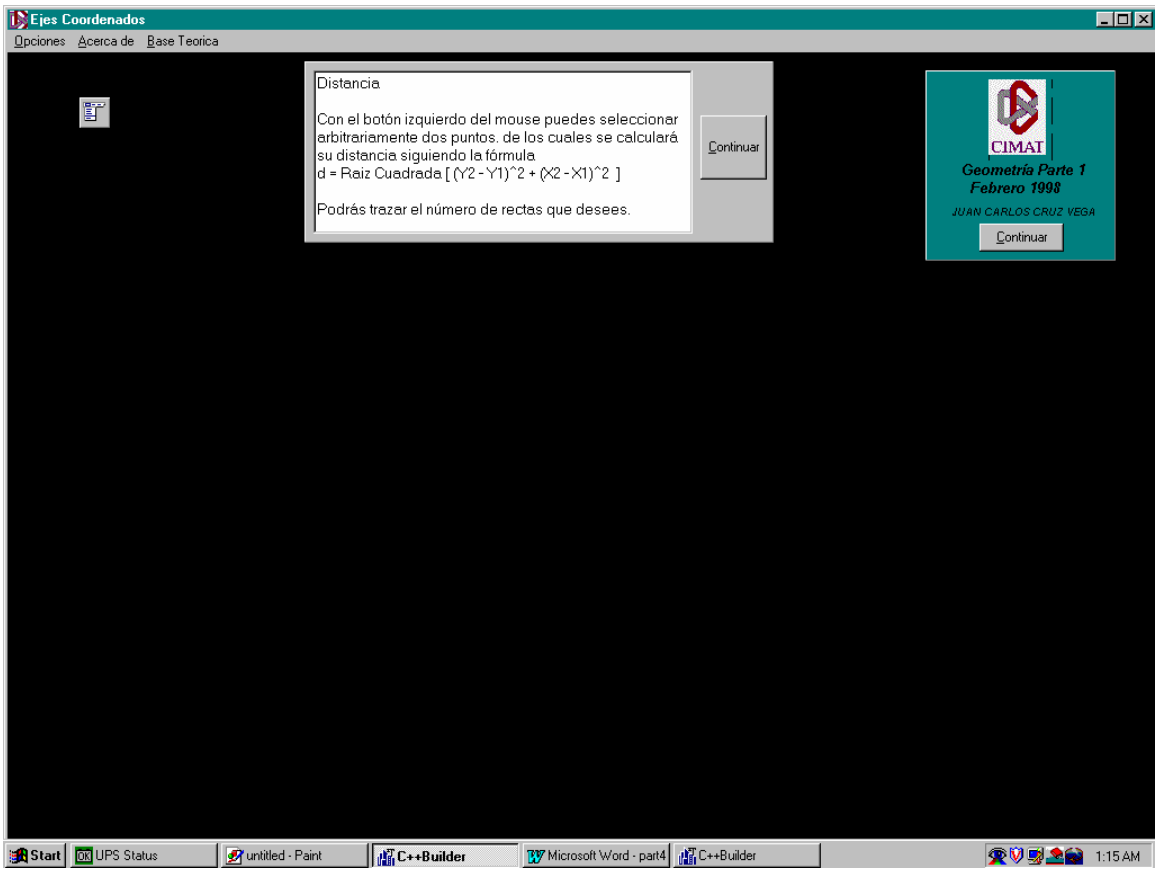


FIG. 4.2.9 Componentes conjuntos para una aplicación.

26.- Seleccione el editor de código y al final de la funciones ya capturadas introduzca las funciones con las que complementará el modulo de programa, según el Apéndice b.

27.- Seleccione File ⇒ Save Project As… y obtendrá una ventana de dialogo en la que podrá darle nombre a la aplicación.

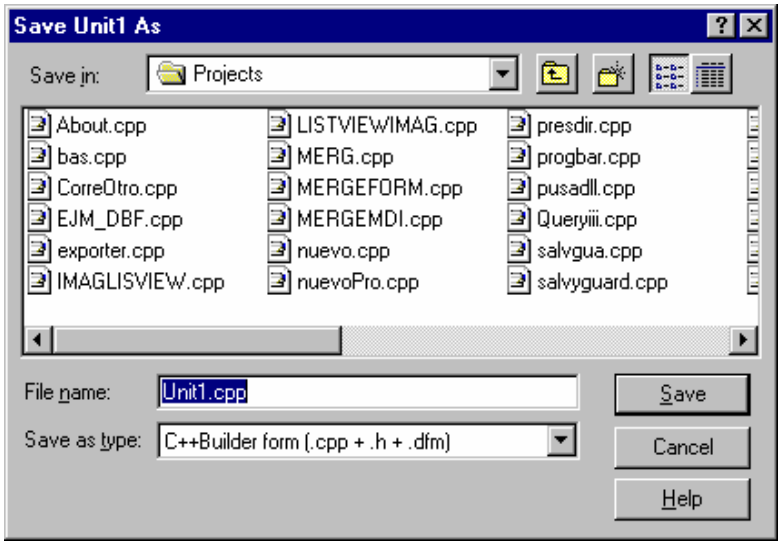


FIG. 4.2.10 Almacenamiento en disco de un módulo.

En esta parte se almacenan los módulos .cpp .h y .dmf

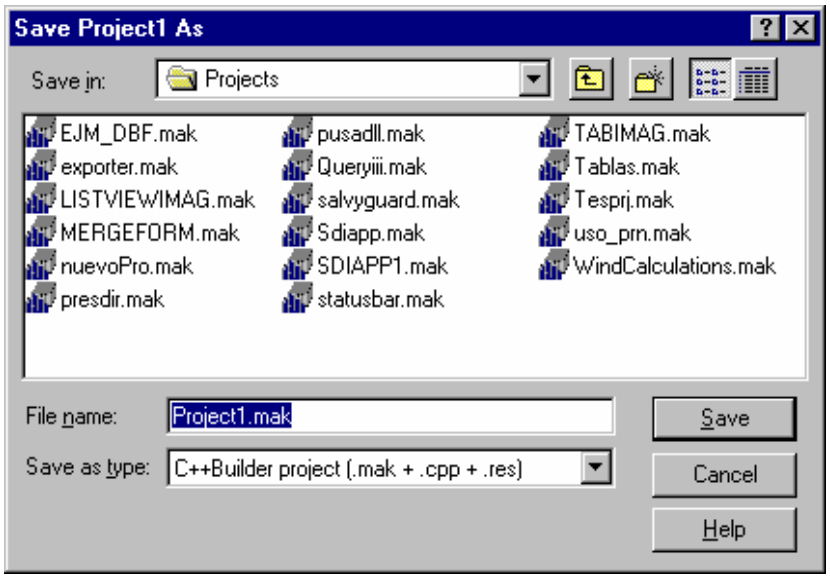


FIG. 4.2.11 Almacenamiento en disco de un proyecto.

En esta parte se almacenan los módulos .mak, .cpp del project manager, y .res

28.- Ahora puede presionar la tecla de función F9 para ejecutar la aplicación.

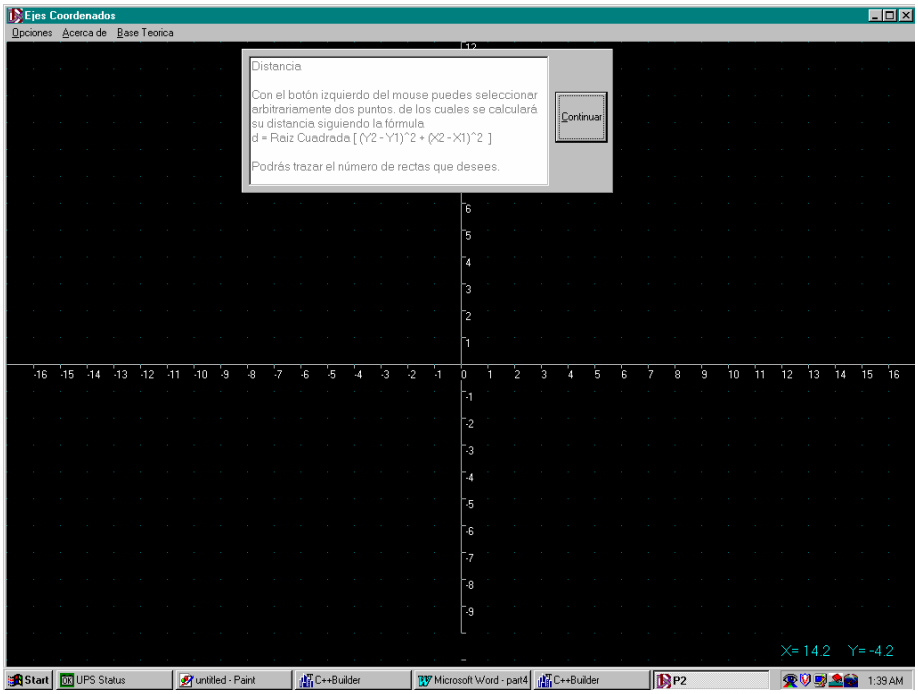


FIG. 4.2.12 Ejecusión del Módulo No.2.

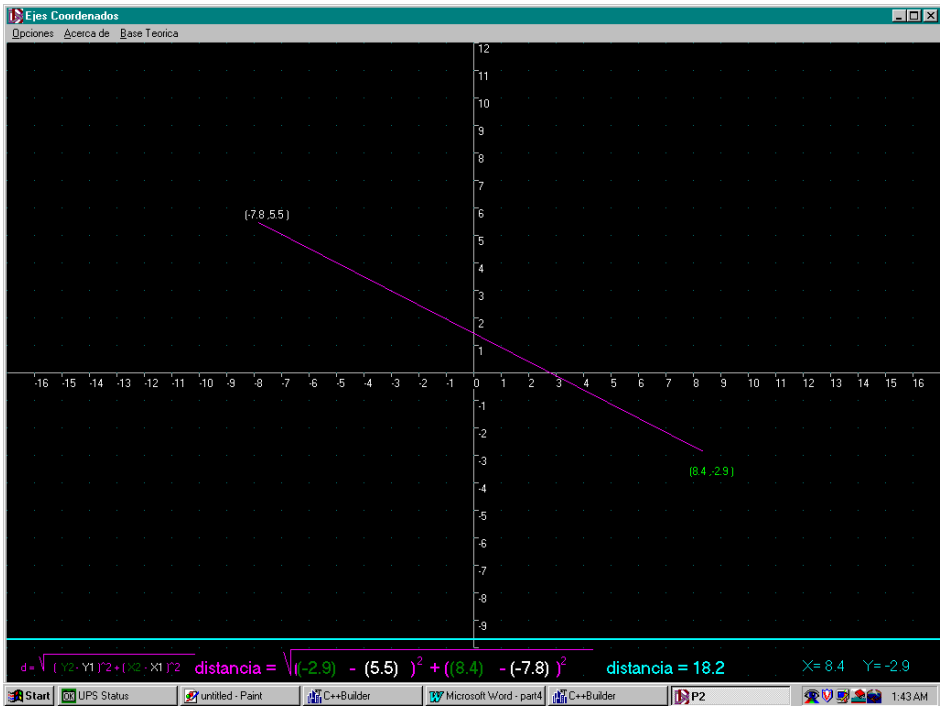


FIG. 4.2.13 Ejecusión del Módulo No.2.

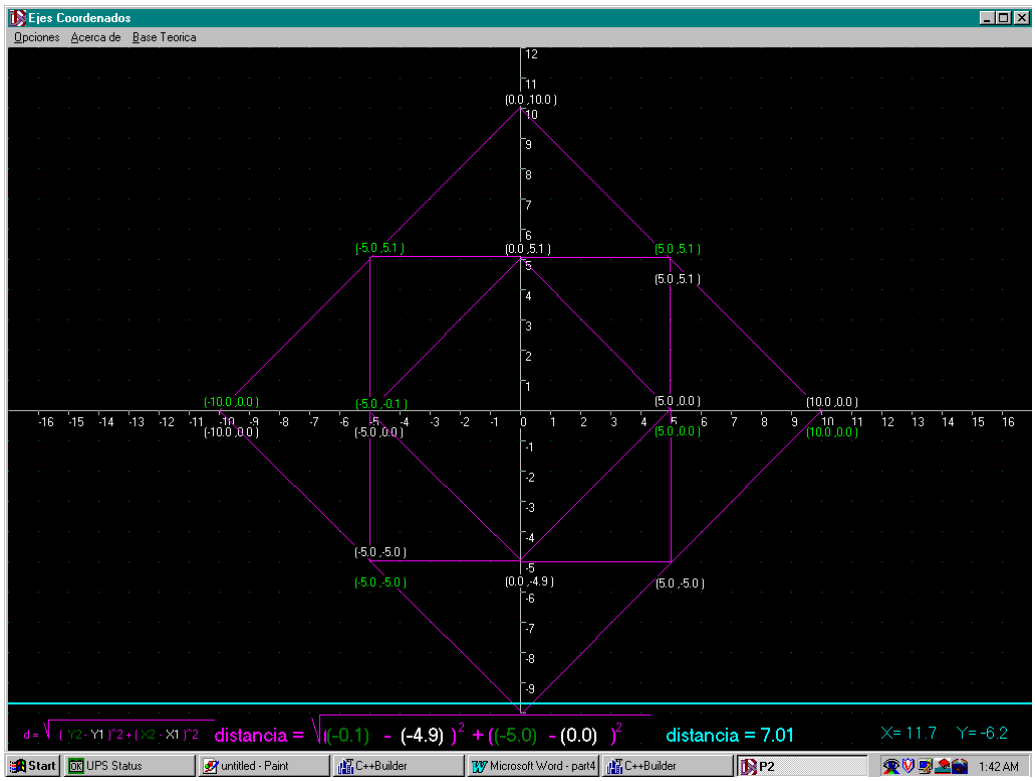


FIG. 4.2.14 Ejecución del Módulo No.2.

4.3 TERCER MODULO INTERACTIVO
(Programa Pendient.Cpp)

Objetivo: Aplicar los conocimientos antes adquiridos tales como las coordenadas de un punto, y la distancia existente entre dos distintos puntos; y ahora ampliar este conocimiento para conocer con los mismos datos que ya tenemos de 2 puntos distintos, cual es el ángulo que forma la línea recta (o segmento de línea) que une ambos puntos con el eje X.

Estrategia: Utilizando los datos y conocimientos obtenidos con los módulos anteriores, aplicar una sencilla fórmula matemática para descubrir el ángulo de inclinación o pendiente de una recta que pasa por dos puntos.
Con los módulos anteriores conocimos una parte importante del estudio de la Geometría Analítica ahora lo ampliaremos un poco más.
Presentación del texto relacionado al tema mediante un Hipertexto asociado a la aplicación final.

Para la elaboración del tercer módulo se diseña una máscara que presenta:

1.- Una ventana de diálogo que tenga las instrucciones de manejo del programa.
2.- Una ventana de diálogo que contenga información del modulo.
3.- Una ventana de diálogo que contenga aviso de Pendiente Indeterminada.
4.- Un menú de opciones que contenga:

a) Un Campo Opciones.
   &Salir ⇒ Para finalizar el programa.
   &Limpiar ⇒ Para limpiar la ventana de trabajo.
b) Un Campo
   &Ayuda ⇒ Información sobre el uso del módulo.
c) Un campo
   &Acerca de… ⇒ Información de la elaboración del módulo.
d) Un Campo Base Teórica. Hipertexto de lo esencial del tema.

Diseño:
La construcción del módulo se realizó mediante los siguientes pasos:

1.- Seleccionamos del menú principal File ⇒ New Application, para tener una forma sin inicializar.

2.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	" Ejes Coordenados ".
b) La propiedad ClientHeight	⇒	721

c) La propiedad ClientWidth ⇒ 1024
d) La propiedad Color ⇒ " clBack "
e) La propiedad Icon ⇒ BildCim.ico
f) La propiedad Left ⇒ 0
g) La propiedad Top ⇒ 0

3.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

4.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	273
c) La propiedad Width	⇒	417
d) La propiedad Color	⇒	" clBtnFace "
e) La propiedad Left	⇒	248
f) La propiedad Top	⇒	158

5.- De la paleta de componentes Standard seleccione un objeto Button  y colóquelo en el Panel.

6.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Continuar.
b) La propiedad Height	⇒	65
c) La propiedad Width	⇒	57
d) La propiedad Left	⇒	344
e) La propiedad Top	⇒	112

Haga doble clic en el botón Continuar e introduzca el siguiente código.

```
Panel1->Visible=false;
```

7.- De la paleta de componentes Standard seleccione un objeto Memo  y colóquelo en elPanel.

8.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	241
b) La propiedad Width	⇒	313
c) La propiedad Left	⇒	16
d) La propiedad Top	⇒	16
e) La propiedad Lines	⇒	Doble clic

Aparecerá el editor de Líneas, en el se tecleará el siguiente Texto.

Distancia

Con el botón izquierdo del mouse, puedes
seleccionar arbitrariamente dos puntos
de los cuales se calculará su distancia
siguiendo la fórmula :
d = Raiz Cuadrada [(Y2 - Y1)^2 + (X2 - X1)^2]
Paralelamente se calculará la Pendiente o ángulo
de inclinación de la recta según la formula:

$$m = \frac{Y2 - Y1}{X2 - X1}$$

Puedes seleccionar varias rectas.

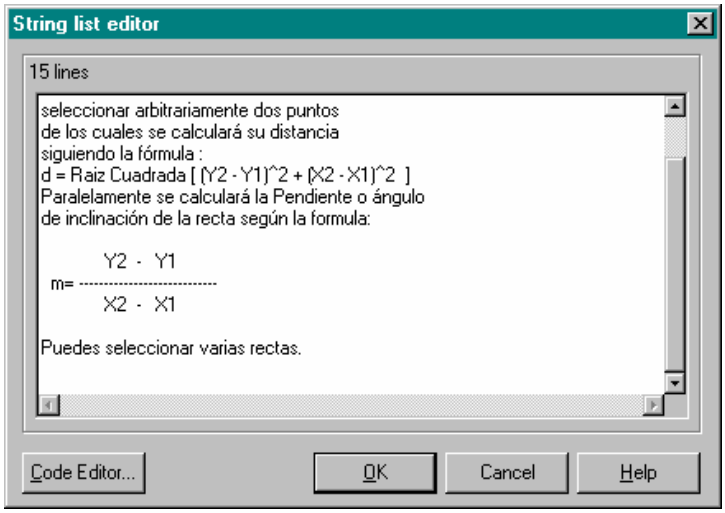


FIG. 4.3.1 Editor de líneas para componentes Memo.

9.- Selecciona el botón Ok para cerrar la ventana de dialogo.

10.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

11.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	233
c) La propiedad Width	⇒	369
d) La propiedad Color	⇒	" clBtnFace "
e) La propiedad Left	⇒	256
f) La propiedad Top	⇒	192

12.- De la paleta de componentes Standard seleccione un objeto Button  y colóquelo en el Panel2.

13.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Continuar.
b) La propiedad Height	⇒	49
c) La propiedad Width	⇒	57
d) La propiedad Left	⇒	304
e) La propiedad Top	⇒	96

Haga doble clic en el botón Continuar e introduzca el siguiente código.

```
Panel2->Visible=false;
```

14.- De la paleta de componentes Standard seleccione un objeto Memo  y colóquelo en elPanel2.

15.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	201

b) La propiedad Width ⇒ 281
c) La propiedad Left ⇒ 16
d) La propiedad Top ⇒ 16
e) La propiedad Lines ⇒ Doble clic

Aparecerá el editor de Líneas, en el se tecleará el siguiente Texto.

Pendiente Indeterminada.

Cuendo la línea recta hace una paralela con la vertical
los valores de las x para ambos puntos son iguales,
y al hacer la resta queda cero; por lo tanto la fórmula
para la pendiente quedaría de la siguiente manera:

$$m = \frac{Y2 - Y1}{0}$$

Por lo que el valor de la pendiente es:

INDETERMINADO.

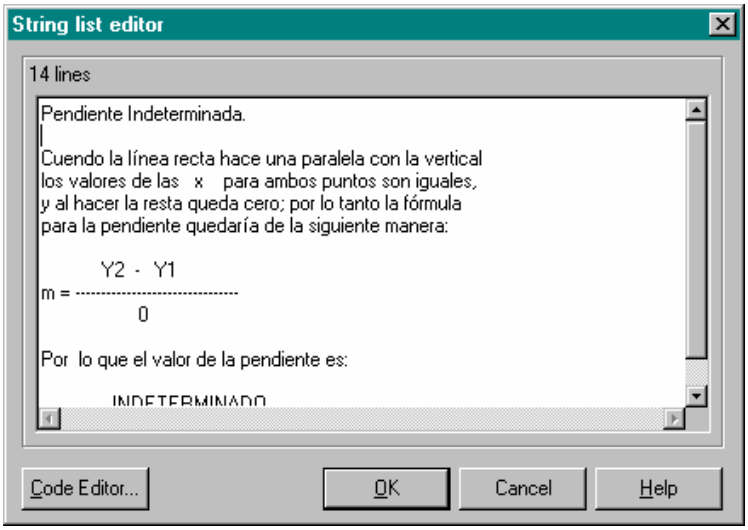


FIG. 4.3.2 Editor de líneas para componentes Memo.

16.- Selecciona el botón Ok para cerrar la ventana de dialogo.

17.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

18.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	177
c) La propiedad Width	⇒	145
d) La propiedad Color	⇒	“ clTeal “
e) La propiedad Left	⇒	856
f) La propiedad Top	⇒	16

19.- De la paleta de componentes Additional seleccione un objeto Image  y colóquelo en el Panel3.

20.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	73
b) La propiedad Width	⇒	57
c) La propiedad Left	⇒	48
f) La propiedad Top	⇒	8
e) La propiedad Stretch	⇒	True
f) La propiedad Picture	⇒	Doble clic

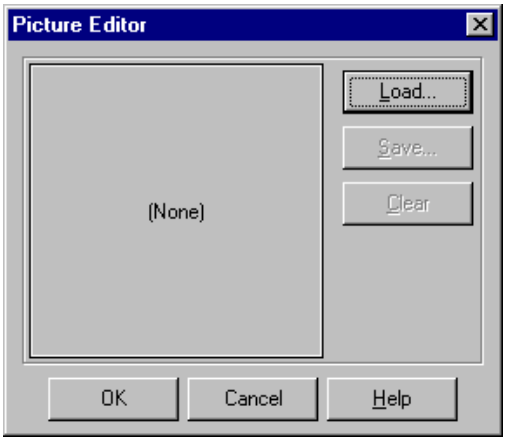


FIG. 4.3.3 Presentador de Imágenes.

Presione el botón Load y obtendrá una caja de dialogo para seleccionar el icono deseado o podrá buscarlo en caso de no encontrarse en el directorio mostrado.

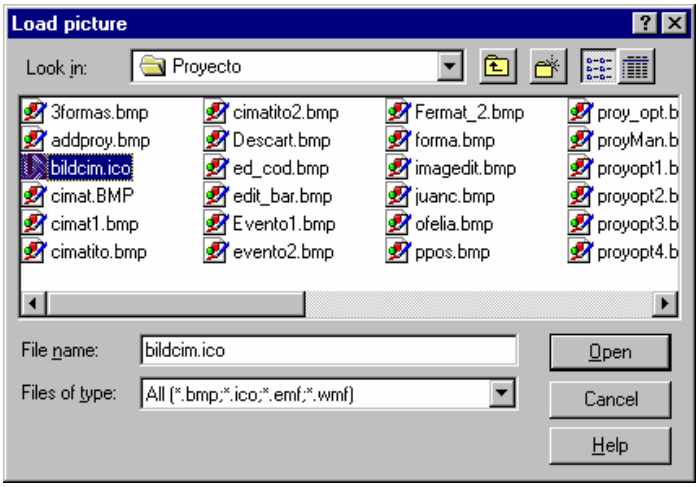


FIG. 4.3.4 Selección de Imágenes.

Seleccione la imagen deseada y oprima el botón Open.

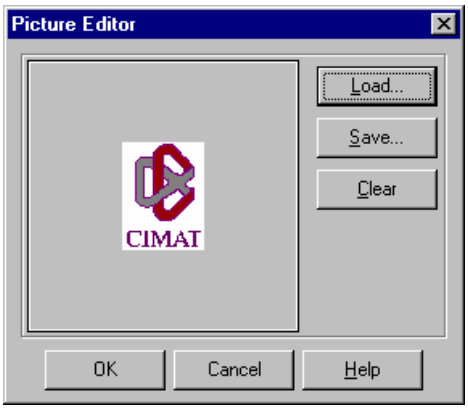


FIG. 4.3.5 Selección de Imágenes.

Oprima el botón Ok.

21.- De la paleta de componentes Standard seleccione un objeto Button 

y colóquelo en el Panel3.

22.- Modifiquemos los siguientes atributos:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Continuar
b) El evento OnClick	⇒	Panel3->Visible=false;

23.- De la paleta de componentes Standard seleccione un Label  y colóquelo en el Panel3 con otras dos copias.

24.- Modifiquemos los siguientes atributos:

Label1.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Geometría Parte 1.
b) La propiedad Height	⇒	16
c) La propiedad Width	⇒	110
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	16
f) La propiedad Top	⇒	80
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	10

Label2.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	FEBRERO DE 1998
b) La propiedad Height	⇒	16
c) La propiedad Width	⇒	80
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	24
f) La propiedad Top	⇒	96
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	10

Label3.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	JUAN CARLOS CRUZ VEGA
b) La propiedad Height	⇒	11
c) La propiedad Width	⇒	128
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	8
f) La propiedad Top	⇒	128
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	7

25.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

26.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	73
c) La propiedad Align	⇒	" alBottom "
d) La propiedad Color	⇒	" clTeal "

27.- De la paleta de componentes Standard seleccione un objeto Main Menu  y colóquelo en la forma.

28.- Haga doble clic en este componente y obtendrá el diseñador de menús:

29.- Con el inspector de objetos cambie las propiedades:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Opciones + Enter
b) La propiedad Caption	⇒	&Ayuda + Enter
c) La propiedad Caption	⇒	A&Cerca de… + Enter
d) La propiedad Caption	⇒	&Base teórica + Enter

30.- Seleccione la hoja de eventos y haga doble clic en el evento OnClick e introduzca el siguiente código.

Componente	**Código**
Opción **&Base Teórica**;	Application->HelpCommand(HELP_CONTENTS,0);
Evento ***OnClick***	Invalidate();

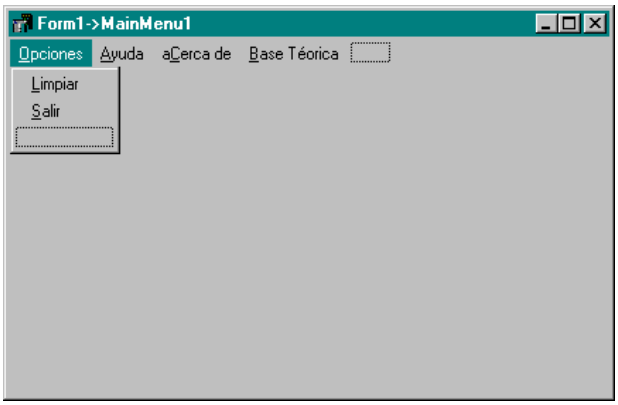


FIG. 4.3.6 Diseñador de menús.

31.- En el sub menú Opciones anexar 2 campos mas.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Limpiar
a) La propiedad Caption	⇒	&Salir

De manera que tenga el siguiente resultado:

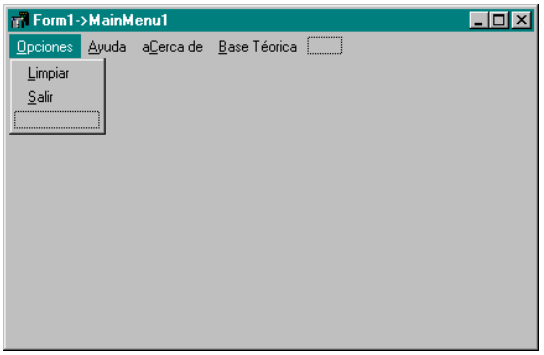


FIG. 4.3.7 Opciones de Sub menú.

32.- Seleccione la hoja de eventos y haga doble clic en el evento OnClick e introduzca el siguiente código.

Componente	**Código**
Opción **&Limpiar**;	Invalidate();
Evento ***OnClick***	
Opción **&Salir**;	Close();
Evento ***OnClick***	Exit(1);

33.- En el sub menú Ayuda colocar el siguiente código en el evento **OnClick**:

Propiedad		Valor o Texto
a) Evento ***OnClick***	⇒	Panel1->Visible=true;

34.- En el Sub menú aCerca de… colocar el siguiente código en el evento **OnClick**:

Propiedad		Valor o Texto
a) Evento ***OnClick***	⇒	Panel2->Visible=true;

35.- Seleccione la forma haciendo un clic en ella, con el inspector de objetos en la hoja de eventos haga doble clic en el evento que se marca e introduzca el código asociado.

Componente	**Código**	**Acción**
Form1 Evento ***OnMouseDown***	Obten_Puntos(X,Y);	Se realiza una llamada a una función que obtiene los puntos seleccionados para el cálculo de la distancia.
Form1 Evento ***OnMouseMove***	Float X1,Y1; X1=(float(X-(ClientWidth/2))/30.0); Y1=(float((ClientHeight/2)-Y)/30.0); Sprintf(text," ",X1,Y1); Canvas->TextOut(820,5,text); Canvas->TextOut(820,10,text); Canvas->TextOut(820,15,text); Canvas->Font->Size=12; Canvas->Font->Color=clRed; Sprintf(text,"X= %2.1f Y= %2.1f ",X1,Y1); Canvas->TextOut(820,10,text); Canvas->Font->Color=clWhite;	Se obtienen los valores de la posición del mouse mientras este se esta moviendo por la pantalla. Se realizan los cálculos en base al valor en pixeles de la pantalla visual en la que se esta trabajando, se cambia de valor de pixeles a valor de unidades de medida. Los valores obtenidos en las operaciones anteriores son desplegados en la pantalla visual.
Form1 Evento OnPaint	Ejes();	Se rredibujan los ejes y sus graduaciones.
Form1 Evento OnReSize	Invalidate();	Borra lo que se tenia para el ajuste de tamaño.

Hasta ahora tenemos integrado nuestro modulo con la siguiente presentación:

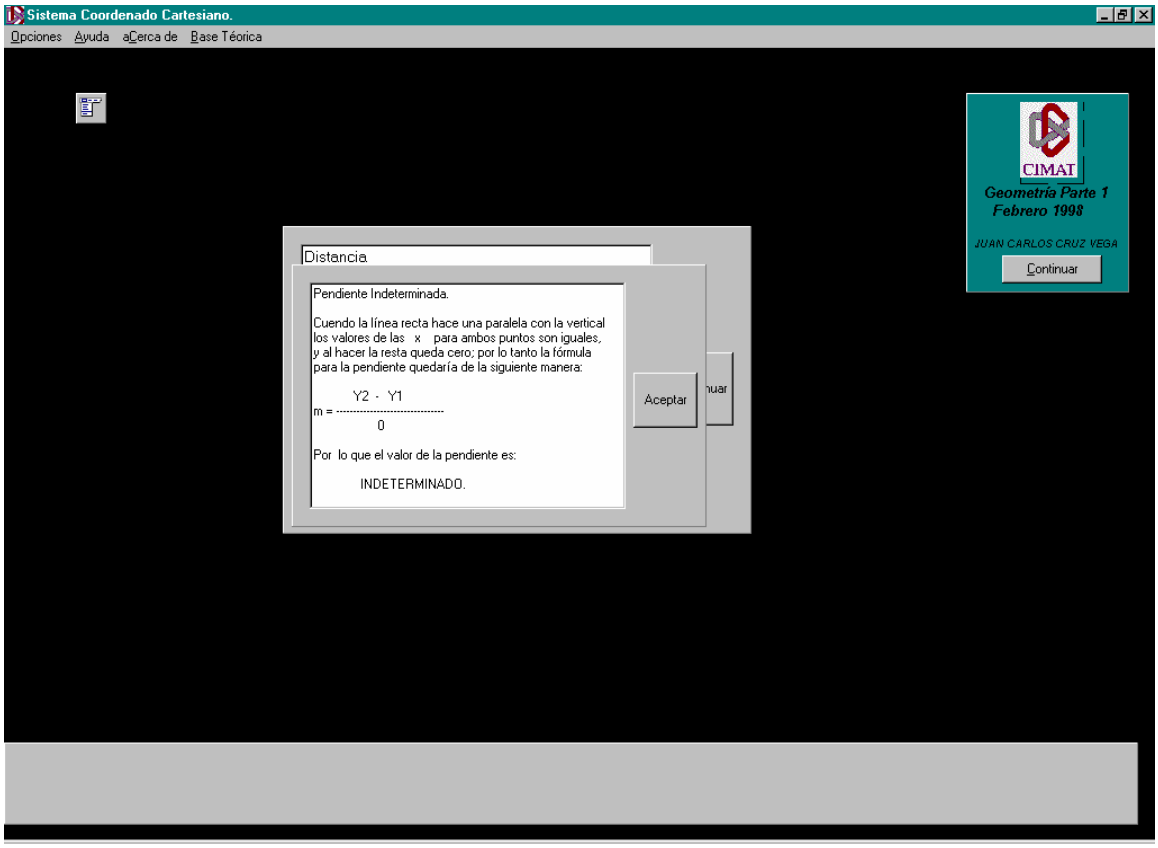


FIG. 4.3.8 Ambiente del módulo No.3.

Nota: Los pasos 21 al 25 de la sección anterior podrían estar en un archivo de cabecera ya que son similares en estos dos casos, pero existen pequeñas diferencias que se hicieron a propósito para que se construyan totalmente y se practique, posteriormente para efectos de espacio en almacenamiento y rapidez, se pueden integrar en un archivo de cabecera como una librería.

36.- Seleccione el editor de código y al final de la funciones ya capturadas introduzca las funciones con las que complementará el modulo de programa, según el Apéndice b.

37.- Seleccione File ⇒ Save Project As… para almacenar el proyecto en disco.

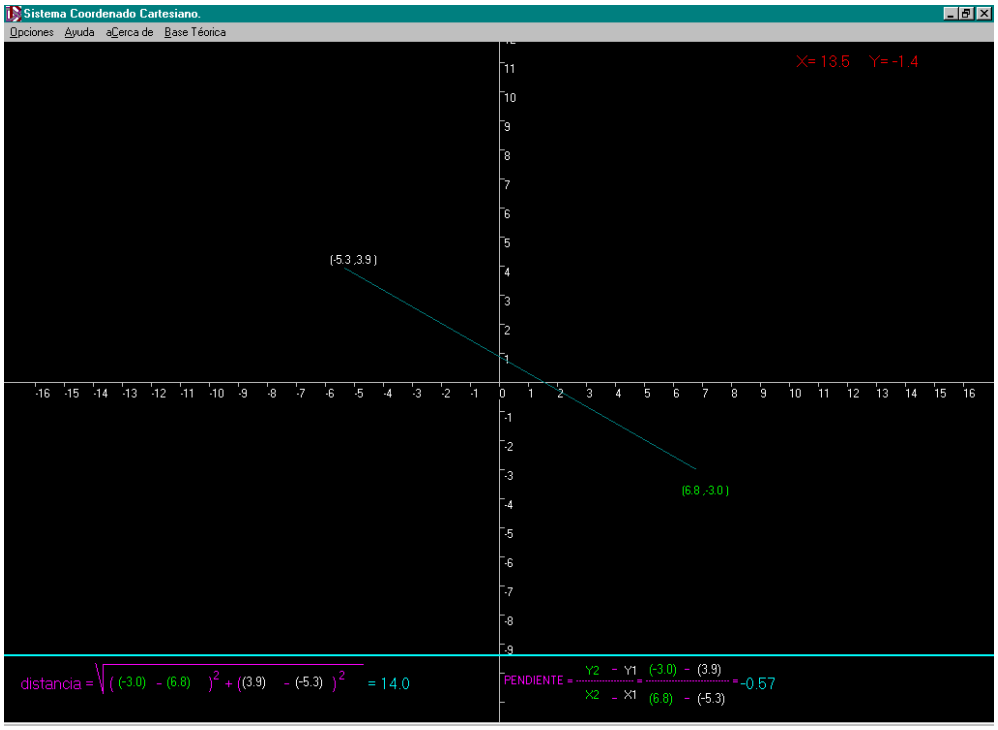


FIG. 4.3.9 Ejecución del módulo No.3.

4.4 CUARTO MODULO INTERACTIVO (Programa NuevPend.Cpp)

Objetivo: Aplicar los conocimientos antes adquiridos es decir, las coordenadas de un punto, y la distancia existente entre dos distintos puntos; ahora ampliar este conocimiento a conocer con los mismos datos que ya tenemos de 2 puntos distintos, cual es la pendiente que forma la línea recta (o segmento de línea) que une ambos puntos, checar como varían los datos al mover una de los puntos entes dados.

Estrategia: Utilizando los datos y conocimientos obtenidos con los módulos anteriores, aplicar una sencilla fórmula matemática para descubrir el ángulo de inclinación o pendiente de una recta que pasa por dos puntos.
Imprimir en pantalla todo cambio de coordenadas de un punto si es que este cambia de posición
Presentación del texto relacionado al tema mediante un Hipertexto asociado a la aplicación final.

Parecerá que este módulo está de más, ya que en el módulo anterior se contempló el estudio de la pendiente, pero particularmente, éste implementa acciones sobre la línea recta como son la rotación o traslación de un segmento fijo a otra posición cualesquiera en la ventana visual.

Para la elaboración del cuarto módulo se diseña una máscara que presenta:

1.- Una ventana de diálogo que tenga las instrucciones de manejo del programa.
2.- Una ventana de diálogo que contenga información del modulo.
3.- Una ventana de diálogo que contenga aviso de Pendiente Indeterminada.
4.- Una ventana de ayuda de las acciones permitidas.
5.- Un menú de opciones que contenga:

a) Un Campo &Opciones.

&Salir ⇒ Para finalizar el programa.

&Limpiar ⇒ Para limpiar la ventana de trabajo.

b) Un Campo &Acción

&Rotar ⇒ Eventos que se realizan para rotar un punto.

&Trasladar ⇒ Eventos que se realizan para trasladar un punto.

c) Un campo A&yuda

&Uso ⇒ Instrucciones Iniciales de uso.

&Acerca de… ⇒ Información de la elaboración del módulo.

d) Un Campo Base Teórica. Hipertexto de lo esencial del tema.

Diseño:
La construcción del módulo se realizó mediante los siguientes pasos:

1.- Seleccionamos del menú principal File ⇒ New Application, para tener una forma sin inicializar.

2.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	" Ejes Coordenados Cartesianos ".
b) La propiedad ClientHeight	⇒	721
c) La propiedad ClientWidth	⇒	1024
d) La propiedad Color	⇒	" clBack "
e) La propiedad Icon	⇒	BildCim.ico
f) La propiedad Left	⇒	0
g) La propiedad Top	⇒	0

3.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

4.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	211
c) La propiedad Width	⇒	425
d) La propiedad Color	⇒	" clGray "
e) La propiedad Left	⇒	248
f) La propiedad Top	⇒	158

5.- De la paleta de componentes Standard seleccione un objeto Button 

y colóquelo en el Panel.

6.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Continuar.
b) La propiedad Height	⇒	105
c) La propiedad Width	⇒	75
d) La propiedad Left	⇒	336
e) La propiedad Top	⇒	56

Haga doble clic en el botón Continuar e introduzca el siguiente código.

Panel1->Visible=false;

7.- De la paleta de componentes Standard seleccione un objeto Memo  y colóquelo en elPanel.

8.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	177
b) La propiedad Width	⇒	313
c) La propiedad Left	⇒	16
d) La propiedad Top	⇒	16
e) La propiedad Lines	⇒	Doble clic

Aparecerá el editor de Líneas, en el se tecleará el siguiente Texto.

Pendiente.

Con el botón izquierdo del mouse, puedes
seleccionar arbitrariamente dos puntos
de los cuales se calculará Pendiente o ángulo
de inclinación de la recta según la formula:

$$m = \frac{Y2 - Y1}{X2 - X1}$$

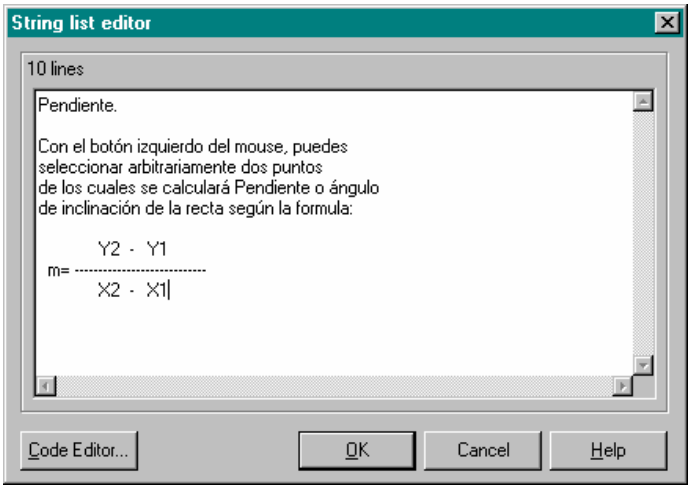


FIG. 4.4.1 Editor de líneas para componentes Memo.

9.- Selecciona el botón Ok para cerrar la ventana de dialogo.

10.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

11.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	65
c) La propiedad Align	⇒	"alBottom "
d) La propiedad Color	⇒	" clGray "
e) La propiedad Left	⇒	0
f) La propiedad Top	⇒	0

12.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

13.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	105
c) La propiedad Width	⇒	497
d) La propiedad Color	⇒	" clBtnFace "
e) La propiedad Left	⇒	8
f) La propiedad Top	⇒	616

14.- De la paleta de componentes Standard seleccione un objeto Button  y colóquelo en el Panel3.

15.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Aceptar.
b) La propiedad Height	⇒	73
c) La propiedad Width	⇒	65

d) La propiedad Left ⇒ 424
e) La propiedad Top ⇒ 16

Haga doble clic en el botón Aceptar e introduzca el siguiente código.

Panel3->Visible=false;

16.- De la paleta de componentes Standard seleccione un objeto Memo  y colóquelo en elPanel3.

17.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	89
b) La propiedad Width	⇒	409
c) La propiedad Left	⇒	8
d) La propiedad Top	⇒	8
e) La propiedad Lines	⇒	Doble clic

Aparecerá el editor de Líneas, en el se tecleará el siguiente Texto.

Pendiente Indeterminada. Cuendo la línea recta hace una paralela con la vertical, los valores de las x para ambos puntos son iguales, y al hacer la resta queda cero;
por lo tanto la fórmula para la pendiente quedaría de la siguiente manera:

$$m = \frac{Y2 - Y1}{0}$$ Por lo que el valor de la pendiente es: INDETERMINADO.

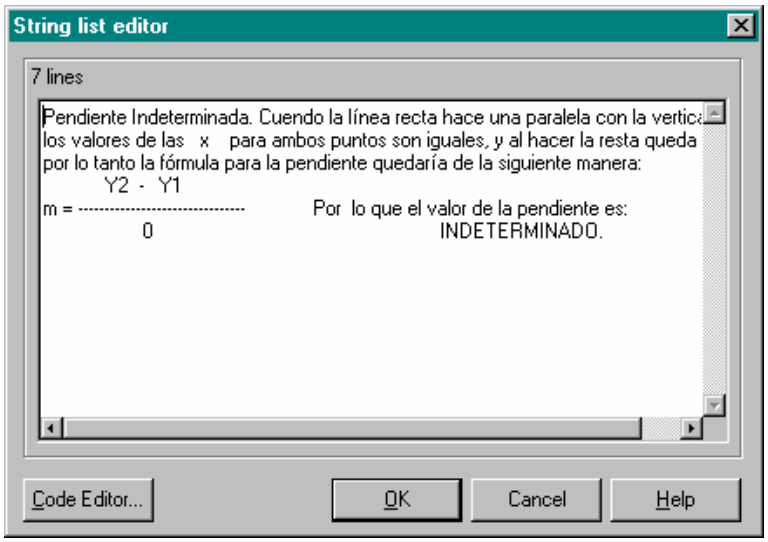


FIG. 4.4.2 Editor de líneas para componentes Memo.

18.- Selecciona el botón Ok para cerrar la ventana de dialogo.

19.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

20.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	177
c) La propiedad Width	⇒	145
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	736
f) La propiedad Top	⇒	144

21.- De la paleta de componentes Additional seleccione un objeto Image  y colóquelo en el Panel4.

22.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	73
b) La propiedad Width	⇒	57
c) La propiedad Left	⇒	48

f) La propiedad Top ⇒ 8
e) La propiedad Stretch ⇒ True
f) La propiedad Picture ⇒ Doble clic

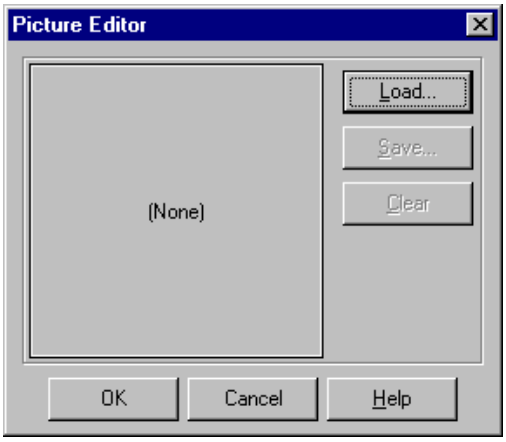


FIG. 4.4.3 Presentador de Imágenes.

Presione el botón Load y obtendrá una caja de dialogo para seleccionar el icono deseado o podrá buscarlo en caso de no encontrarse en el directorio mostrado.

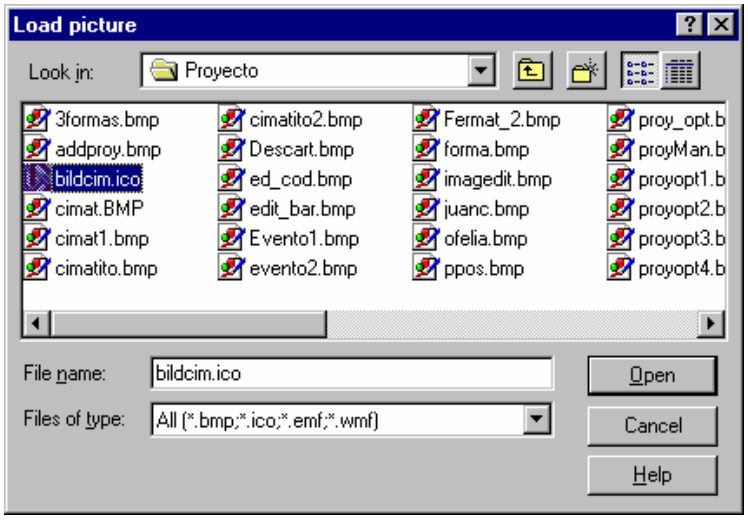


FIG. 4.4.4 Selección de Imágenes.

Seleccione la imagen deseada y oprima el botón Open.

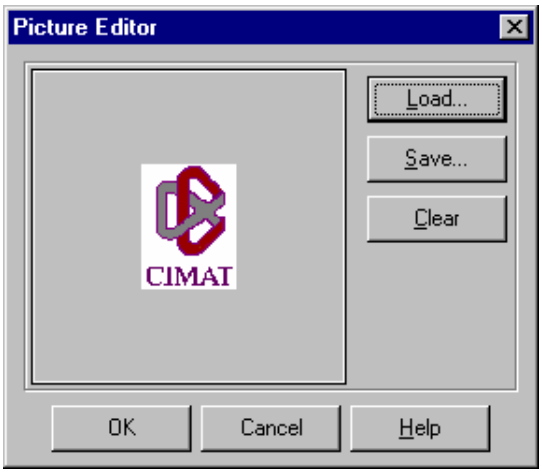


FIG. 4.4.5 Selección de Imágenes.

Oprima el botón Ok.

23.- De la paleta de componentes Standard seleccione un objeto Button 

y colóquelo en el Panel4.

24.- Modifiquemos los siguientes atributos:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Continuar
b) El evento OnClick	⇒	Panel4->Visible=false;

25.- De la paleta de componentes Standard seleccione un Label 

y colóquelo en el Panel4 con otras dos copias.

26.- Modifiquemos los siguientes atributos:

Label1.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Geometría Parte 1.
b) La propiedad Height	⇒	16
c) La propiedad Width	⇒	110
d) La propiedad Color	⇒	“ clTeal “
e) La propiedad Left	⇒	16
f) La propiedad Top	⇒	80
g) La propiedad Font Style	⇒	Bold Italic

h) La propiedad Font Size ⇒ 10

Label2.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	FEBRERO DE 1998
b) La propiedad Height	⇒	16
c) La propiedad Width	⇒	80
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	24
f) La propiedad Top	⇒	96
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	10

Label3.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	JUAN CARLOS CRUZ VEGA
b) La propiedad Height	⇒	11
c) La propiedad Width	⇒	128
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	8
f) La propiedad Top	⇒	128
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	7

27.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

28.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	73
c) La propiedad Left	⇒	8
d) La propiedad Top	⇒	0
f) La propiedad Color	⇒	" clTeal "

29.- De la paleta de componentes Standard seleccione un objeto Button  y colóquelo en el Panel5.

30.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Aceptar.
b) La propiedad Height	⇒	89
c) La propiedad Width	⇒	977
d) La propiedad Left	⇒	8
e) La propiedad Top	⇒	0

Haga doble clic en el botón Aceptar e introduzca el siguiente código.

```
Panel5->Visible=false;
```

31.- De la paleta de componentes Standard seleccione un objeto Memo  y colóquelo en elPanel3.

32.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	73
b) La propiedad Width	⇒	873
c) La propiedad Left	⇒	8
d) La propiedad Top	⇒	8
e) La propiedad Lines	⇒	Doble clic

Aparecerá el editor de Líneas, en el se tecleará el siguiente Texto.

ROTAR O TRASLADAR. Para rotar o trasladar la recta en tu ventana visual, primeramente tienes que seleccionar del menu Acción lo que quieras hacer, enseguida debes tomar uno de los extremos del segmento de recta dando un click con el botón izquierdo del mouse, una vez que
lo sujetes podrás trasladar o mover la recta a la posición que tu desees.
Para soltar la recta, tendrás que hacer doble click con el boton izquierdo del mouse.

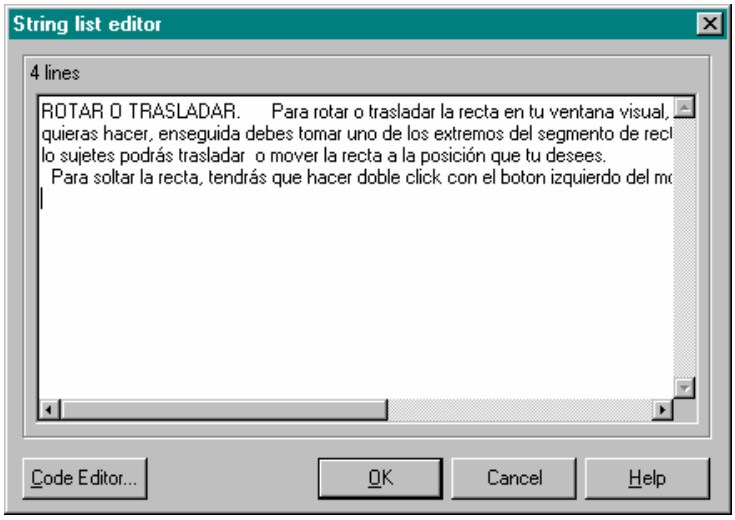


FIG. 4.4.6 Editor de líneas para componentes Memo.

33.- Selecciona el botón Ok para cerrar la ventana de dialogo.

34.- De la paleta de componentes Standard seleccione un objeto Main Menu  y colóquelo en la forma.

35.- Haga doble clic en este componente y obtendrá el diseñador de menus:

36.- Con el inspector de objetos cambie las propiedades:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Opciones + Enter
b) La propiedad Caption	⇒	&Ayuda + Enter
c) La propiedad Caption	⇒	A&Cerca de… + Enter
d) La propiedad Caption	⇒	&Base teórica + Enter

37.- Seleccione la hoja de eventos y haga doble clic en el evento OnClick e introduzca el siguiente código.

Componente	**Código**
	Application->HelpCommand(HELP_CONTENTS,0);
Opción **&Base Teórica**;	Invalidate();
Evento ***OnClick***	

38.- En el sub menú Opciones anexar 2 campos mas.

Propiedad	Valor o Texto

a) La propiedad Caption ⇒ &Limpiar
a) La propiedad Caption ⇒ &Salir

39.- En el sub menú Acción anexar 2 campos mas.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Rotar
a) La propiedad Caption	⇒	&Trasladar

40.- En el sub menú Ayuda colocar el siguiente código en el evento **OnClick**:

Propiedad		Valor o Texto
a) Evento ***OnClick***	⇒	Panel1->Visible=true;

41.- En el Sub menú aCerca de… colocar el siguiente código en el evento **OnClick**:

Propiedad		Valor o Texto
a) Evento ***OnClick***	⇒	Panel2->Visible=true;

42.- Seleccione la forma haciendo un clic en ella, con el inspector de objetos en la hoja de eventos haga doble clic en el evento que se marca e introduzca el código asociado.

Componente	**Código**	**Acción**

Form1
 	
Evento
OnMouseDown
	if(Tag<1)Obten_Puntos(X,Y);
	if(t==true) Punto_A_mover(X,Y,1);
	if(e==true) Punto_A_mover(X,Y,2);
		Guarda la posición de los puntos que se fijaron en el eje coordenado, la variables t y e guardan la posición del primer y segundo punto por si ese se selecciona para trasladar o rotar

Form1
 	
Evento
OnMouseMove
	float t1,t2;
	t1=(float(X-(ClientWidth/2))/30.0);
	t2=(float((ClientHeight/2)-Y)/30.0);
	sprintf(text,"
	",X1,Y1);
	Canvas->TextOut(120,660,text);
	Canvas->TextOut(120,665,text);
	Canvas->TextOut(120,670,text);
	Canvas->Font->Size=12;
	Canvas->Font->Color=clAqua;
	Sprintf(text,"X= %2.1f Y= %2.1f
	",t1,t2);
	Canvas->TextOut(120,665,text);
	Canvas->Font->Color=clWhite;
	if(Tag>=1){ Xf=X; Yf=Y;
	If(w==true)
	{ Mueve(Xf,Yf); }
	if(ee==true)
	{ Traslada(Xf,Yf); }
	}
	}
		Calcúla e imprime en la pantalla la posición en el eje coordenado por el cual se está desplazando el mouse.
		
		Guarda las posiciones de los puntos puestos en el sistema coordenado para las acciones de rotación y traslación.

Evento **OnDblClick**	z=true; e=t=n=s=nn=ss=false;	
Form1 Evento **OnPaint**	Ejes();	Se redibujan los ejes y sus graduaciones.
Form1 Evento **OnReSize**	Invalidate();	Borra lo que se tenía para el ajuste de tamaño.

43.- Así se verá la construcción:

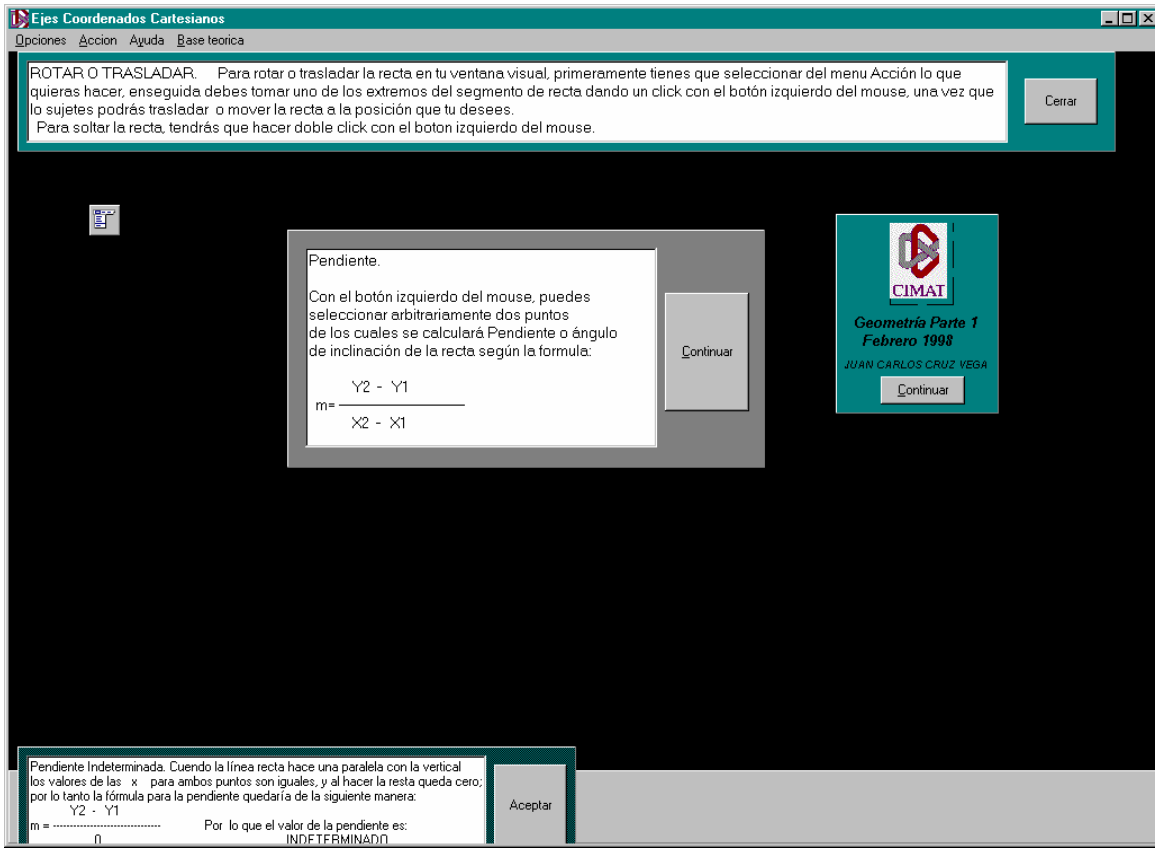


FIG. 4.4.7 Construcción Módulo No. 4.

44.- Seleccione el editor de código y al final de la funciones ya capturadas introduzca las funciones con las que complementará el modulo de programa, según el Apéndice b.

45.- Seleccione File ⇒ Save Project As… Para almacenar al archivo en disco.

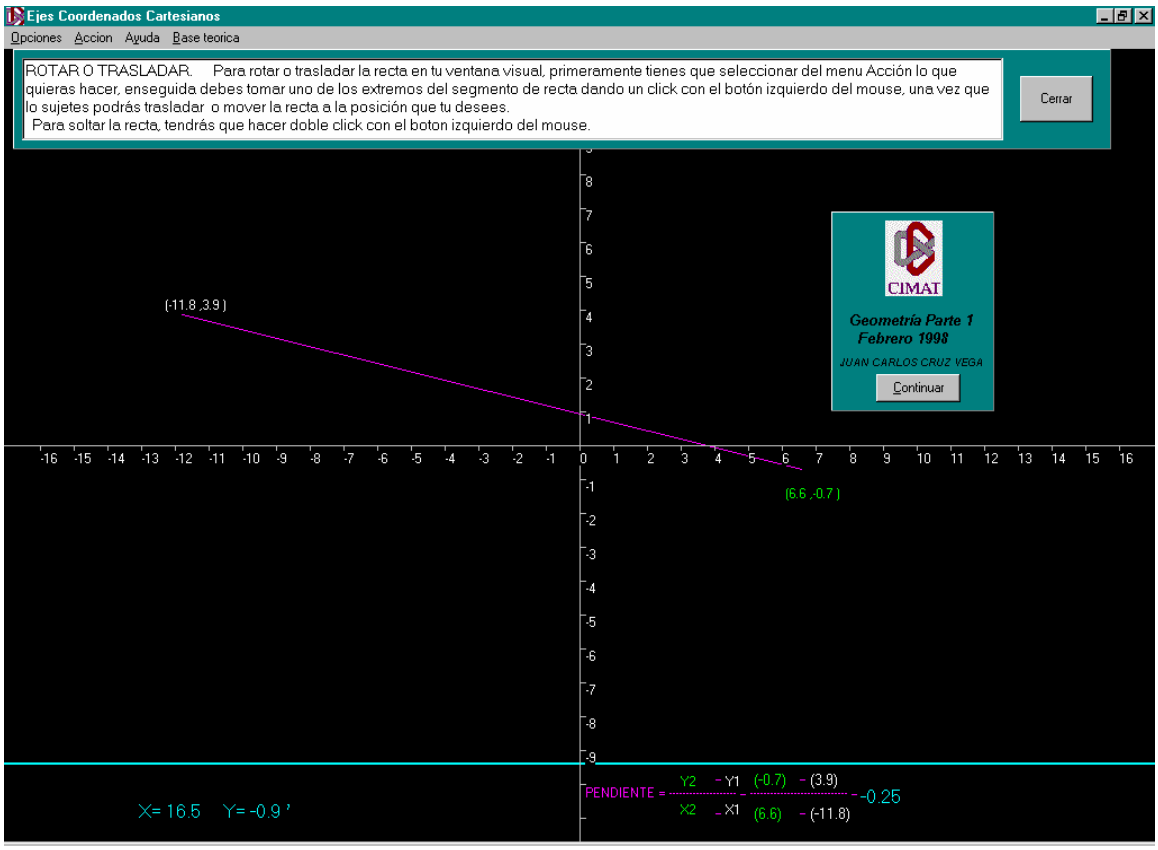


FIG. 4.4.8 Ejecución Módulo No. 4.

4.5 QUINTO MODULO INTERACTIVO (Programa Mxplusb.Cpp)

Objetivo: Presentación Gráfica de la recta y sus diferentes ecuaciones.

Estrategia: Elaboración de un módulo en el cual el usuario mediante el mouse coloque 2 puntos aleatorios en el sistema coordenado, y el programa muestre según los puntos marcados, el trazo de la recta que pasa por esos puntos, además de visualizar sus diferentes representaciones o ecuaciones.

Presentación del texto relacionado al tema mediante un Hipertexto asociado a la aplicación final.

Para la elaboración del Quinto módulo se diseña la máscara siguiente:

1.- Una ventana de diálogo que tenga las instrucciones de manejo del programa.
2.- Una ventana de diálogo que contenga información del módulo.
3.- Un menú de opciones que contenga:

a) Un Campo Opciones.

&Salir ⇒ Para finalizar el programa.
&Limpiar ⇒ Para limpiar la ventana de trabajo.

b) Un Campo Ayuda
&Uso ⇒ Información del manejo del módulo.
&Indicaciones ⇒ Información de manipulación del objeto.

c) Un Campo A&cerca de… Información de la construcción del módulo.
d) Un Campo Base Teórica. Hipertexto de lo esencial del tema.

Diseño:
La construcción del módulo se realizó mediante los siguientes pasos:

1.- Seleccionamos del menú principal File ⇒ New Application, para tener una forma sin inicializar.

2.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	" LA RECTA CIMAT 1998 ".
b) La propiedad ClientHeight	⇒	721
c) La propiedad ClientWidth	⇒	1024
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Icon	⇒	BildCim.ico
f) La propiedad Left	⇒	0
g) La propiedad Top	⇒	0

3.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

4.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	145
c) La propiedad Width	⇒	609
d) La propiedad Color	⇒	" clsilver "
e) La propiedad Left	⇒	0
f) La propiedad Top	⇒	548

5.- De la paleta de componentes Standard seleccione un objeto Button  y colóquelo en el Panel.

6.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Aceptar.
b) La propiedad Height	⇒	97
c) La propiedad Width	⇒	49
d) La propiedad Left	⇒	552
e) La propiedad Top	⇒	32

Haga doble clic en el botón Continuar e introduzca el siguiente código.

```
Panel1->Visible=false;
```

7.- De la paleta de componentes Standard seleccione un objeto Memo  y colóquelo en el Panel.

8.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	129

b) La propiedad Width ⇒ 539
c) La propiedad Left ⇒ 6
d) La propiedad Top ⇒ 8
e) La propiedad Lines ⇒ Doble clic

Aparecerá el editor de Líneas, en el se tecleará el siguiente Texto.

Pendiente Indeterminada. Cuando la línea recta hace una paralela con la vertical los valores de las x para ambos puntos son iguales, y al hacer la resta queda cero; por lo tanto la fórmula para la pendiente quedaría de la siguiente manera:

$$m = \frac{Y2 - Y1}{0}$$ Por lo que el valor de la pendiente es: INDETERMINADO.

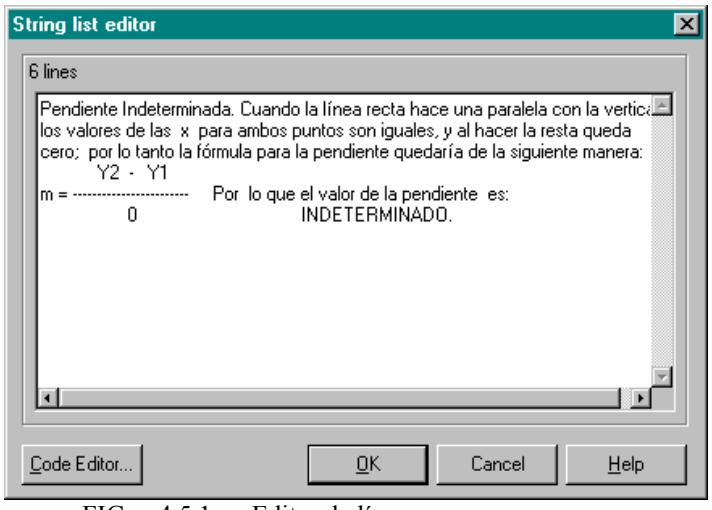


FIG. 4.5.1 Editor de líneas para campo memo.

9.- Selecciona el botón Ok para cerrar la ventana de dialogo.

10.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

11.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	697
c) La propiedad Width	⇒	415
d) La propiedad Color	⇒	“ clTeal “
e) La propiedad Left	⇒	601
f) La propiedad Top	⇒	0

12.- De la paleta de componentes Standard seleccione un objeto Label  y colóquelo en el Panel2, realice esta operación 15 veces mas de manera que tenga 16 componentes Label en la forma2.

13.- Modificamos sus propiedades de la siguiente manera:

Label1

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Pendiente
b) La propiedad Height	⇒	20
c) La propiedad Width	⇒	91
d) La propiedad Color	⇒	" clGray "
e) La propiedad Left	⇒	24
f) La propiedad Top	⇒	248

Label2

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	y – y1 = m(x – x1)
b) La propiedad Height	⇒	29
c) La propiedad Width	⇒	138
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	24
f) La propiedad Top	⇒	264

Label3

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	y = mx – mx1 + y1
b) La propiedad Height	⇒	29
c) La propiedad Width	⇒	140
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	24
f) La propiedad Top	⇒	288

Label4

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Y= mx – mxi + y1
b) La propiedad Height	⇒	25

c) La propiedad Width ⇒ 158
d) La propiedad Color ⇒ " clTeal "
e) La propiedad Left ⇒ 24
f) La propiedad Top ⇒ 328

Label5

Propiedad Valor o Texto

a) La propiedad Caption ⇒ Forma Punto Pendiente
b) La propiedad Height ⇒ 20
c) La propiedad Width ⇒ 199
d) La propiedad Color ⇒ " clGray "
e) La propiedad Left ⇒ 96
f) La propiedad Top ⇒ 368

Label6

Propiedad Valor o Texto

a) La propiedad Caption ⇒ y – y1 = m (x - x1)
b) La propiedad Height ⇒ 29
c) La propiedad Width ⇒ 292
d) La propiedad Color ⇒ " clTeal "
e) La propiedad Left ⇒ 40
f) La propiedad Top ⇒ 400

Label7

Propiedad Valor o Texto

a) La propiedad Caption ⇒ Pendiente - Ordenada al origen
b) La propiedad Height ⇒ 20
c) La propiedad Width ⇒ 263
d) La propiedad Color ⇒ " clGray "
e) La propiedad Left ⇒ 64
f) La propiedad Top ⇒ 440

Label8

Propiedad Valor o Texto

a) La propiedad Caption ⇒ y = mx + b
b) La propiedad Height ⇒ 29
c) La propiedad Width ⇒ 104
d) La propiedad Color ⇒ " clTeal "
e) La propiedad Left ⇒ 128

f) La propiedad Top ⇒ 464

Label9

Propiedad	Valor o Texto
a) La propiedad Caption	⇒ y – y1 = m (x - x1)
b) La propiedad Height	⇒ 29
c) La propiedad Width	⇒ 292
d) La propiedad Color	⇒ “ clTeal “
e) La propiedad Left	⇒ 40
f) La propiedad Top	⇒ 400

Label10

Propiedad	Valor o Texto
a) La propiedad Caption	⇒ y =
b) La propiedad Height	⇒ 29
c) La propiedad Width	⇒ 25
d) La propiedad Color	⇒ “ clTeal “
e) La propiedad Left	⇒ 40
f) La propiedad Top	⇒ 512

Label11

Propiedad	Valor o Texto
a) La propiedad Caption	⇒ x +
b) La propiedad Height	⇒ 29
c) La propiedad Width	⇒ 50
d) La propiedad Color	⇒ “ clTeal “
e) La propiedad Left	⇒ 152
f) La propiedad Top	⇒ 512

Label12

Propiedad	Valor o Texto
a) La propiedad Caption	⇒ Ecuación General de la Recta
b) La propiedad Height	⇒ 20
c) La propiedad Width	⇒ 250
d) La propiedad Color	⇒ “ clGray “
e) La propiedad Left	⇒ 64
f) La propiedad Top	⇒ 560

Label13

Propiedad	Valor o Texto
a) La propiedad Caption	⇒ Ax + By + C = 0

b) La propiedad Height ⇒ 29
c) La propiedad Width ⇒ 166
d) La propiedad Color ⇒ " clTeal "
e) La propiedad Left ⇒ 96
f) La propiedad Top ⇒ 584

Label14

Propiedad Valor o Texto

a) La propiedad Caption ⇒ x +
b) La propiedad Height ⇒ 27
c) La propiedad Width ⇒ 39
d) La propiedad Color ⇒ " clTeal "
e) La propiedad Left ⇒ 80
f) La propiedad Top ⇒ 624

Label15

Propiedad Valor o Texto

a) La propiedad Caption ⇒ y +
b) La propiedad Height ⇒ 27
c) La propiedad Width ⇒ 31
d) La propiedad Color ⇒ " clTeal "
e) La propiedad Left ⇒ 200
f) La propiedad Top ⇒ 624

Label16

Propiedad Valor o Texto

a) La propiedad Caption ⇒ = 0
b) La propiedad Height ⇒ 29
c) La propiedad Width ⇒ 34
d) La propiedad Color ⇒ " clTeal "
e) La propiedad Left ⇒ 328
f) La propiedad Top ⇒ 624

14.- De la paleta de componentes Standard seleccione un objeto TEditl  y colóquelo en el Panel2, realice esta operación 4 veces más de manera que tenga 5 componentes TEdit en la forma2.

15.- Modificamos sus propiedades de la siguiente manera:

Tedit1

Propiedad		**Valor o Texto**
a) La propiedad Text	⇒	0
b) La propiedad Height	⇒	32
c) La propiedad Width	⇒	57
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	72
f) La propiedad Top	⇒	512

Eventos

OnChange

Código

```
float W;
 strcpy(text,MODIm->Text.c_str());
 sscanf(text,"%f",&INTER1);
 IncM->Position=INTER1*10;
 W=INTER1;
 sprintf(text,"%.1f",W);
 Label16->Caption=text;
 R=INTER1;
 M=R;
 if(INTER1<=300) Traza_Recta(R);
 else Limpia();
```

Tedit2

Propiedad		**Valor o Texto**
a) La propiedad Text	⇒	0
b) La propiedad Height	⇒	32
c) La propiedad Width	⇒	57
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	200
f) La propiedad Top	⇒	512

Eventos

OnChange

```
float W;
 strcpy(text,MODIb->Text.c_str());
 sscanf(text,"%f",&INTER);
 Incb->Position=INTER*10.0;
 W=INTER;
 Recta2(W);
```

Tedit3

Propiedad		Valor o Texto
a) La propiedad Text	⇒	0

b) La propiedad Height ⇒ 28
c) La propiedad Width ⇒ 57
d) La propiedad Color ⇒ “ clTeal “
e) La propiedad Left ⇒ 8
f) La propiedad Top ⇒ 624

Eventos

OnChange

```
float W;
strcpy(text,MODIa->Text.c_str());
sscanf(text,"%f",&INTER1);
W=INTER1;
sprintf(text,"%.1f",W);
Label16->Caption=text;
MODIm->Text=text;
COEX->Position=INTER1*10;
IncM->Position=INTER1*10;
R=INTER1;
M=R;
if(INTER1<=300) Traza_Recta(R);
else Limpia();
```

Tedit4

Propiedad Valor o Texto

a) La propiedad Text ⇒ 0
b) La propiedad Height ⇒ 28
c) La propiedad Width ⇒ 57
d) La propiedad Color ⇒ “ clTeal “
e) La propiedad Left ⇒ 128
f) La propiedad Top ⇒ 624

Eventos

OnChange

Tedit5

Propiedad Valor o Texto

a) La propiedad Text ⇒ 0
b) La propiedad Height ⇒ 28
c) La propiedad Width ⇒ 57
d) La propiedad Color ⇒ “ clTeal “
e) La propiedad Left ⇒ 248
f) La propiedad Top ⇒ 624

Eventos	float W;
	strcpy(text,MODIC->Text.c_str());
	sscanf(text,"%f",&INTER);
	CTE->Position=INTER*10.0;
	Incb->Position=INTER*10.0;
	sprintf(text,"%.1f",INTER);
	MODIb->Text=text;
	W=INTER;
	Recta2(W);

OnChange	

16.- De la paleta de componentes Win95 seleccione un objeto UpDown  y colóquelo en el Panel2, realice esta operación 4 veces más de manera que tenga 5 componentes UpDown en la forma2.

17.- Modificamos sus propiedades de la siguiente manera:

UpDown1

Propiedad		**Valor o Texto**
a) La propiedad Increment	⇒	1
b) La propiedad Height	⇒	33
c) La propiedad Width	⇒	17
d) La propiedad Max	⇒	500
e) La propiedad Min	⇒	-500
f) La propiedad Left	⇒	128
g) La propiedad Top	⇒	512
h) La propiedad Name	⇒	“ IncM “

Eventos	**Código**

OnClick	MODIm->Text=IncM->Position*0.1;
	strcpy(text,MODIm->Text.c_str());
	sscanf(text,"%f",&INTER1); // M ESTA EN reales
	sprintf(text,"%.1f",INTER1);
	Label16->Caption=text;
	R=INTER1;
	M=R;
	Traza_Recta(R);

UpDown2

Propiedad		**Valor o Texto**
a) La propiedad Increment	⇒	1
b) La propiedad Height	⇒	33
c) La propiedad Width	⇒	17
d) La propiedad Max	⇒	500
e) La propiedad Min	⇒	-500
f) La propiedad Left	⇒	256
g) La propiedad Top	⇒	512
h) La propiedad Name	⇒	“ Incb “

Código

Eventos

OnClick

```
MODIb->Text=Incb->Position*0.1;
strcpy(text,MODIb->Text.c_str());
sscanf(text,"%f",&INTER);              // M ESTA EN reales
b=INTER;
Recta2(b);
```

UpDown3

Propiedad		**Valor o Texto**
a) La propiedad Increment	⇒	1
b) La propiedad Height	⇒	33
c) La propiedad Width	⇒	17
d) La propiedad Max	⇒	500
e) La propiedad Min	⇒	-500
f) La propiedad Left	⇒	64
g) La propiedad Top	⇒	624
h) La propiedad Name	⇒	“ COEX “

Código

Eventos

OnClick

```
MODIa->Text=COEX->Position*0.1;
strcpy(text,MODIa->Text.c_str());
sscanf(text,"%f",&INTER1);             // M ESTA EN reales
sprintf(text,"%.1f",INTER1);
Label16->Caption=text;
R=INTER1;
M=R;
Traza_Recta(R);
```

UpDown4

Propiedad		Valor o Texto
a) La propiedad Increment	⇒	1
b) La propiedad Height	⇒	33
c) La propiedad Width	⇒	17
d) La propiedad Max	⇒	500
e) La propiedad Min	⇒	-500
f) La propiedad Left	⇒	304
g) La propiedad Top	⇒	624
h) La propiedad Name	⇒	“ CTE “

Eventos	Código

OnClick	float cte;
	MODIC->Text=CTE->Position*0.1;
	strcpy(text,MODIC->Text.c_str());
	sscanf(text,"%f",&cte);

18.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

19.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	169
c) La propiedad Width	⇒	153
d) La propiedad Color	⇒	“ clSilver “
e) La propiedad Left	⇒	838
f) La propiedad Top	⇒	200

20.- De la paleta de componentes Additional seleccione un objeto Image  y colóquelo en el Panel2.

21.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	73
b) La propiedad Width	⇒	57
c) La propiedad Left	⇒	48
f) La propiedad Top	⇒	8
e) La propiedad Stretch	⇒	True
f) La propiedad Picture	⇒	Doble clic

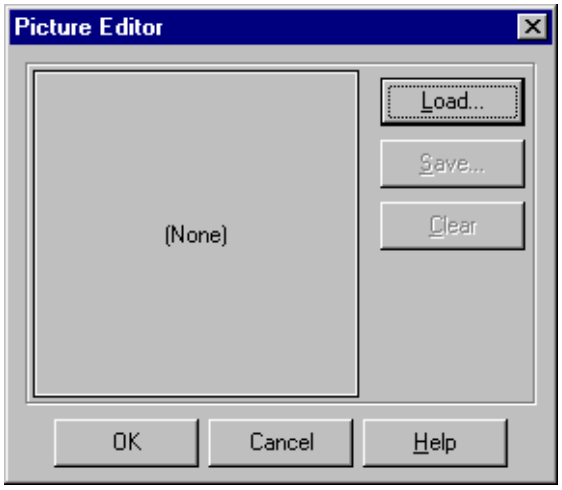


FIG. 4.5.2 Presentador de Imágenes.

Presione el botón Load y obtendrá una caja de dialogo para seleccionar el icono deseado o podrá buscarlo en caso de no encontrarse en el directorio mostrado.

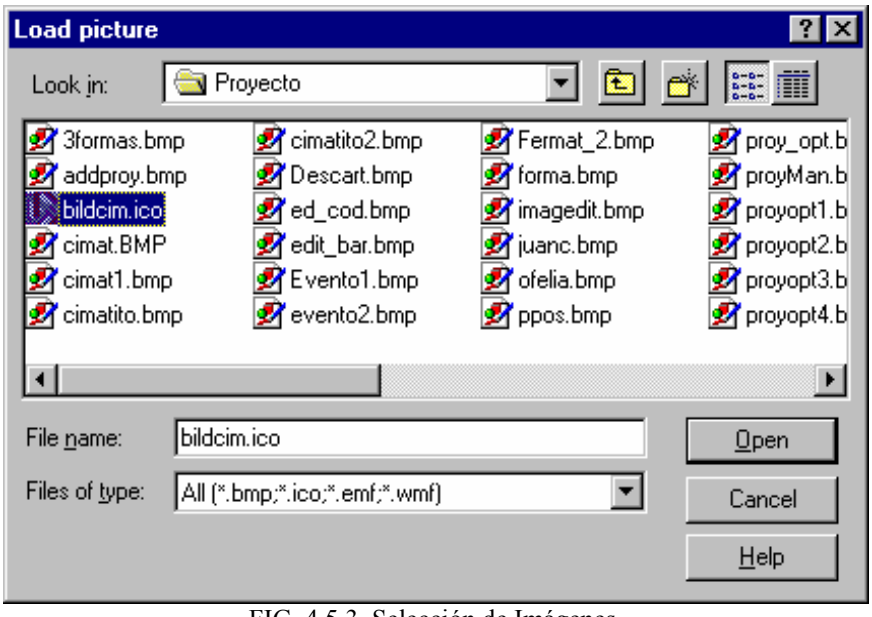


FIG. 4.5.3 Selección de Imágenes.

Seleccione la imagen deseada y oprima el botón Open.

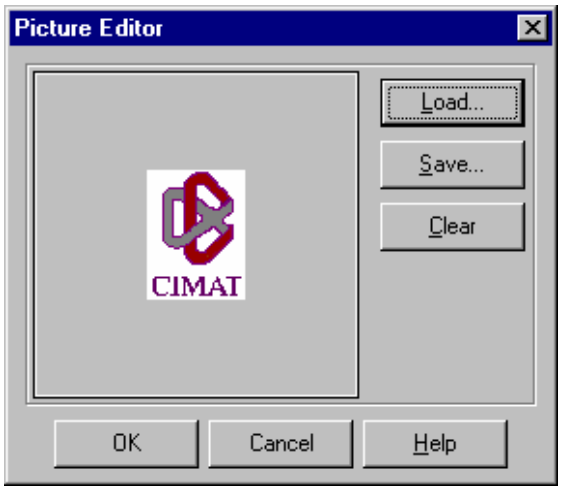


FIG. 4.5.4 Selección de Imágenes.

Oprima el botón Ok.

22.- De la paleta de componentes Standard seleccione un objeto Button  y colóquelo en el Panel2.

23.- Modifiquemos los siguientes atributos:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Continuar
b) El evento OnClick	⇒	Panel2->Visible=false;

24.- De la paleta de componentes Standard seleccione un Label  y colóquelo en el Panel2 con otras dos copias.

25.- Modifiquemos los siguientes atributos:

Label1.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Geometría Parte 1.
b) La propiedad Height	⇒	24
c) La propiedad Width	⇒	179
d) La propiedad Color	⇒	“ clGray “
e) La propiedad Left	⇒	16

f) La propiedad Top ⇒ 120
g) La propiedad Font Style ⇒ Bold Italic
h) La propiedad Font Size ⇒ 16

Label2.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	JUAN CARLOS CRUZ VEGA
b) La propiedad Height	⇒	13
c) La propiedad Width	⇒	167
d) La propiedad Color	⇒	" clGray "
e) La propiedad Left	⇒	16
f) La propiedad Top	⇒	144
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	9

Label3.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	FEBRERO DE 1998
b) La propiedad Height	⇒	13
c) La propiedad Width	⇒	118
d) La propiedad Color	⇒	" clGray "
e) La propiedad Left	⇒	32
f) La propiedad Top	⇒	160
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	9

26.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

27.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	233
c) La propiedad Width	⇒	404
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	0
f) La propiedad Top	⇒	8

28.- De la paleta de componentes Standard seleccione un objeto Memo  y colóquelo en el Panel.

29.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	209
b) La propiedad Width	⇒	396
c) La propiedad Left	⇒	5
d) La propiedad Top	⇒	8
e) La propiedad Lines	⇒	Doble clic

Aparecerá el editor de Líneas, en el se tecleará el siguiente Texto.

USO DE LA APLICACION.

Después de trazar la recta, y que conoscas los datos de la ecuación original, puedes apreciar que al modificar la pendiente (m), el efecto que toma la recta es el de un giro apartir de un punto conocido.

Si por el contrario modificas la b, observas que el efecto que toma la recta es de una traslación a través del parámetro que modificas que es el cruce con el eje y.

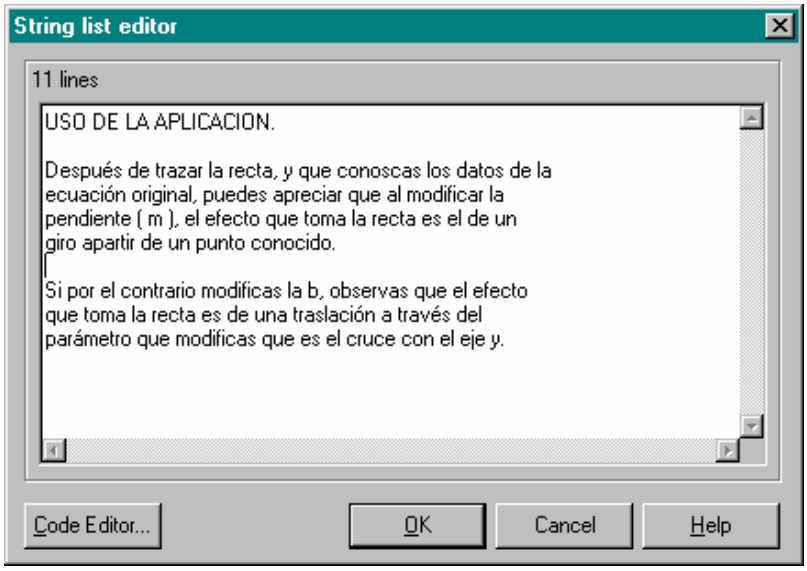


FIG. 4.5.5 Editor de líneas.

30.- Selecciona el botón Ok para cerrar la ventana de dialogo.

31.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

32.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	233
c) La propiedad Width	⇒	417
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Left	⇒	0
f) La propiedad Top	⇒	8

33.- De la paleta de componentes Standard seleccione un objeto Memo  y colóquelo en el Panel.

34.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	217
b) La propiedad Width	⇒	401
c) La propiedad Left	⇒	8
d) La propiedad Top	⇒	8
e) La propiedad Lines	⇒	Doble clic

Aparecerá el editor de Líneas, en el se tecleará el siguiente Texto.

INDICACIONES:
Para iniciar, debes con el botón izquierdo del mouse
seleccionar 2 puntos arbitrarios en tu Sistema
Coordenado; A partir de estos puntos se calcularán
Las siguientes ecuaciones de la recta.

a) Pendiente.
b) $y - y1 = m (x - x1)$
c) $y = mx + b$
d) $Ax + By + C = 0$

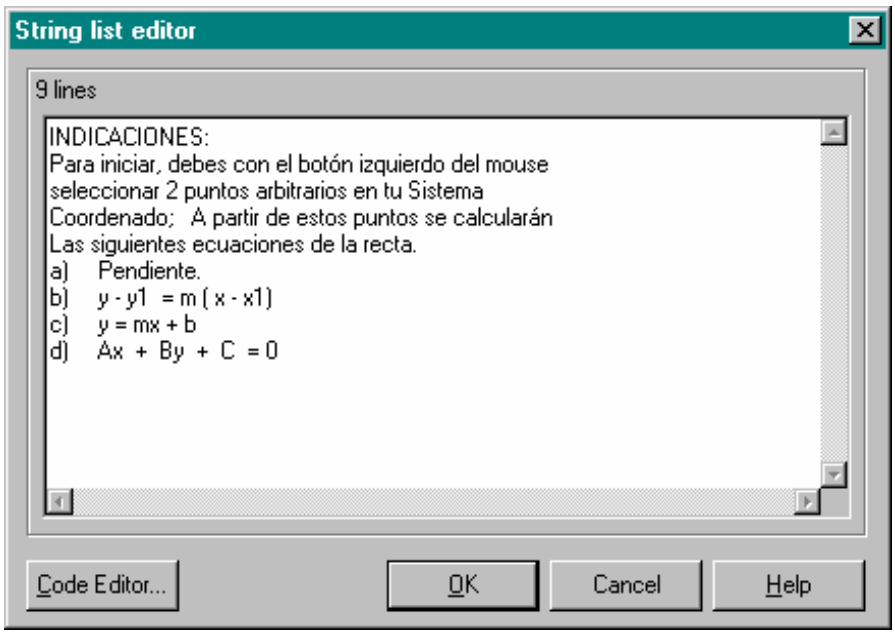


FIG. 4.5.6 Editor de Líneas.

35.- Selecciona el botón Ok para cerrar la ventana de dialogo.

36.- De la paleta de componentes Standard seleccione un objeto Main Menú  y colóquelo en la forma.

37.- haga doble clic en este componente y obtendrá el diseñador de menús:

38.- Con el inspector de objetos cambie las propiedades:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Opciones + Enter
b) La propiedad Caption	⇒	&Ayuda + Enter
c) La propiedad Caption	⇒	A&cerca de… + Enter
c) La propiedad Caption	⇒	&Base teórica + Enter

39.- Seleccione la hoja de eventos y haga doble clic en el evento OnClick e introduzca el siguiente código.

Componente	Código
Opción **&Base Teórica**;	Application->HelpCommand(HELP_CONTENTS,0);
Evento ***OnClick***	Invalidate();

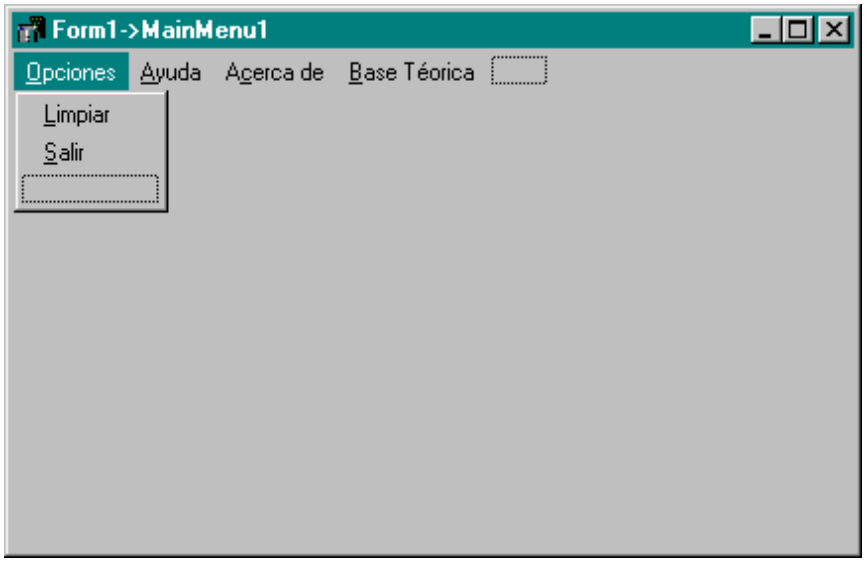


FIG. 4.5.7 Diseñador de menús.

40.- En el sub menú Opciones anexar 2 campos mas.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Limpiar
a) La propiedad Caption	⇒	&Salir

41.- Seleccione la hoja de eventos y haga doble clic en el evento OnClick e introduzca el siguiente código.

Componente	Código
Opción **&Limpiar**;	Invalidate();
Evento ***OnClick***	
Opción **&Salir**;	Close();
Evento ***OnClick***	Exit(1);

42.- En el sub menú Ayuda anexar 2 campos mas.

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Uso
a) La propiedad Caption	⇒	&Indicaciones

43.- Seleccione la hoja de eventos y haga doble clic en el evento OnClick e introduzca el siguiente código.

Componente	**Código**
Opción **&Uso**;	Panel1->Visible=true;
Evento ***OnClick***	Invalidate();
Opción **a&Cerca de…**	Panel2->Visible=true;
Evento ***OnClick***	

44.- Seleccione la forma haciendo un clic en ella, con el inspector de objetos en la hoja de eventos haga doble clic en el evento que se marca e introduzca el código asociado.

Componente	**Código**	**Acción**

Form1
 	
Evento
OnMouseDown
	Canvas->MoveTo(X,Y);
	Canvas->Pen->Color=clAqua;
	Canvas->Ellipse(X,Y,X,Y);
	Canvas->Ellipse(X,Y,X+1,Y+1);
	Canvas->Ellipse(X,Y,X+2,Y+2);
	Canvas->Ellipse(X,Y,X+3,Y+3);
	Canvas->Ellipse(X,Y,X+4,Y+4);
	old_x=X;
	old_y=Y;
	Pinta_Coordenada(old_x,old_y);
		Se selecciona un punto en la pantalla y se remarca de manera que no se vea un solo pixel, sino un grupo de pixeles.
		
		Se hace una llamada a la función que pintará los valores numéricos de la coordenada.

Form1 Evento ***OnMouseMove***	float X1,Y1; X1=(float(X-(ClientWidth/2))/30.0); Y1=(float((ClientHeight/2)-Y)/30.0); Sprintf(text," ",X1,Y1); Canvas->TextOut(5,5,text); Canvas->TextOut(0,0,text); Canvas->TextOut(5,10,text); Canvas->Font->Color=clAqua; Sprintf(text,"X= %2.1f Y= %2.1f ",X1,Y1); Canvas->TextOut(5,5,text);	Se obtienen los valores de la posición del mouse mientras este se esta moviendo. Se realizan los cálculos en base al valor en pixeles de la pantalla visual en la que se esta trabajando, se cambia de valor de pixeles a valor de unidades de medida.

	Canvas->Font->Color=clWhite;	Los valores obtenidos en las operaciones anteriores son desplegados en la pantalla visual.
Form1 Evento OnPaint	Ejes();	Se realiza una llamada a la función que marca las graduaciones de las coordenadas.
Form1 Evento OnReSize	Invalidate();	Borra lo que se tenia para el ajuste de tamaño.

Hasta ahora tenemos integrado nuestro módulo con la siguiente presentación:

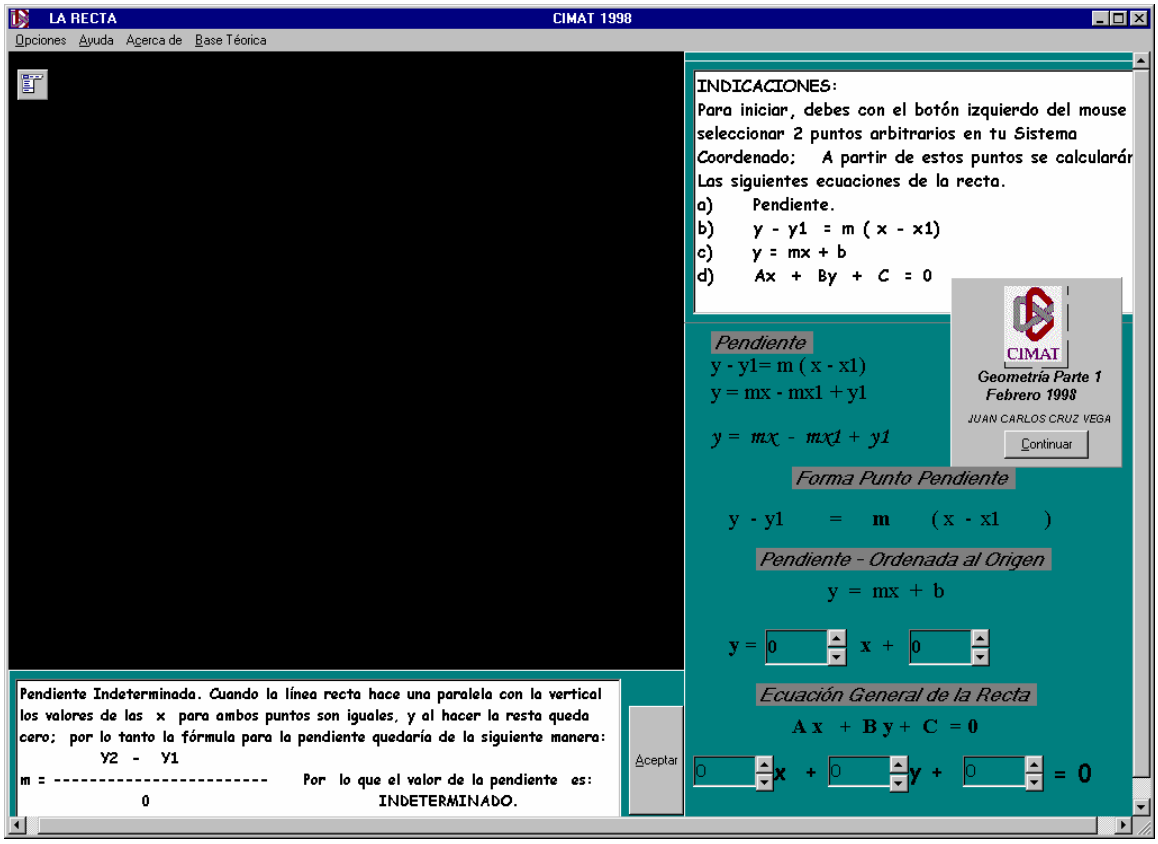


FIG. 4.5.8 Componentes del módulo 5.

45.- Seleccione el editor de código y al final de la funciones ya capturadas introduzca las funciones con las que complementará el modulo de programa, según el Apéndice b.

46.- Seleccione File ⇒ Save Project As… y obtendrá una ventana de dialogo en la que podrá darle nombre a la aplicación.

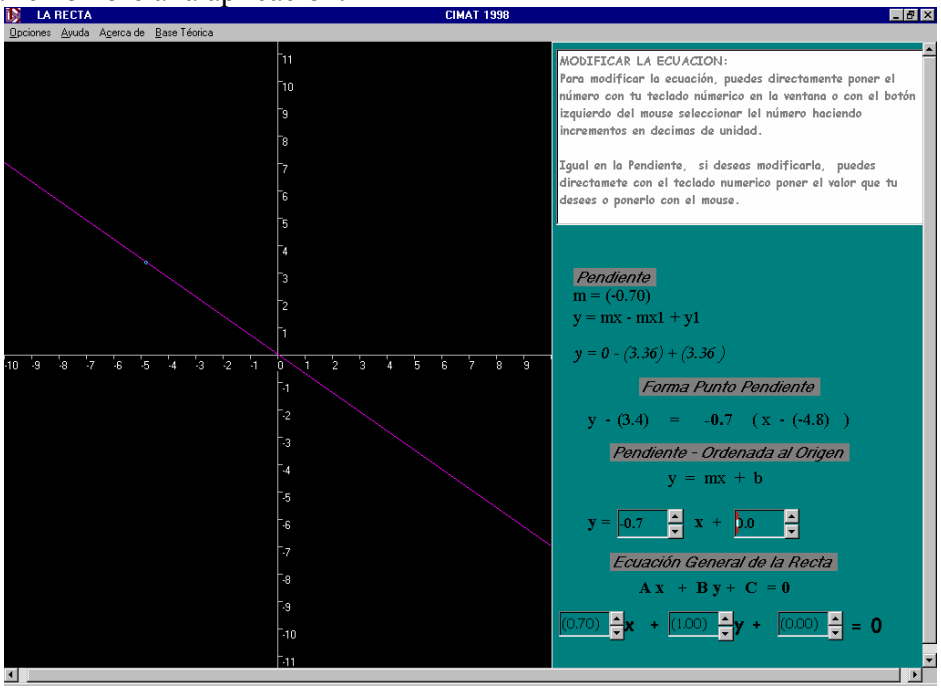


FIG. 4.5.9 ejecución del módulo 5.

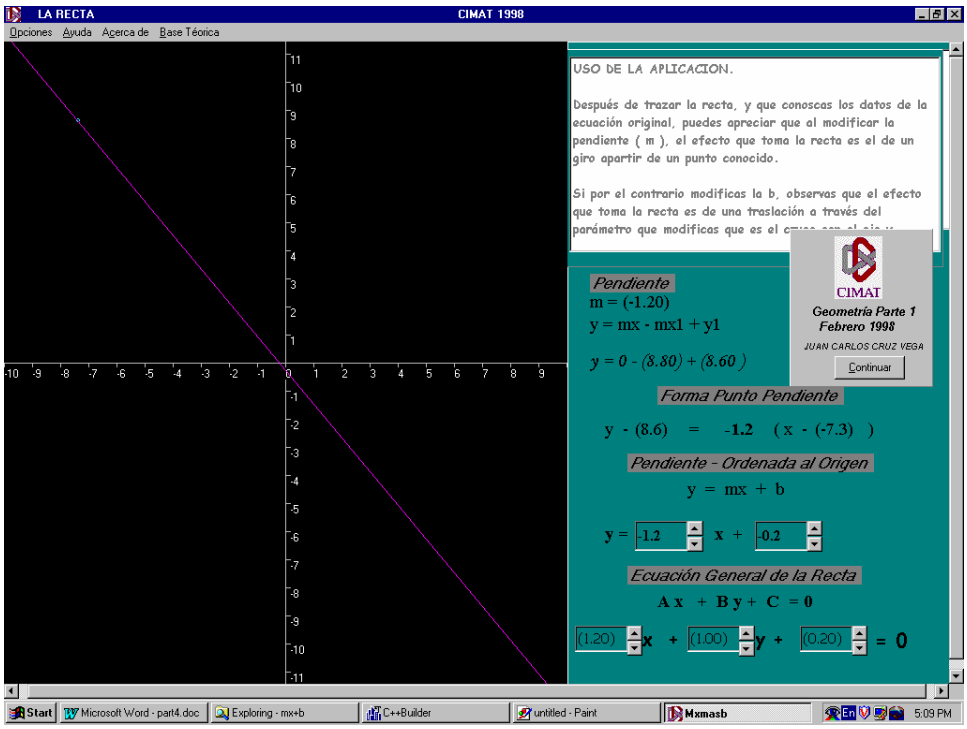


FIG. 4.5.10 ejecución del módulo 5.

4.6 PRESENTACION (Sim.cpp)

Para la elaboración de la presentación se diseña una máscara que presenta:

1.- El logotipo de la institución donde se desarrollo el proyecto.
2.- El logotipo de la institución donde se entregará el proyecto.
3.- Botones que controlen el acceso a los diferentes módulos construidos.
4.- Un menú de opciones que contenga:

c) Una opción para salir del proyecto.
d) Una ventana de dialogo con información de la elaboración del módulo.

Y pasamos a la construcción del módulo con los siguientes pasos:

1.- Seleccionamos del menú principal File ⇒ New Application, para tener una forma sin inicializar.

2.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	" Herramientas Gráficas Interactivas ".
b) La propiedad ClientHeight	⇒	479
c) La propiedad ClientWidth	⇒	570
d) La propiedad Color	⇒	" clTeal "
e) La propiedad Icon	⇒	BildCim.ico
f) La propiedad Left	⇒	213
g) La propiedad Top	⇒	120

Y lo que veremos será la ventana en la cual tendremos el control del programa:

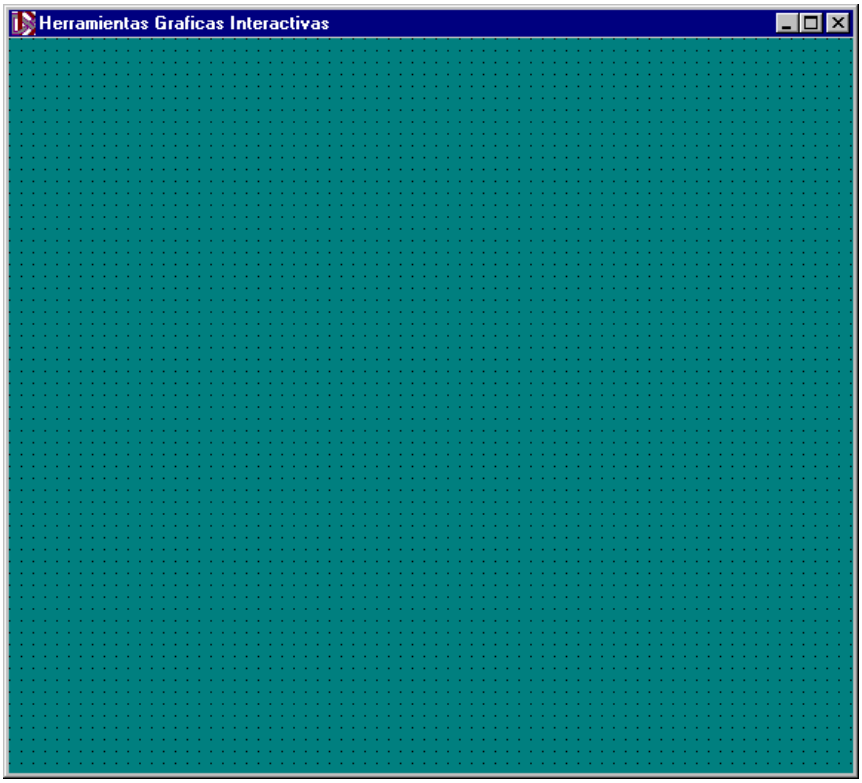


FIG. 4.6.1 Diseño de la forma.

3.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

4.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	129
c) La propiedad Width	⇒	544
d) La propiedad Color	⇒	“ clTeal “
e) La propiedad Align	⇒	AlBottom

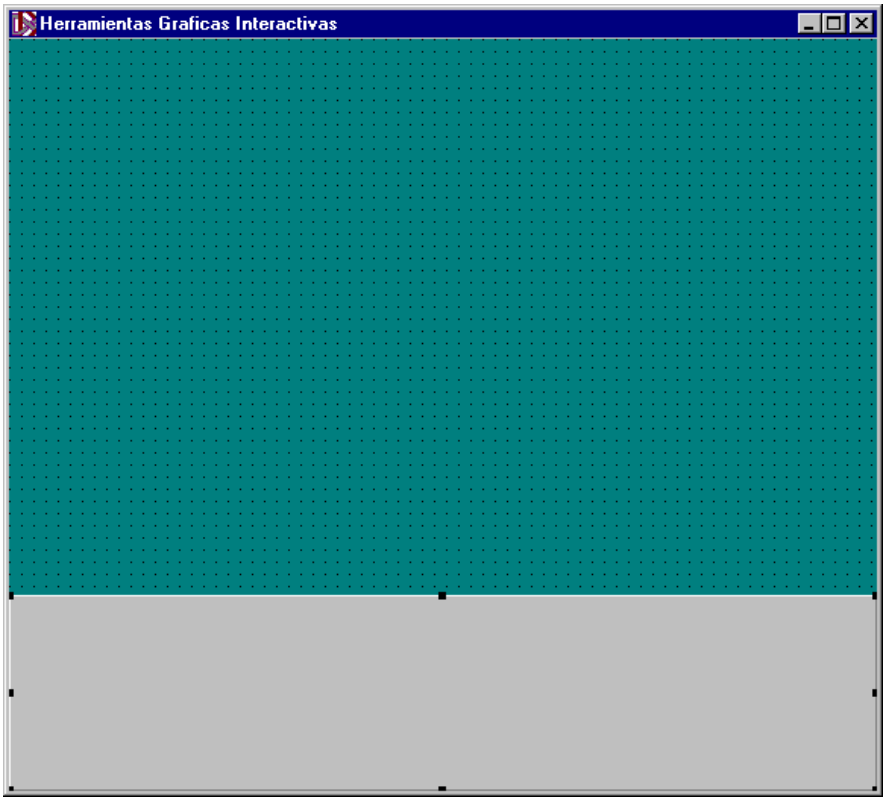


FIG. 4.6.2 Diseño de la forma.

5.- De la paleta de componentes Standard seleccione un objeto Button 

y colóquelo en la forma repitiendo la acción 4 veces mas colocándolos en el panel.

6.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Introducción.
b) La propiedad Height	⇒	32
c) La propiedad Width	⇒	209
d) La propiedad Left	⇒	40
e) La propiedad Top	⇒	14

Haga doble clic en el botón introducción e introduzca el siguiente código.

```
WinExec("SIMULA.EXE",SW_SHOWDEFAULT);
```

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Coordenadas Rectangulares.
b) La propiedad Height	⇒	32
c) La propiedad Width	⇒	227
d) La propiedad Left	⇒	270
e) La propiedad Top	⇒	14

Haga doble clic en el botón Coordenada Rectangulares e introduzca el siguiente código.

```
WinExec("PUNTO1.EXE",SW_SHOWDEFAULT);
```

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Distancia Entre Puntos.
b) La propiedad Height	⇒	32
c) La propiedad Width	⇒	178
d) La propiedad Left	⇒	15
e) La propiedad Top	⇒	72

Haga doble clic en el botón Distancia Entre Puntos e introduzca el siguiente código.

```
WinExec("P2.EXE",SW_SHOWDEFAULT);
```

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Pendiente.
b) La propiedad Height	⇒	32
c) La propiedad Width	⇒	161
d) La propiedad Left	⇒	205
e) La propiedad Top	⇒	72

Haga doble clic en el botón Pendiente e introduzca el siguiente código.

```
WinExec("N_pend.EXE",SW_SHOWDEFAULT);
```

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Ecuaciones de la Recta.
b) La propiedad Height	⇒	32
c) La propiedad Width	⇒	154
d) La propiedad Left	⇒	380
e) La propiedad Top	⇒	72

Haga doble clic en el botón Ecuaciones de la Recta e introduzca el siguiente código.

```
WinExec("MXMASB.EXE",SW_SHOWDEFAULT);
```

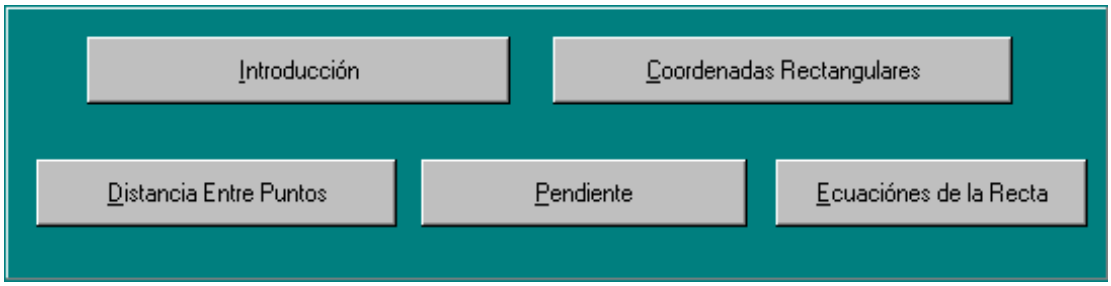


FIG. 4.6.3 Botones Controladores.

7.- De la paleta de componentes Additional seleccione un objeto Image  y colóquelo en la forma.

8.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	312
b) La propiedad Width	⇒	264
c) La propiedad Left	⇒	8
d) La propiedad Top	⇒	8
e) La propiedad Stretch	⇒	True
f) La propiedad Picture	⇒	Doble clic

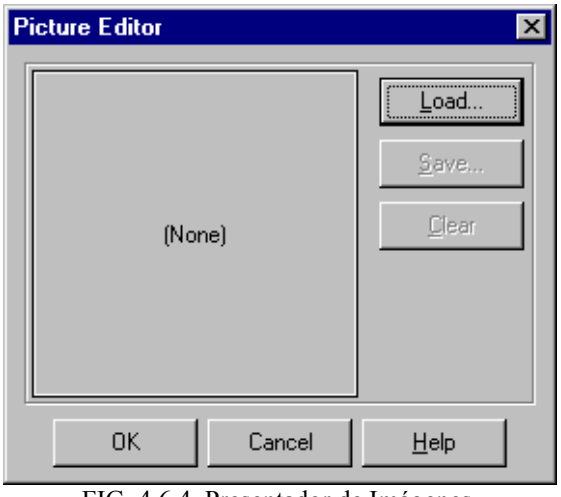


FIG. 4.6.4 Presentador de Imágenes.

Presione el botón Load y obtendrá una caja de dialogo para seleccionar el icono deseado o podrá buscarlo en caso de no encontrarse en el directorio mostrado.

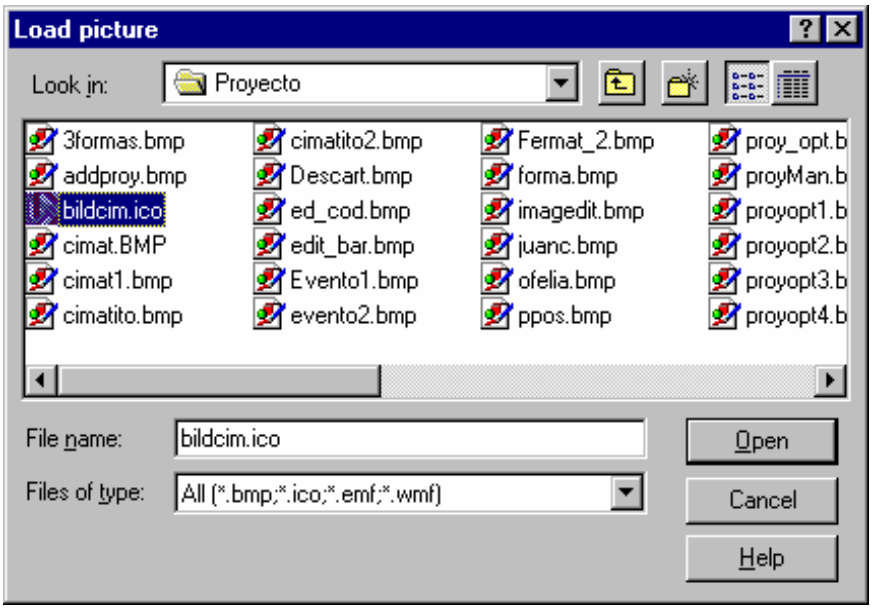


FIG. 4.6.5 Selección de Imágenes.

Seleccione la imagen deseada y oprima el botón Open.

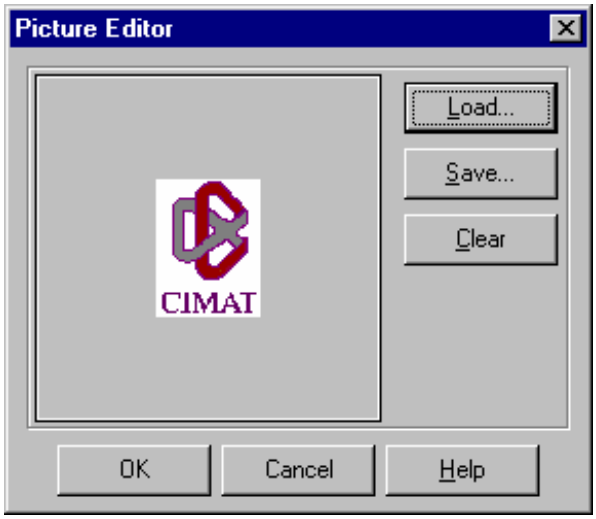


FIG. 4.6.6 Selección de Imágenes.

Oprima el botón Ok.

9.- De la paleta de componentes Additional seleccione un objeto Image  y colóquelo en la forma.

10.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	312
b) La propiedad Width	⇒	264
c) La propiedad Left	⇒	280
d) La propiedad Top	⇒	8
e) La propiedad Stretch	⇒	True
f) La propiedad Picture	⇒	Doble clic

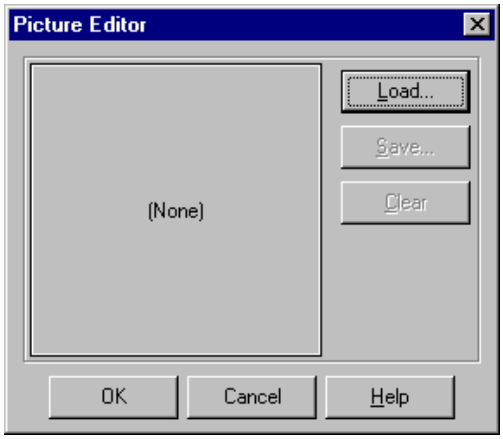


FIG. 4.6.7 Presentador de Imágenes.

Presione el botón Load y obtendrá una caja de dialogo para seleccionar el icono deseado o podrá buscarlo en caso de no encontrarse en el directorio mostrado.

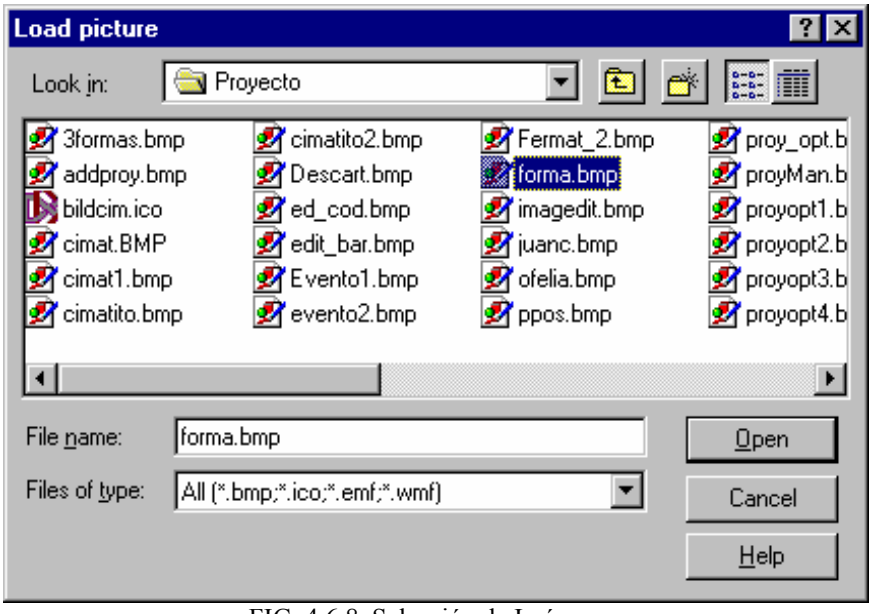


FIG. 4.6.8 Selección de Imágenes.

Seleccione la imagen deseada y oprima el botón Open.

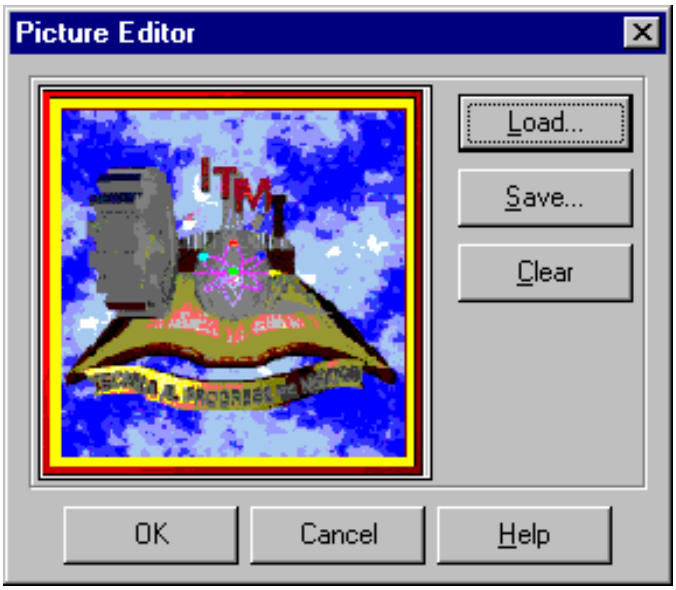


FIG. 4.6.9 Selección de la Imagen.

Oprima el botón Ok.

Con estos pasos tendrá las imágenes disponibles ya en su forma.

11.- De la paleta de componentes Standard seleccione un objeto Panel  y colóquelo en la forma.

12.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Se elimina, ya que no se usará.
b) La propiedad Height	⇒	216
c) La propiedad Width	⇒	209
d) La propiedad Color	⇒	“ clGray “
e) La propiedad Left	⇒	192
f) La propiedad Top	⇒	64

13.- De la paleta de componentes Additional seleccione un objeto Image  y colóquelo en el Panel2.

14.- Modificamos sus propiedades de la siguiente manera:

Propiedad		Valor o Texto
a) La propiedad Height	⇒	104
b) La propiedad Width	⇒	88
e) La propiedad Left	⇒	56
f) La propiedad Top	⇒	8

Repetimos las instrucciones del paso 8 para cargar el BMP de CIMAT.

15.- De la paleta de componentes Standard seleccione un objeto Button 

y colóquelo en la forma.

16.- Modifiquemos los siguientes atributos:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Continuar
b) El evento OnClick	⇒	Panel2->Visible=false;

17.- De la paleta de componentes Standard seleccione un Label 

y colóquelo en la forma con otras dos copias.

18.- Modifiquemos los siguientes atributos:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	Geometría Parte 1.
b) La propiedad Height	⇒	24
c) La propiedad Width	⇒	179
d) La propiedad Color	⇒	“ clGray “
e) La propiedad Left	⇒	16
f) La propiedad Top	⇒	120
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	16

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	JUAN CARLOS CRUZ VEGA
b) La propiedad Height	⇒	13
c) La propiedad Width	⇒	167
d) La propiedad Color	⇒	“ clGray “

e) La propiedad Left ⇒ 16
f) La propiedad Top ⇒ 144
g) La propiedad Font Style ⇒ Bold Italic
h) La propiedad Font Size ⇒ 9

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	FEBRERO DE 1998
b) La propiedad Height	⇒	13
c) La propiedad Width	⇒	118
d) La propiedad Color	⇒	“ clGray “
e) La propiedad Left	⇒	32
f) La propiedad Top	⇒	160
g) La propiedad Font Style	⇒	Bold Italic
h) La propiedad Font Size	⇒	9

19.- De la paleta de componentes Standard seleccione un objeto Main Menu  y colóquelo en la forma.

20.- haga doble clic en este componente y obtendrá el diseñador de menús:

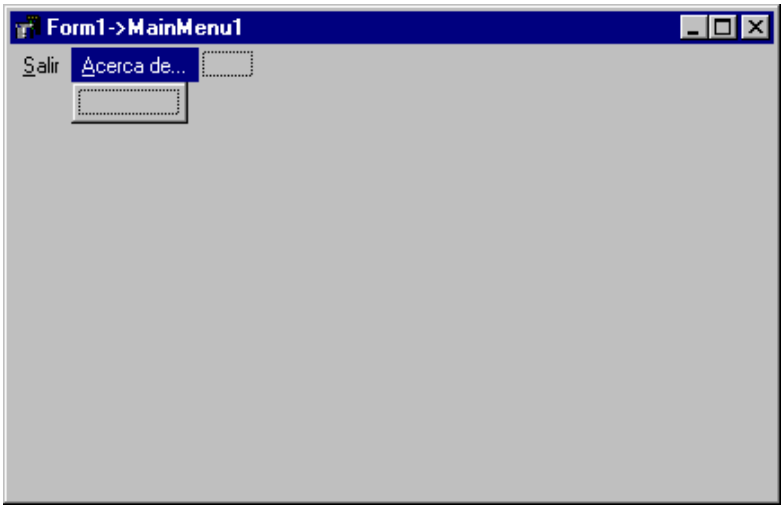


FIG. 4.6.10 Diseñador de menús.

21.- Con el inspector de objetos cambie las propiedades:

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Salir
b) El evento OnClick	⇒	exit(0);

Propiedad		Valor o Texto
a) La propiedad Caption	⇒	&Acerca de…
b) El evento OnClick	⇒	Panel2->Visible=true;

22.- Coloque el siguiente código en la función mostrada:

```
__fastcall TForm1::TForm1(TComponent* Owner)
        : TForm(Owner)
{
  Application->HelpFile = "DOCUMENTO.HLP";
}
```

23.- Presione la tecla F9 para ejecutar la aplicación:

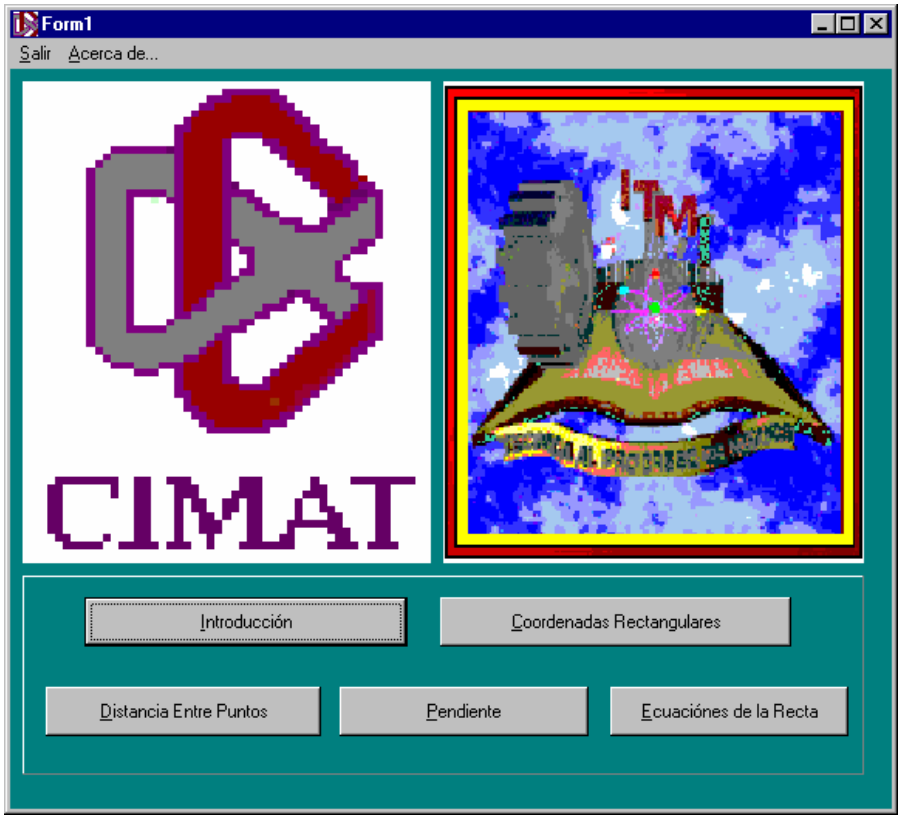


FIG. 4.6.11 Aplicación final.

4.7 ELABORACION DEL HIPERTEXTO.

Para la elaboración del hipertexto, tomé en cuenta los módulos que se elaboraron, así que cada módulo tendrá en su aplicación una opción de revisión del texto asociado al tema:

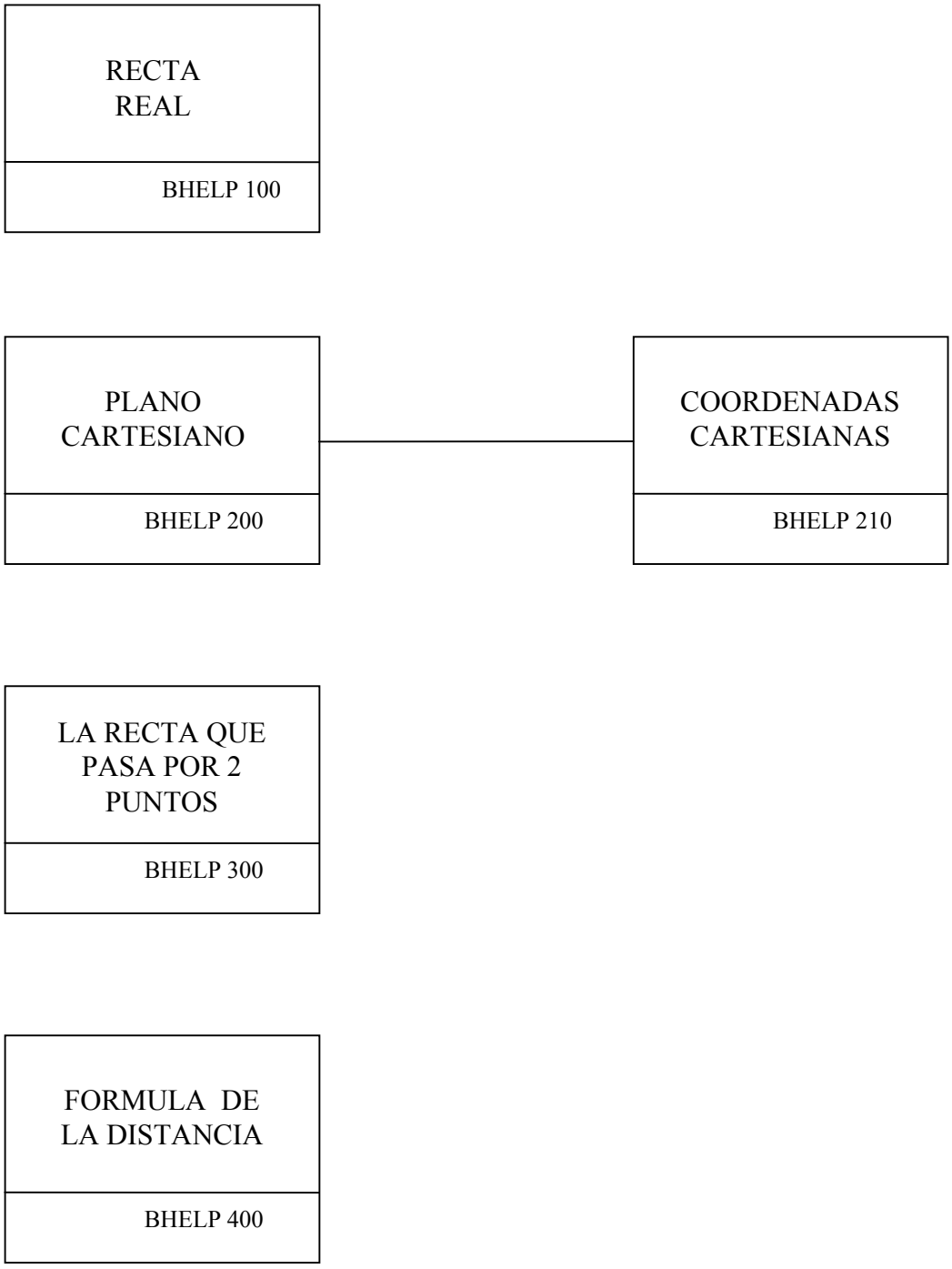
Los Módulos fueron los siguientes:

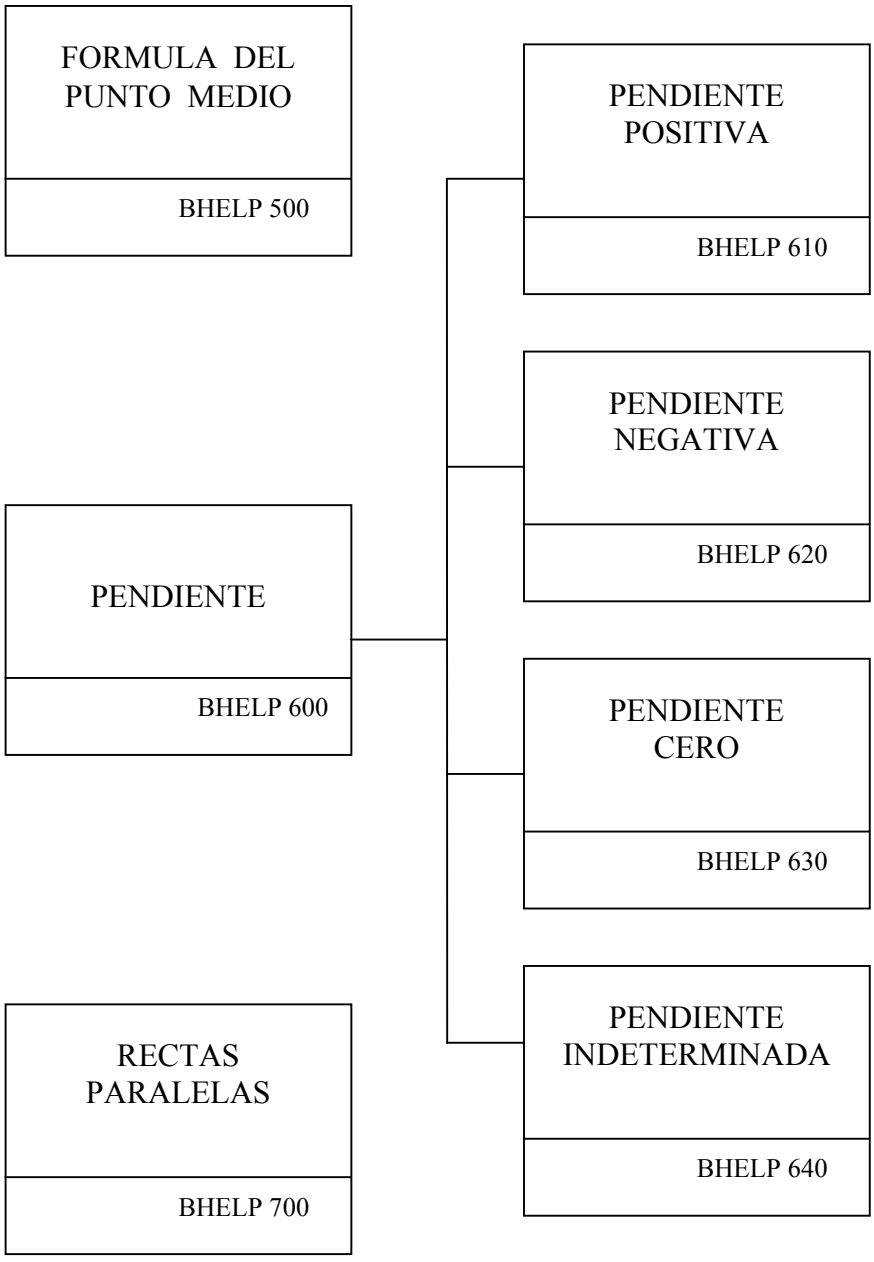
1.- Puntos en el sistema coordenado cartesiano.
2.- Distancia entre dos puntos.
3.- Pendiente formada por la recta que pasa por dos puntos.
4.- Ecuaciones de la recta

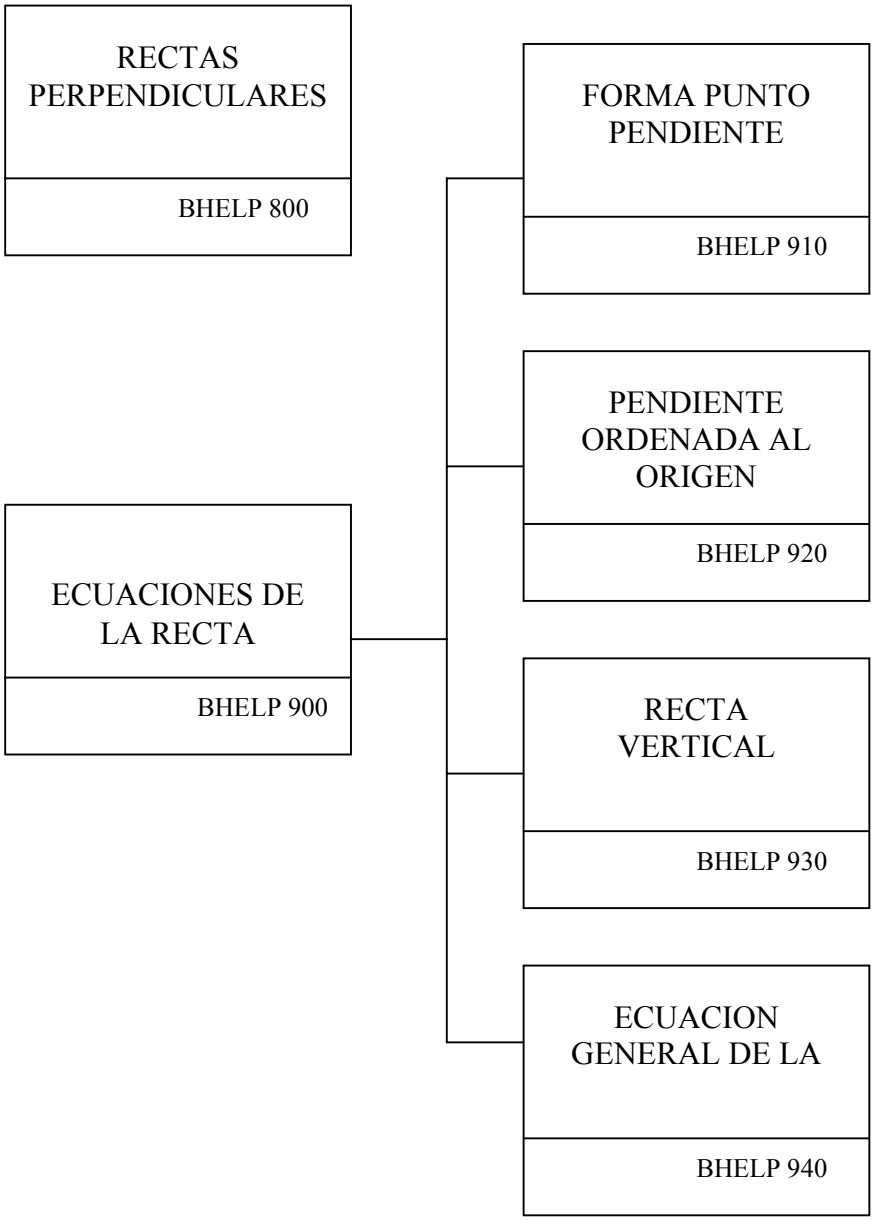
Con estos puntos bien marcados, comencé a recopilar material sobre los distintos temas, en los libros mas comunes de Geometría Analítica, sobre el siguiente programa:

- Recta real.
- Plano cartesiano.
- Coordenadas cartesianas.
- La recta que pasa por dos puntos.
- Formula de la distancia.
- Formula del punto medio.
- Pendiente
  - Pendiente positiva.
  - Pendiente negativa.
  - Pendiente cero.
  - Pendiente Indeterminada.
- Ecuaciones de la recta.
  - Forma punto pendiente
  - Forma pendiente ordenada al origen.
  - Ecuación de una recta vertical.
  - Ecuación general de la recta.
- Rectas Paralelas.
- Rectas perpendiculares.

Con el material recopilado, procedí a esquematizar como presentaría cada uno de los siguientes temas y cuales se relacionarían, por lo que decidí hacer el siguiente esquema:


RECTA
REAL
BHELP 100
PLANO
CARTESIANO
BHELP 200
COORDENADAS
CARTESIANAS
BHELP 210
LA RECTA QUE
PASA POR 2
PUNTOS
BHELP 300
FORMULA DE
LA DISTANCIA
BHELP 400


FORMULA DEL PUNTO MEDIO
BHELP 500
PENDIENTE
BHELP 600
RECTAS PARALELAS
BHELP 700
PENDIENTE POSITIVA
BHELP 610
PENDIENTE NEGATIVA
BHELP 620
PENDIENTE CERO
BHELP 630
PENDIENTE INDETERMINADA
BHELP 640


RECTAS PERPENDICULARES
BHELP 800
ECUACIONES DE LA RECTA
BHELP 900
FORMA PUNTO PENDIENTE
BHELP 910
PENDIENTE ORDENADA AL ORIGEN
BHELP 920
RECTA VERTICAL
BHELP 930
ECUACION GENERAL DE LA
BHELP 940

Antes de iniciar con el desarrollo del hipertexto, debo mencionar, que existen varios programas que ya realizan el hipertexto, usted solo tiene que introducir el texto y controlar sus relaciones; o existen varios editores de texto que en la opción de salvar el documento, cuentan con la opción de salvar en formato RTF (Rich Text Format); que es el formato que nos ayuda para la compilación de este tipo de textos.
Para este trabajo en particular utilicé Word95 V6.0.

Como nota adicional puedo comentar que en Word97 la opción de salvar con formato RTF y algunas otras aplicaciones no funcionan, por lo que utilicé la versión 6.0 de Word.

PROCEDIMIENTO

Agrupé los temas que veía más relacionados, y los dividí por números de nivel, para saber cuando un nivel superior llama a uno inferior, y saber como regresar paso a paso por los niveles en los que se baje.

En la presentación del hipertexto pondré un menú que llamaré *CONTENIDO* y tendrá las siguientes directivas:

Directiva	**Acción**
#	Define un identificador único para el tópico.
$	La palabra que lo acompaña tiene un tópico asociado.
+	Realiza la acción de paso atrás o adelante.
K	La palabra que lo acompaña es la palabra clave para el Indexado.

Para asociarle un texto a una palabra, se hace lo siguiente:

Si a la palabra *Recta* le quiero asociar un texto que se encuentra en otra sección del escrito:
1.- Primero la marco con doble subrayado.

Recta

2.- Junto a la palabra le escribo el identificador del texto al cual quiero asociarlo.

*Recta*Ir_Recta

3.- La palabra Ir_Recta la Marco con línea oculta.

4.- El texto que está asociado debe estar en otra hoja u otras hojas pero debe tener el siguiente encabezado.

#$K+RECTA

y en el pie de página la distribución sería la siguiente:

Ir_Recta
$ Ir_Recta
K Ir_Recta
+ BHELP:110

Donde el Número 110 indica que tiene un tema predecesor marcado por el número 100, y posiblemente este tenga un predecesor con el número 10.

Este texto está grabado en un formato RTF.

5.- Con el archivo almacenado con el formato rtf procedemos a la compilación de este archivo para generar el archivo de ayuda.

6.- Abrimos una aplicación en C++ Builder.

7.- De la barra de menú principal seleccionamos. Tools $\Rightarrow$ Hcw.exe

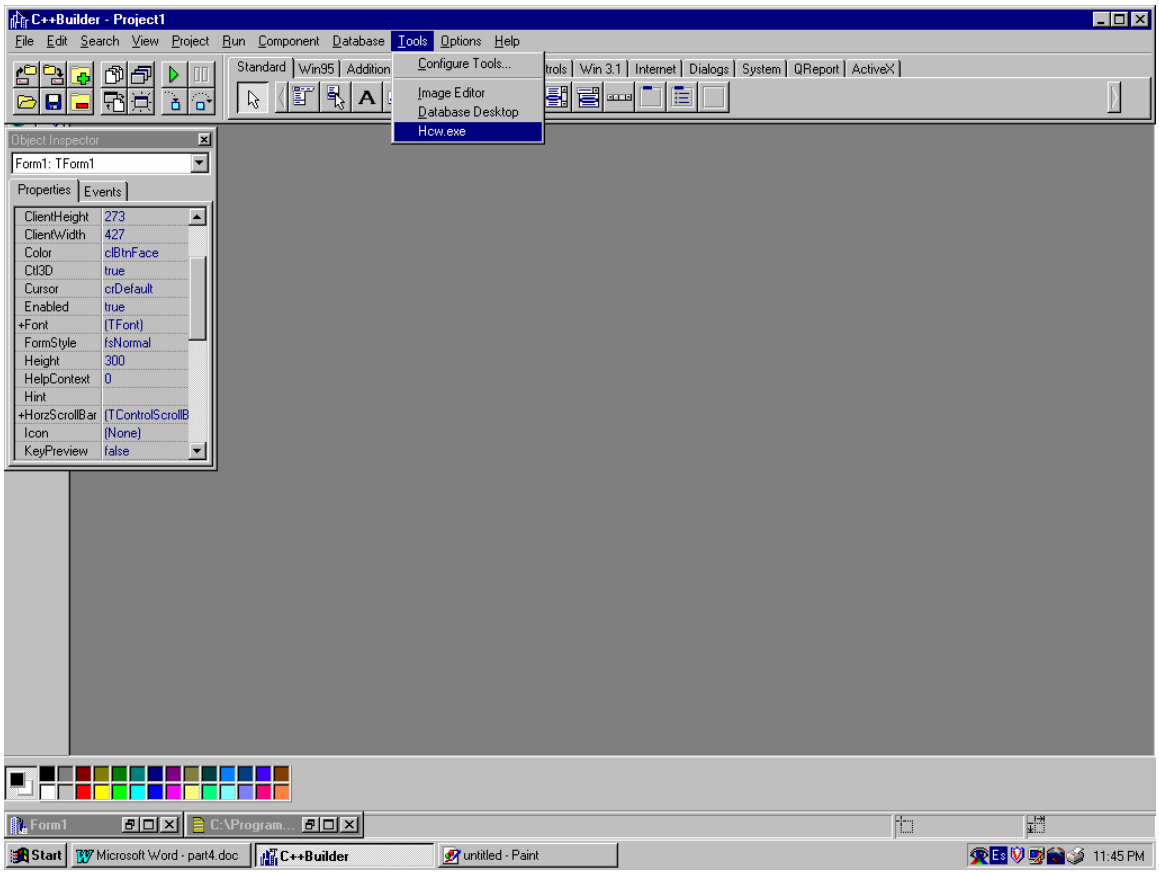


FIG. 4.7.1 Compilador para la elaboración del hipertexto.

Si no se encuentra en este sub menú la utilidad Hcw.exe,

Seleccione del mismo menú tools ⇒ Configure Tools y obtendrá la siguiente ventana de diálogo:

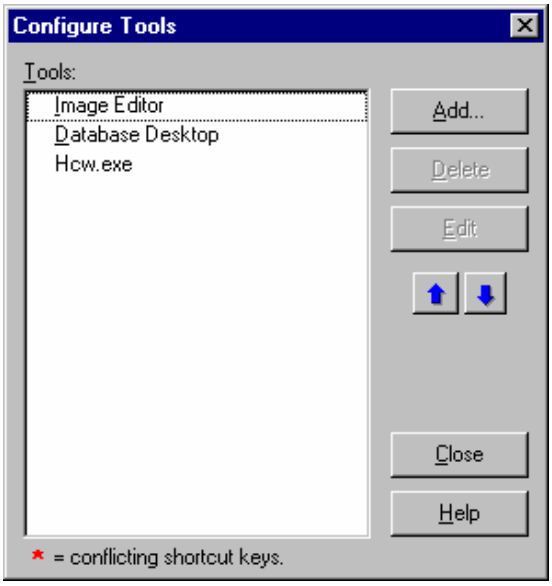


FIG. 4.7.2 Configurado de Herramientas.

En ella podrá añadir la utilidad Hcw.exe marcándola y oprimiendo el botón Add.

8.- Ya seleccionado el Hcw.exe obtendrá la siguiente ventana:

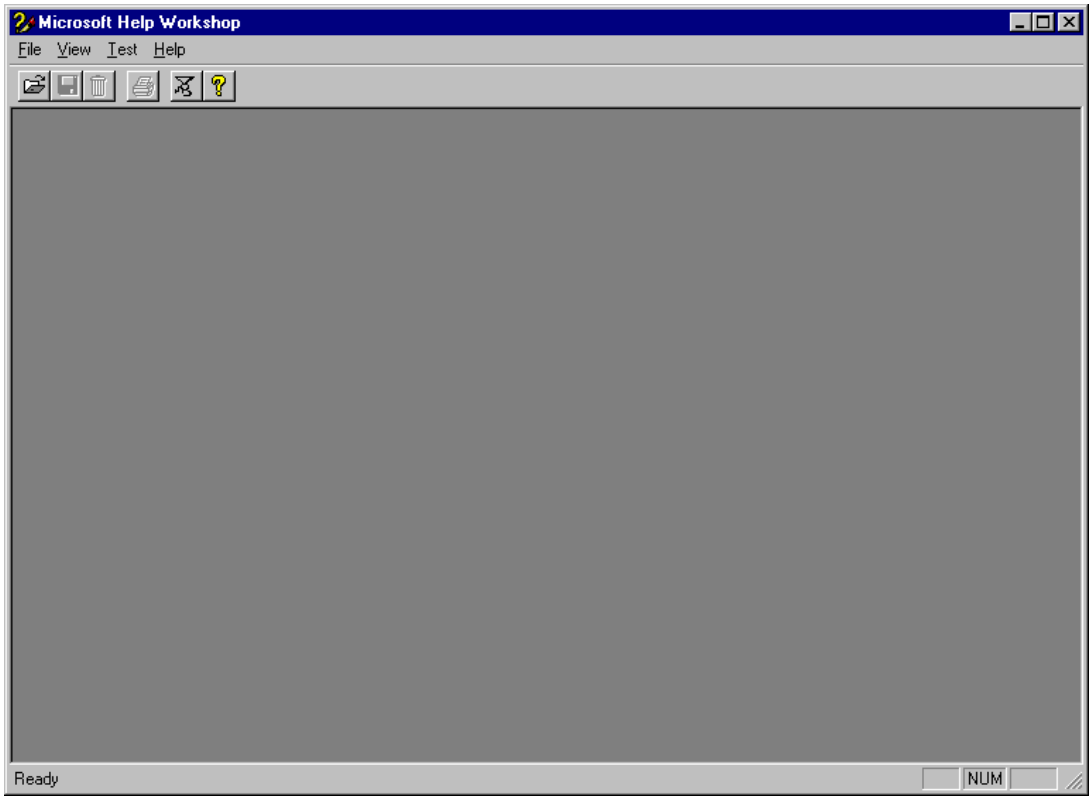


FIG. 4.7.3 Compilador de ayudas.

9.- Seleccione de ella en el menú File New y obtendrá otra ventana.

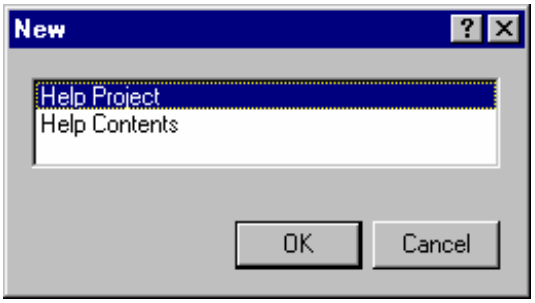


FIG. 4.7.4 Selección de tipo de proyecto.

10.- En ella seleccione Help Project y presione el botón OK.

11.- Obtendrá otra ventana en la que dará nombre a la nueva aplicación de ayuda.

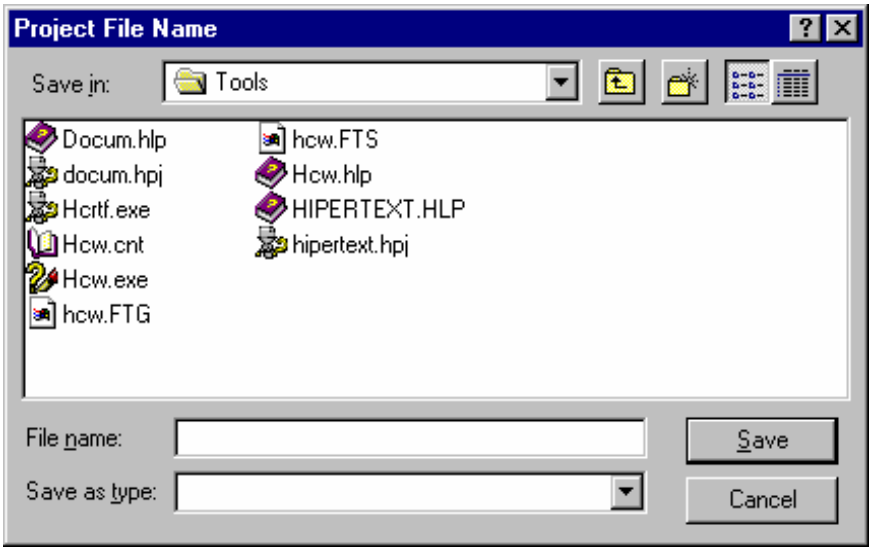


FIG. 4.7.5 Dar nombre a un proyecto .HLP.

12.- Cuando le de nombre aparecerá la siguiente ventana:

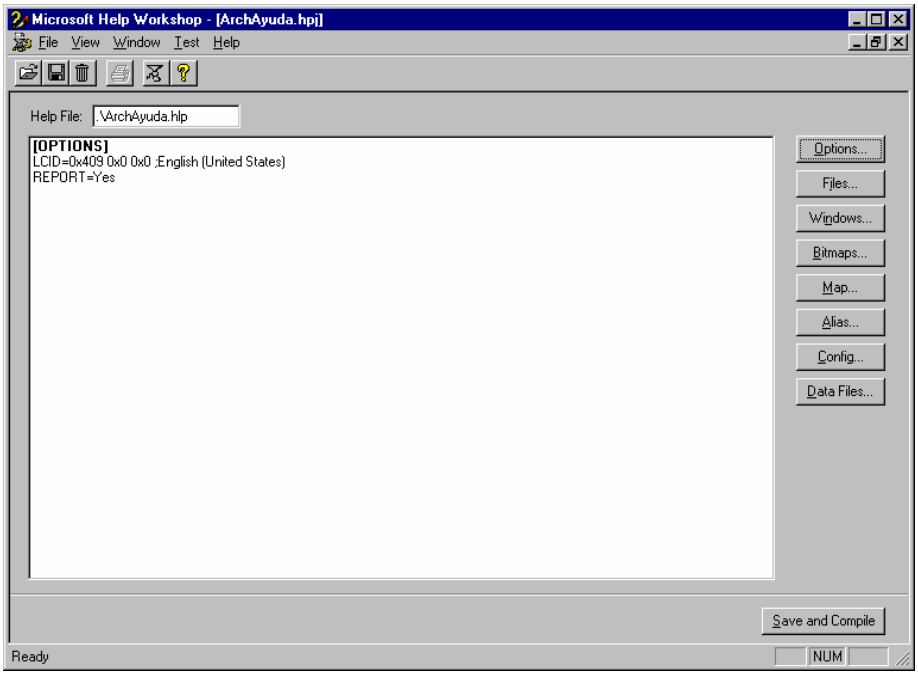


FIG. 4.7.6 Integración de componentes.

13.- En la añadirá el o los archivos que tenga para la elaboración de la ayuda. (Archivos *.rtf), la ruta donde se encuentren las imágenes que valla a llevar la aplicación. Los mapas o las bases de datos. Todo esto según el botón.

14.- Solo pondré un archivo rtf.

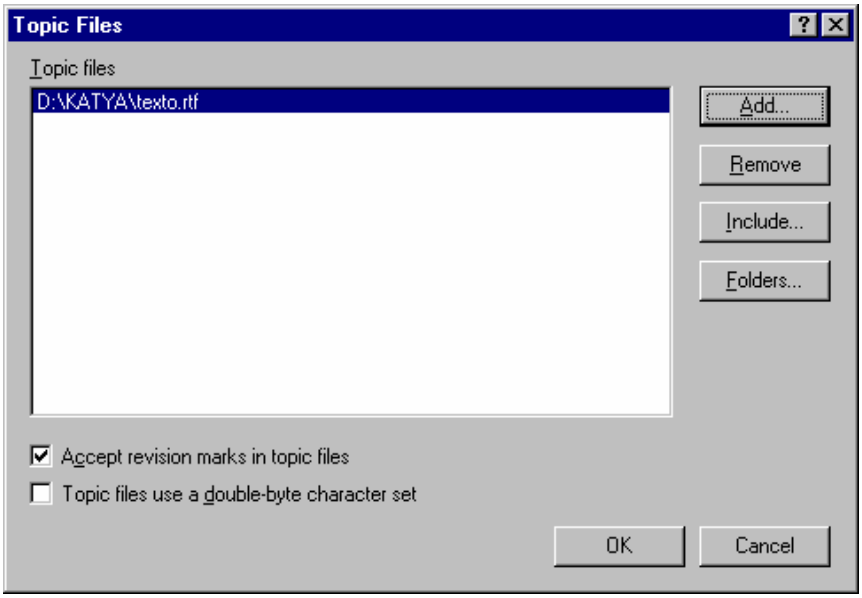


FIG. 4.7.7 Añadir archivo RTF.

Y se oprime el botón OK.

15.- Oprima el botón Save and Compile.

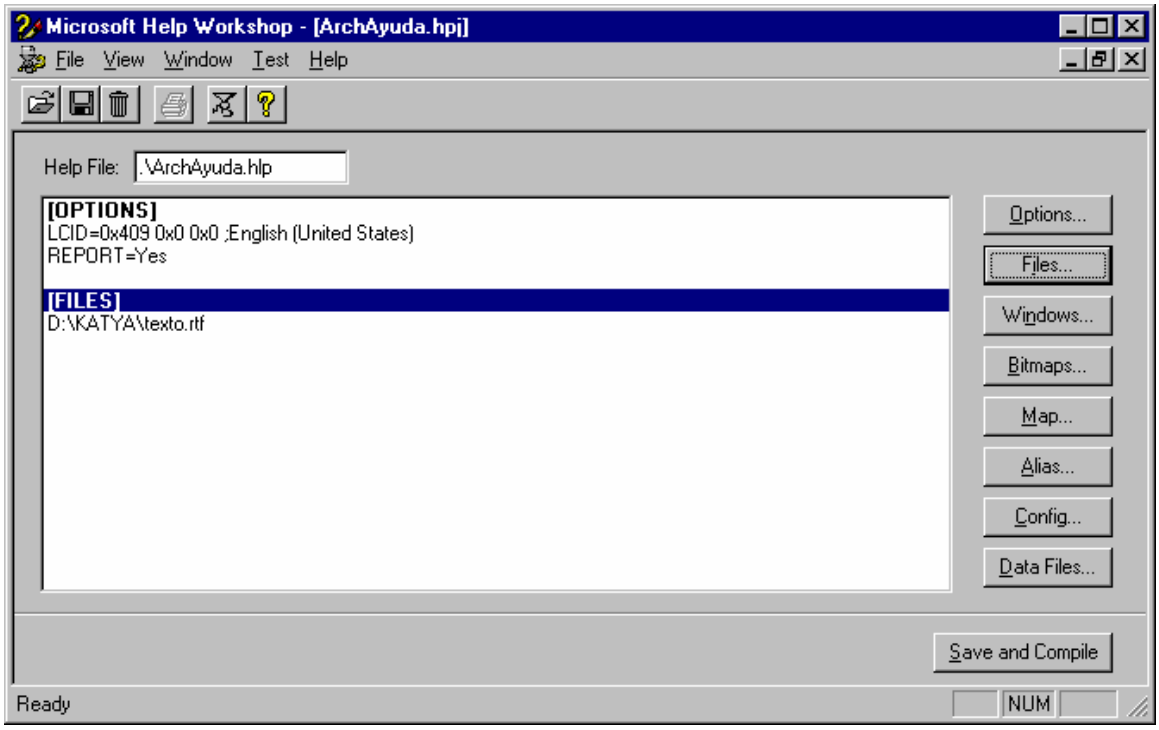


FIG. 4.7.8 Lista de componentes integrados.

16.- y nos mostrará el resultado de la compilación, si hay errores se tendrán que corregir.

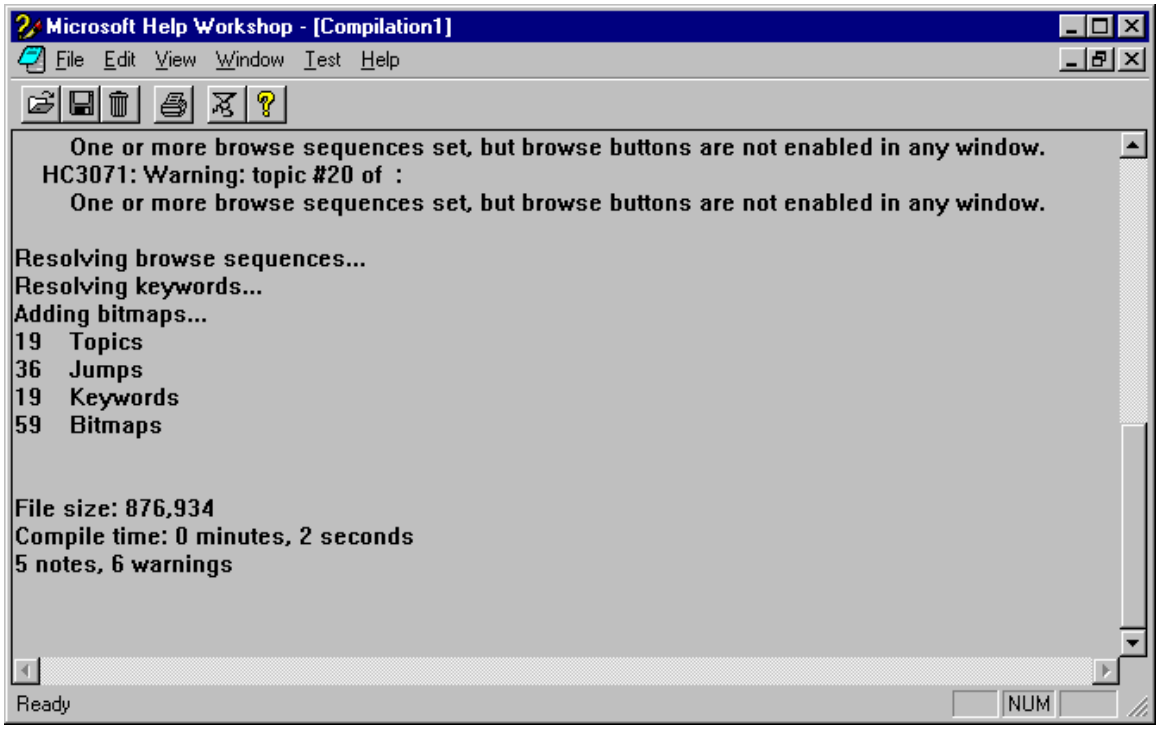


FIG. 4.7.9 Resultado de la compilación.

17.- Si no hay errores se oprime el botón  , y presentará otra ventana.

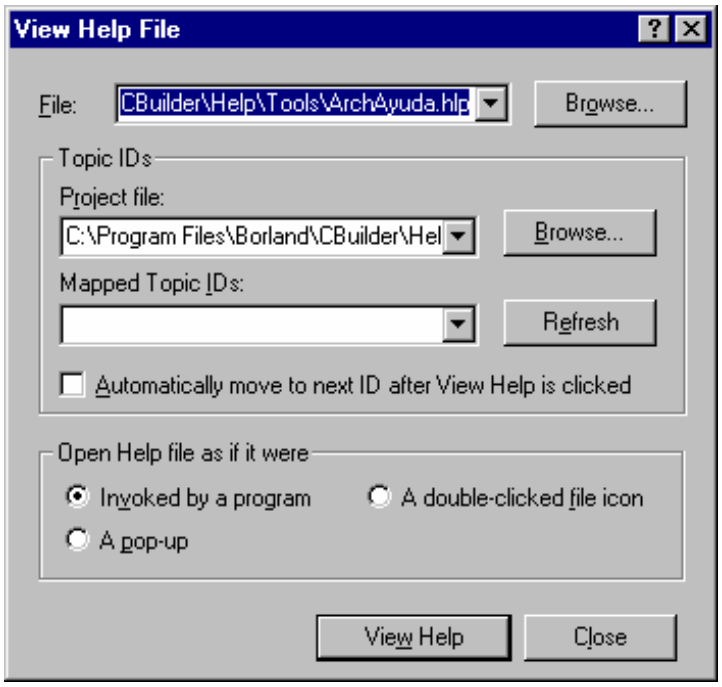


FIG. 4.7.10 Visualización de la ayuda.

18.- Oprima el botón View Help. Y tendrá la ejecución de la aplicación.

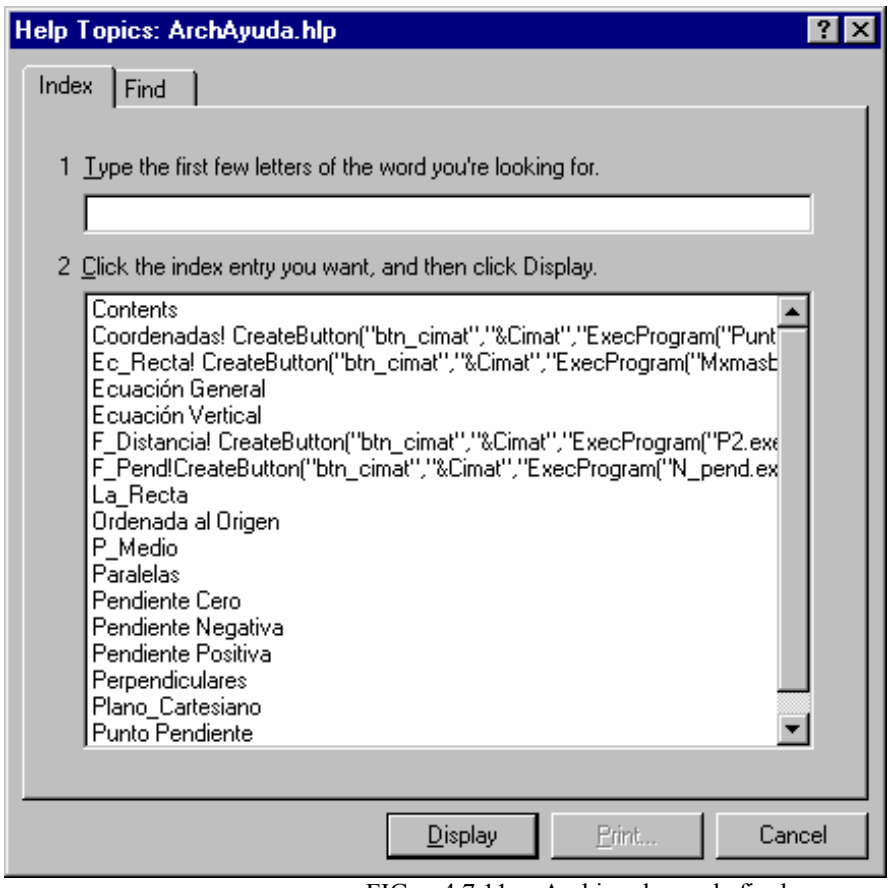


FIG. 4.7.11 Archivo de ayuda final.

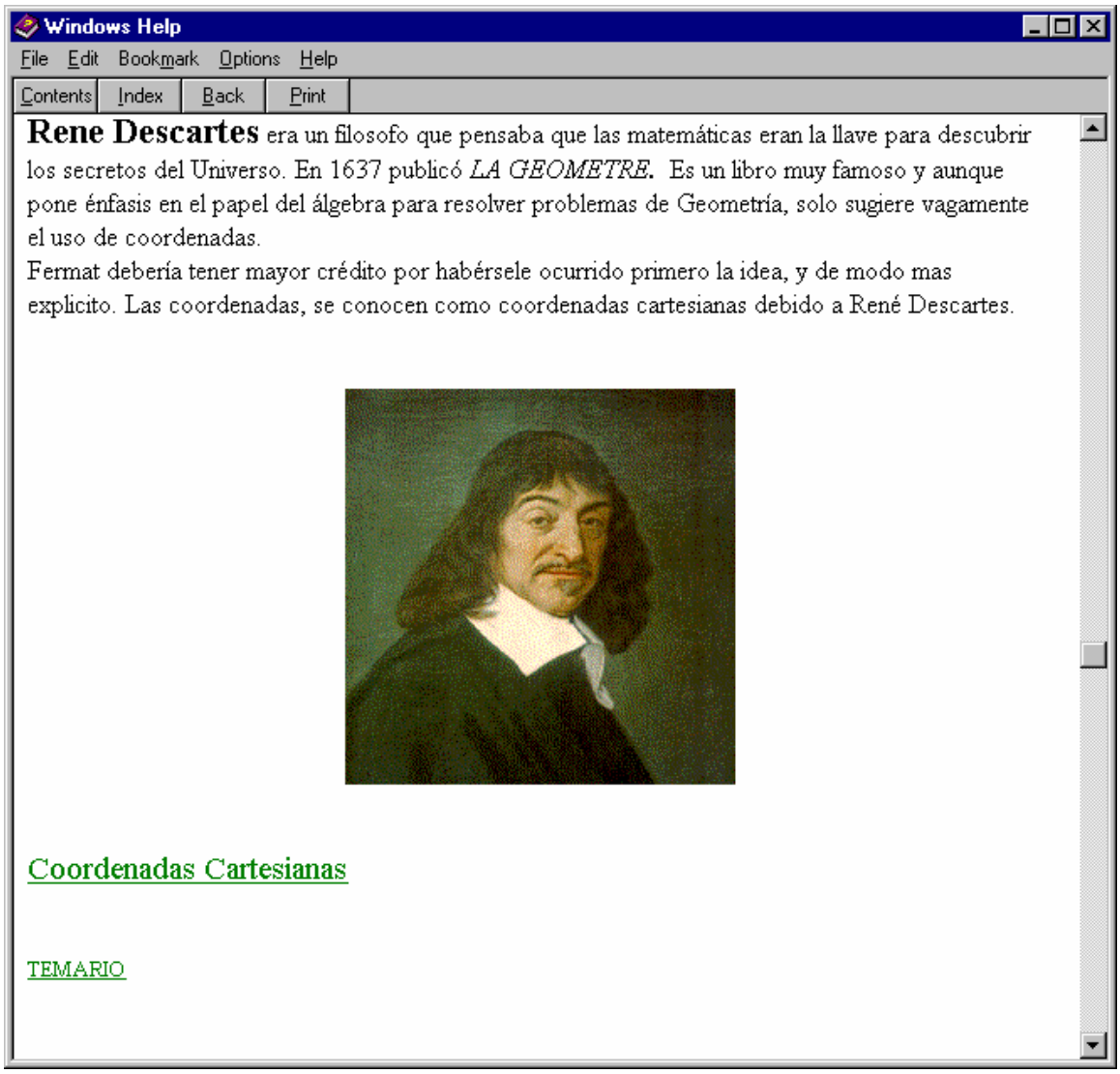


FIG. 4.7.12 Ejemplo de una consulta en el archivo generado.

Ya tenemos un archivo ejecutable que nos muestra el hipertexto, este hipertexto lo podemos añadir a un programa en C++ Builder de la siguiente manera:

Application->HelpFile = "DOCUMENTO.HLP";

Donde DOCUMENTO.HLP es el nombre del hipertexto ya generado después de la compilación con el HCW.EXE

Podemos colocarlo en la función que crea la forma de la siguiente manera:

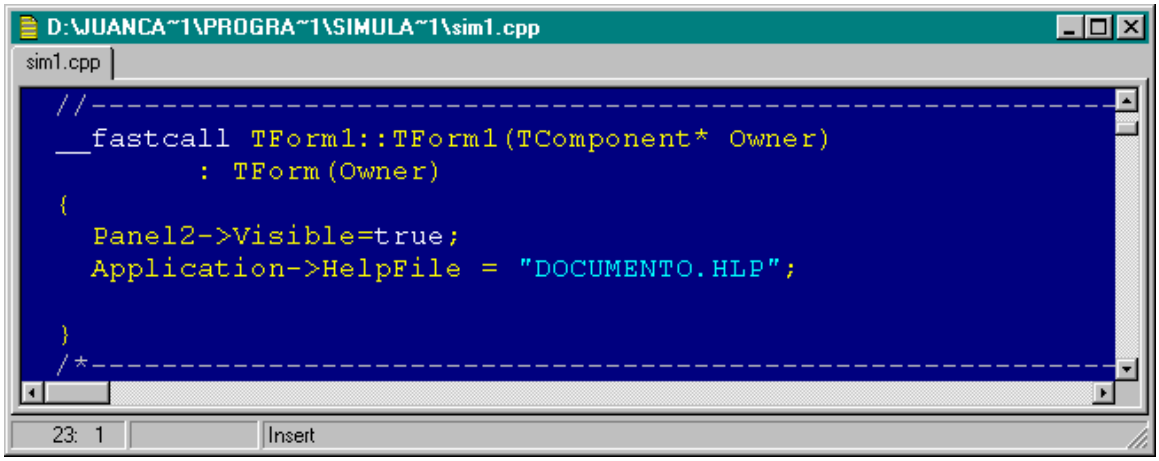


FIG. 4.7.13 Liga en un programa C++ Builder.

Y el llamado particular es según la opción del menú en la que se halla puesto:
Ejemplo:

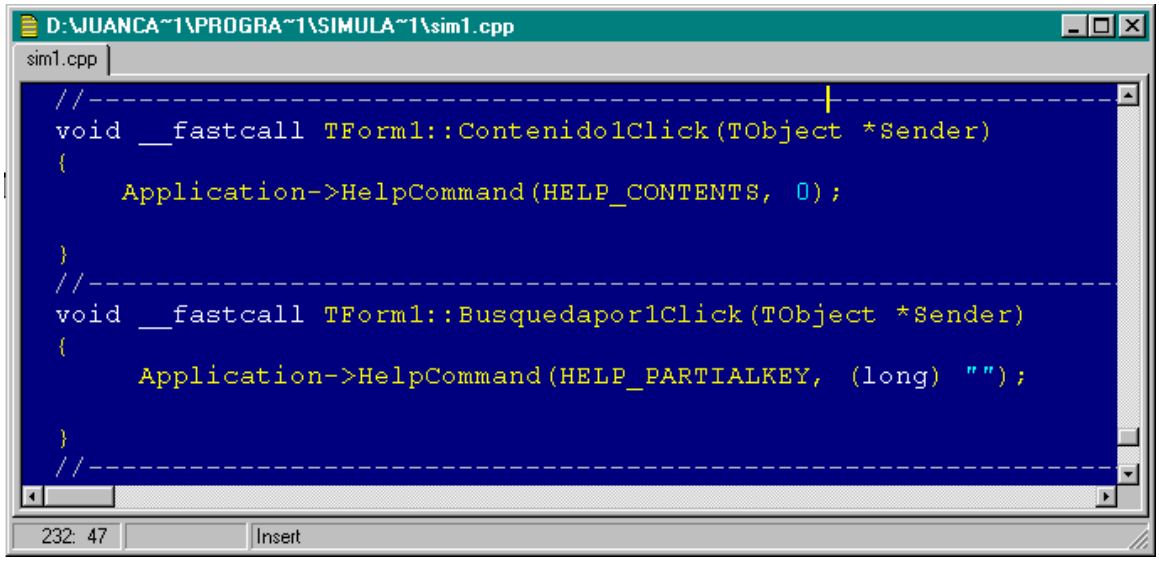


FIG. 4.7.14 Llamadode un módulo .hlp en un programa C++ Builder.

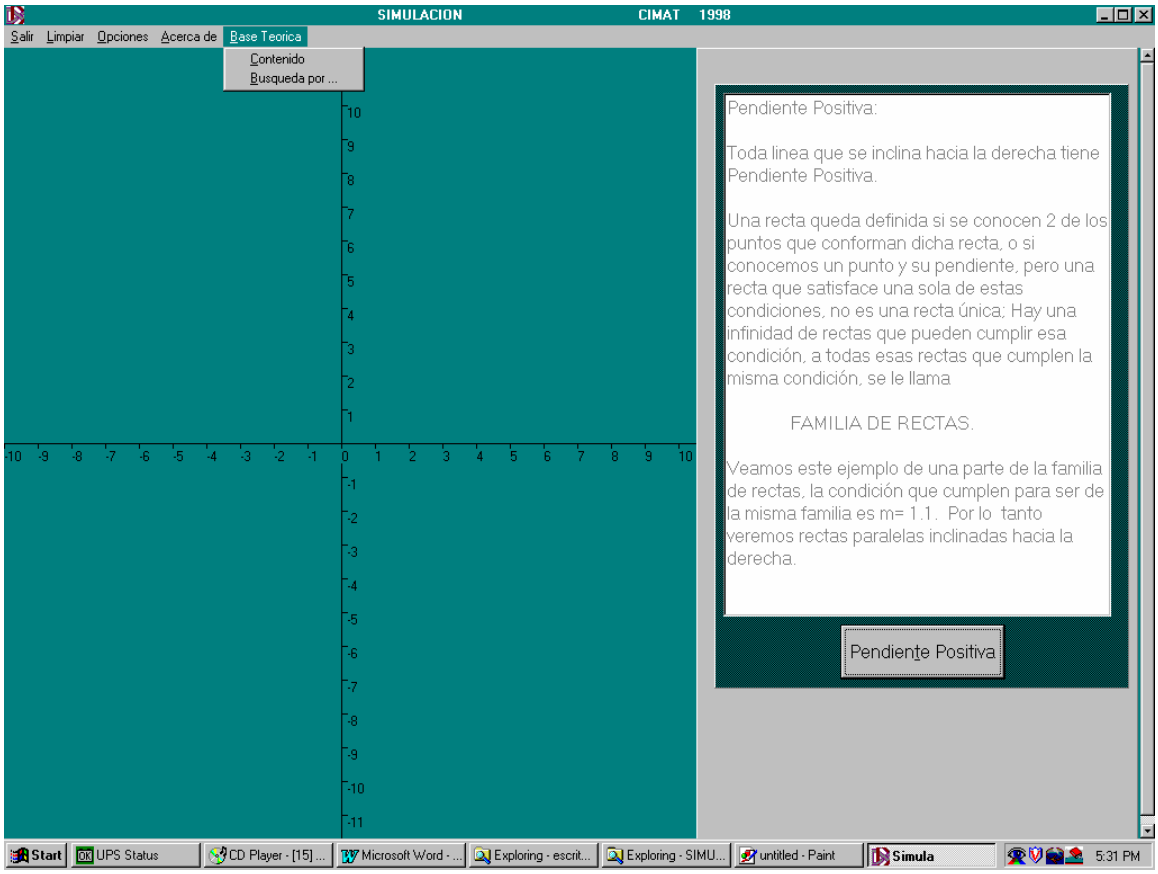


FIG. 4.7.15 Opcion de menú en el que es llamado un módulo de hipertexto.

FIG. 4.7.16 Opcion de menú en el que es llamado un módulo de hipertexto.

Esto es a grandes razgos la creación de un hipertexto en C++ Builder.