AMBIENTE INTEGRADO C++

DEV-C++

UNIVERSIDAD NACIONAL DE COLOMBIA

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA
DE SISTEMAS E INDUSTRIAL

BOGOTÁ D.C.
2004

TABLA DE CONTENIDO

1 Introducción

El presente manual, no es un manual del lenguaje C++, es un manual, para el uso de un entorno de desarrollo de código como lo es Dev-C++, por lo tanto no encontrará ayudas sobre la referencia del lenguaje, ni explicación de la sintaxis. Este manual presenta las facilidades de esta herramienta para el desarrollo de código.

Dev-C++ es un Entorno de Desarrollo Integrado (IDE) para el lenguaje de Programación C/C++ que usa Mingw de GCC (GNU Compiler Collection) como Compilador y permite crear:

- Programas ejecutables para Win32.
- Programas ejecutables para consola.
- Construcción de DLL´s y bibliotecas estáticas.
- Además, se puede utilizar en combinación con otros compiladores basados en GCC.

Dev-C++ es un software de libre distribución sujeto a los términos de la Licencia Pública General (GPL), que facilita la escritura de programas en lenguaje C++; para brindar las facilidades de creación de programas (ejecutables *.exe) se apoya en diversos compiladores, entre ellos MinGW (Minimalist Gnu Windows), también software libre bajo la licencia GNU

Las características de Dev-C++ son:

- Tiene integrado un depurador basado en GDB
- Soporta múltiples lenguajes (el castellano es uno de ellos).
- Mantiene una lista con las clases utilizadas en el desarrollo de un programa.
- Mantiene la lista de funciones definidas en el desarrollo del programa.
- Tiene un manejador de proyectos (se usa cuando el programa se compone de más de un fichero fuente).
- Tiene un editor que resalta la sintaxis del código que es configurable.
- Soporta plantillas para la creación de tus propios tipos de proyectos.
- Permite la creación de Makefile para la compilación separada de archivos fuente.
- Edita y compila ficheros de recursos.
- Soporta la actualización del software y bibliotecas a través de Internet.

Este manual fue elaborado por Nelson Javier Cruz, como un aporte del Centro de estudios de Ingeniería de sistemas CEIS_UN, al departamento de Sistemas e industrial de la universidad Nacional de Colombia. aportes y recomendaciones a: ceis_un@hotmail.com

2 Instalación

La instalación es muy sencilla, es necesario tener un equipo con Windows, funciona en todas las versiones de Windows, excepto las inferiores a win 95, las exigencias de memoria y disco son MUY pequeñas.
Estas son las especificaciones mínimas necesarias, que requiere el computador:

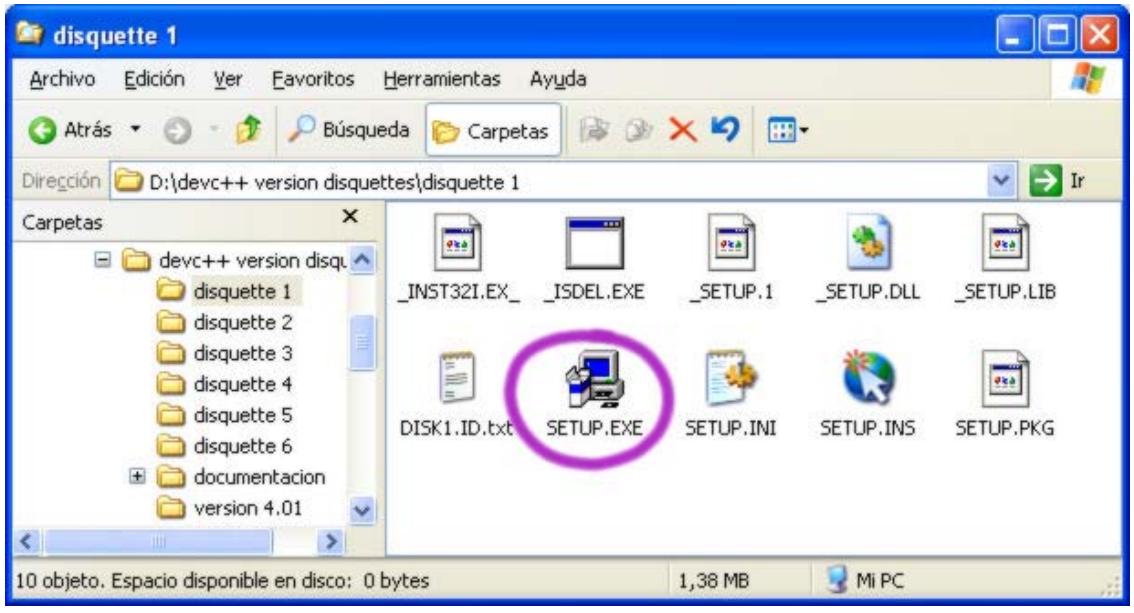
- Microsoft Windows 95, 98, NT 4, 2000, XP, 2003
- 8 MB RAM procesador de 100 Mhz (486 o Pentium)
- 30 MB de espacio en disco.

Él programa se puede conseguir en:
http://www.bloodshed.net/dev/devcpp.html

Aunque en este manual, no se presenta la última versión, se presentará una lo suficientemente versátil para desarrollar los objetivos propuestos en los cursos de programación de computadores y métodos numéricos.

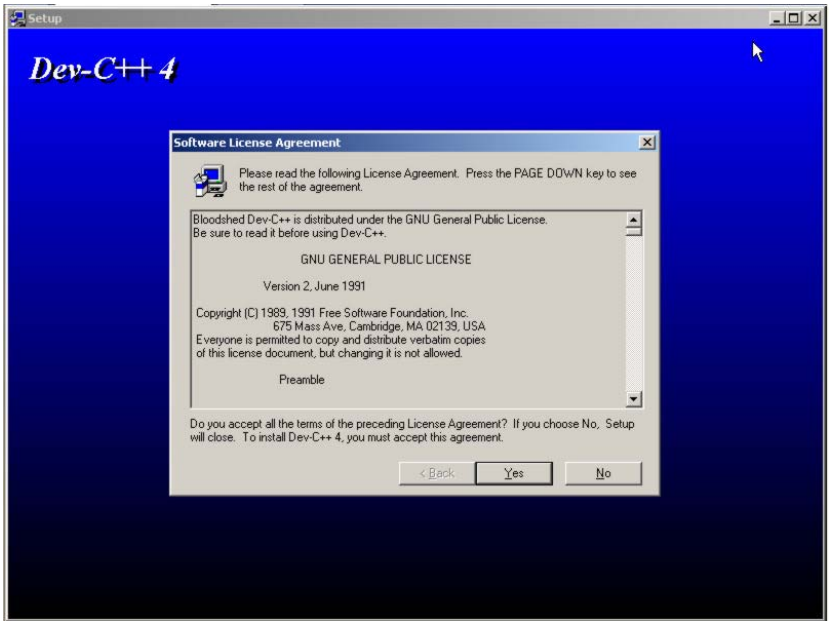
Es posible descargarlo en un solo archivo comprimido de 8Mb, y en versión de 6 disquettes, de la página antes indicada. También en las salas de computadores de la Facultad de Ingeniería (453-230, 453-119, 453-209 y 401-201) se dispone de una copia en CD.

Para iniciar la instalación, basta ubicar entre los instaladores el programa setup e iniciarlo.



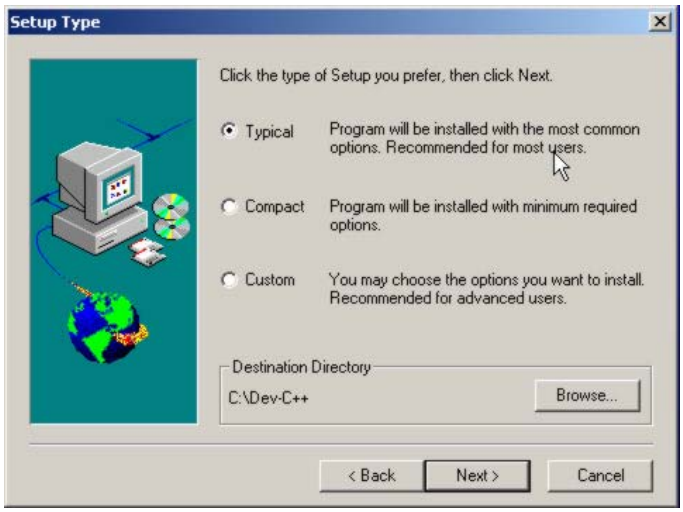
En la ventana de instalación, se siguen instrucciones muy sencillas,

1. Haga click en **YES**, para aceptar los términos de la licencia.



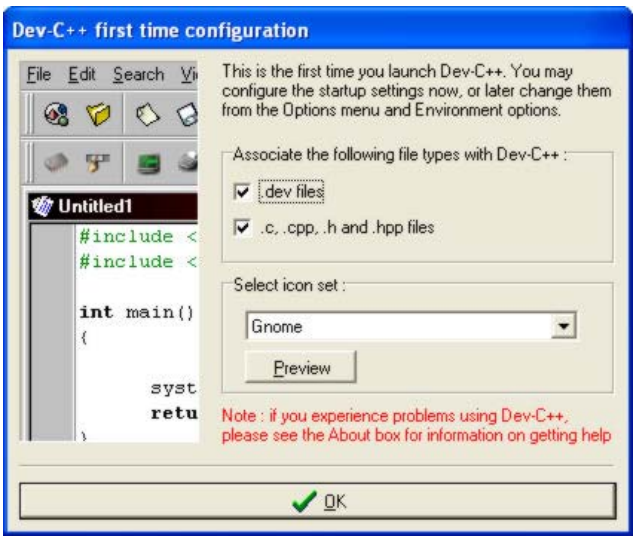
2. Tipo de instalación

Es recomendada, la instalación “Custom” (el usuario selecciona los módulos que desea), ya que se instalarán los recursos requeridos para el trabajo. Se sugiere seleccionar todos los módulos para no tener inconvenientes, luego hacer clic en “Next >” para copiar los archivos.



Al terminar, la instalación presenta la posibilidad, de abrir un archivo que describe el programa, o de iniciar el programa por primera vez. Usted puede seleccionar estas opciones de acuerdo a su preferencia.

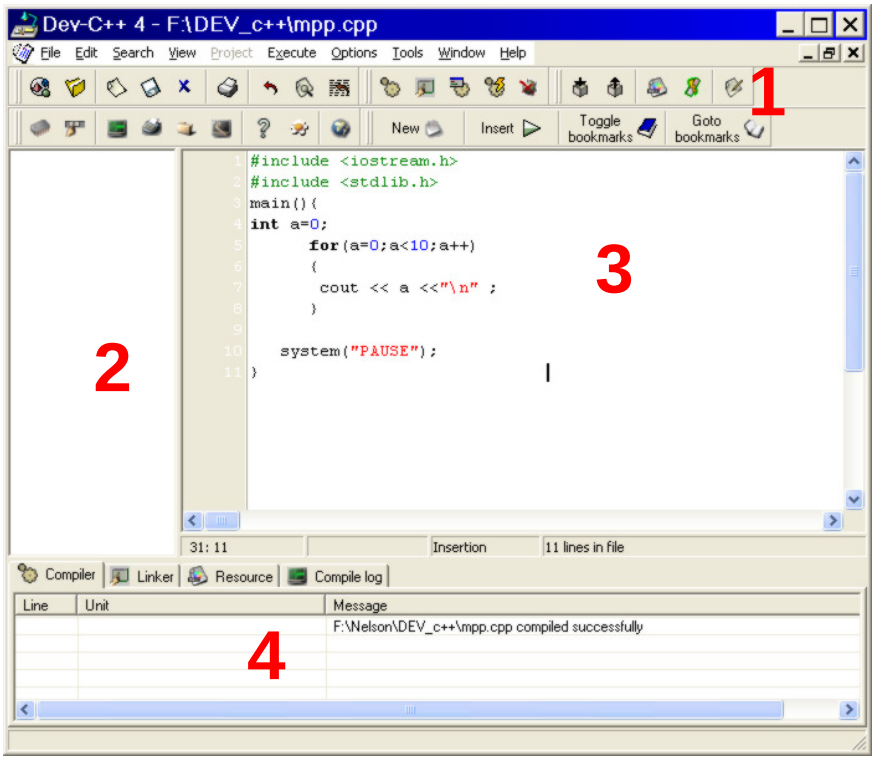
La primera vez que es ejecutado, el programa pedirá la asociación de los archivos propios del lenguaje C al entorno de desarrollo (muy conveniente), y además la selección de un estilo visual para los iconos de entorno de trabajo. Es conveniente hacer una selección de acuerdo a las capacidades del equipo. Ya que no todas las colecciones de iconos se verán adecuadamente en pantallas de baja resolución (frecuente en win 95 o 98).



3 Entorno de trabajo

Se pueden identificar 4 áreas principales:

1. Menú y barras de herramientas
2. Explorador de proyectos
3. Área de trabajo y edición.
4. Resultado de la compilación.



Menú y barras de herramientas

Aquí tenemos los menús con los típicos comandos de Windows (abrir, guardar, copiar y pegar…) También tenemos una serie de iconos en las barras de herramientas que no son más que una parte de las opciones que tenemos en los menús, se puede dejar el ratón encima de un icono durante unos segundos y aparecerá una ayuda emergente. Explicativa de lo que se ejecuta cuando se hace clic sobre la figura. En el numeral 4 se hará una pequeña descripción de cada una de las opciones del menú.

Explorador de proyectos y clases e información de depuración.

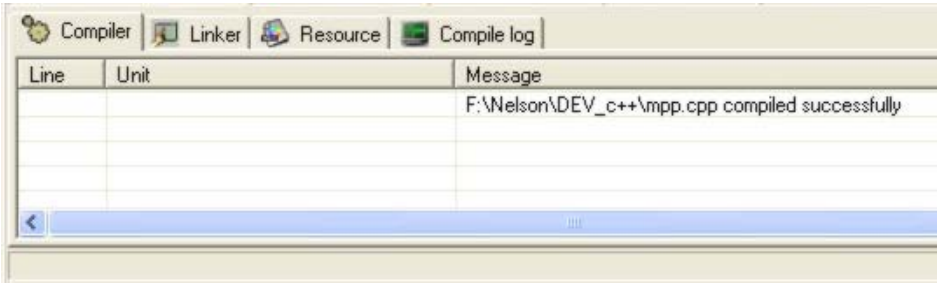
Dependiendo de la pestaña que seleccione en esta área tendrá acceso a:

a) Explorador de proyectos, que muestra los archivos por los que está formado el proyecto -y por tanto su aplicación- bien sean de código, de encabezados, o de recursos.
b) Explorador de clases, una de las funciones más útiles, más adelante se detallará cada una de las estructuras/clases definidas en los archivos del proyecto, así como los métodos y datos que forman parte de la estructura/clase, incluyendo sus argumentos y su tipo. También se verá una lista de las funciones globales que tenemos en el proyecto, también con sus argumentos. Pulsando doble clic en un método, función o clase, se irá directamente al archivo y línea donde se ha definido.
c) Información de depuración, aquí podremos definir las variables que queramos cuando estemos depurando un programa.

Área de edición.

Aquí aparecerán los Archivo de código que se abran. Se Pueden tener abierto más de un Archivo a la vez, y seleccionarlo por medio del menú Window.

Resultados de la compilación y controles de depuración.



En ésta serie de pestañas se encuentra información acerca del proceso de compilación. Cuando se selecciona una pestaña se expandirá para mostrarnos los resultados.

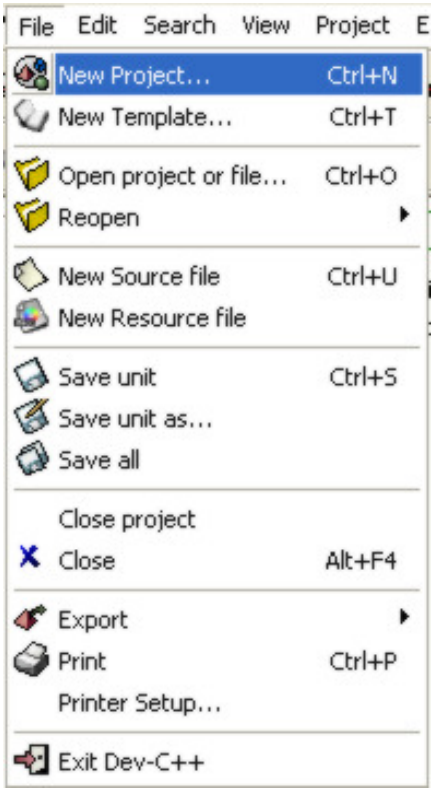
En la pestaña “compiler” (compilador) se ven los errores y advertencias que ha generado la compilación de nuestro código (si los hubiera), pulsando doble clic en uno de ellos se remite directamente a la línea que provocó dicho error o advertencia. También se generan avisos.

También existen otras pestañas, con propósitos más específicos, “linker”, informa acerca de la correcta referencia de las librerías con el código que hemos creado. “resource”, indica posibles advertencias acerca de otros recursos invocados en las líneas de código de nuestro programa, y “Compile Log”, informa acerca de diferentes mensajes que produzca la herramienta de compilación. En muy raras ocasiones hay mensajes el las pestañas “linker”, “resource”, sin embargo, cuando aparecen estos son muy importantes.

4 Descripción de las opciones del Menú

4.1 Menú File

Permite realizar operaciones con ficheros y salir del programa:



Nuevo proyecto

Nueva plantilla

Abrir proyecto existente o Archivo

Abrir archivos recientes

Fichero nuevo

Nuevo recurso

Guardar

Guardar con otro nombre

Guardar todos los Archivos abiertos

Cerrar proyecto

Cerrar ventana actual

Exportar a HTML o RTF (Word)

Imprimir

Setup impresora

Salir del Programa

4.2 Menú Edit

Acciones que se pueden realizar para las tareas de edición de texto.

Deshacer

Rehacer

Cortar

Copiar

Pegar

Insertar

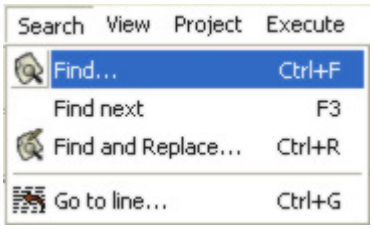
Insertar una marca

Ir a una marca

Seleccionar todo el contenido de la ventana

4.3 Menú Search

Para hacer búsquedas en el texto



Buscar

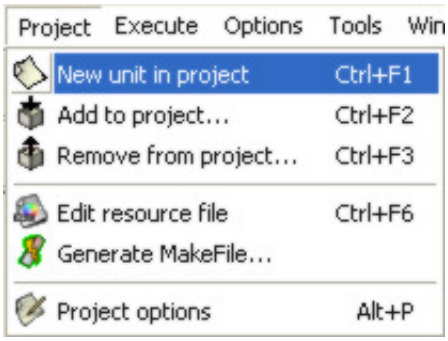
Buscar la siguiente aparición

Buscar y reemplazar

Ir a la línea número. . .

4.4 Menú Project

En este menú se pueden gestionar los elementos que pertenecen a un proyecto



Nuevo proyecto

Nuevo elemento al proyecto (adicionar archivo)

Eliminar archivo del proyecto

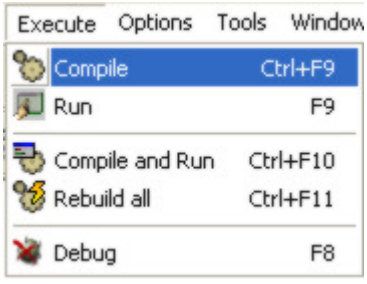
Abrir un archivo para editar

Generar un archivo “make”

Opciones del proyecto.

4.5 Menú Execute

En este menú encuentra las herramientas necesarias para generar los archivos ejecutables de un proyecto. Lo correspondiente al uso de estas opciones se aclarará en el numeral 7.



Compilar (generar un programa)

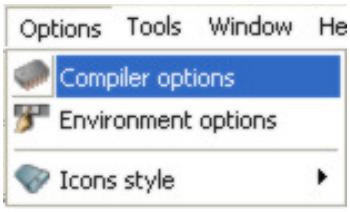
Correr el programa

Compilar y correr

Reconstruir todo

Depurar

4.6 Menú Options

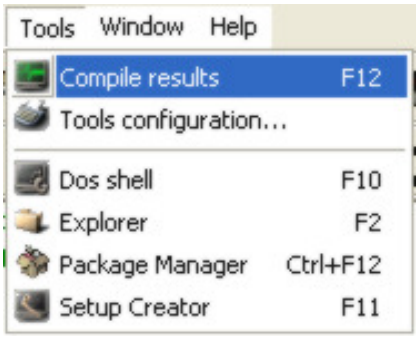


Opciones del compilador

Opciones de ambiente

Estilo visual de iconos.

4.7 Menú Tools



Muestra la ventana de resultado de compilación

Herramientas para configurar Dev C++

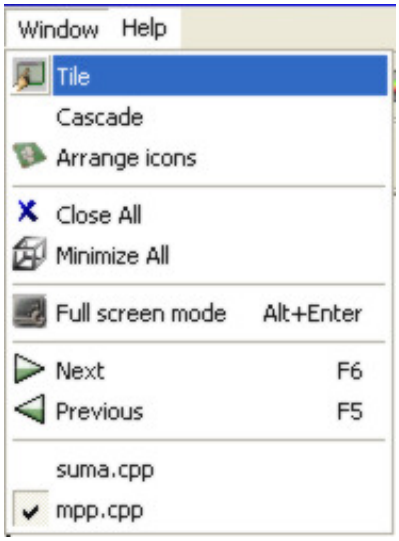
Muestra la línea de comandos del DOS

Activa el Explorador de Windows

Control de grupos de archivos

Creador de archivos de instalación.

4.8 Menú Windows



Ordena las ventanas por mosaico

Ordena las ventanas en cascada

--

Cerrar toas las ventanas

Minimizar todas las ventanas

Modo pantalla completa

Siguiente ventana

Ventana anterior

Muestra las ventanas activas

5 Iniciar un proyecto

Antes de construir un programa, es importante definir el nombre del proyecto, es aconsejable que este no supere los 8 caracteres, y que este describa o de una idea de aquello que pretende hacer.

En Dev-C++, hay diferencias entre utilizar: “New project” (iniciar un proyecto), o “New Source file” (iniciar archivo con código fuente). El primero hace referencia a la creación de uno o varios archivos que hacen parte de uno o varios programas. El segundo, inicia un solo archivo de código C++, incluyendo en éste un código básico, como plantilla de programa.

En su totalidad, el curso de programación, está diseñado para ser trabajado con “ New Source file” (iniciar archivo con código fuente), por lo tanto, una vez iniciado el programa, se selecciona en el menú, “File” la opción “ New Source file” 

o de otra forma, basta oprimir simultáneamente las teclas Ctrl y u. Hecho esto, debe aparecer en el área de edición, el siguiente texto:

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

int main()
{

	system("PAUSE");
	return 0;
}
```

El cual es útil para iniciar a trabajar, pues proporciona el uso de unas librerías básicas (`#include <iostream.h>` y `#include <stdlib.h>`), establece el inicio y final de programa (**`int`** `main(){},` da la opción de que al terminar el programa permita visualizar las salidas (system (“PAUSE”);) y devolver el control al sistema retornando 0 (return 0). Se debe incluir después del corchete `{` y antes de system (“PAUSE”); el código del programa que se desea construir.

6 Colores en la escritura de código

Una de las principales facilidades que un programa para la edición de código ofrece, es el presentar con diferentes colores aquellos códigos que tienen una connotación distinta en el código fuente, así por ejemplo, es posible identificar rápidamente la escritura de una palabra reservada en C++, o establecer si un comentario es tenido en cuenta o no al momento de compilar. La

asignación de colores, es automática, y se describirá brevemente a continuación.
En Dev c++, se utilizan las siguientes estructuras de código:

Directivas de Preprocesador: en DEV-C++, estas líneas de código son de color verde. Se distinguen por ser líneas de código que inician con el carácter #. no hace parte de la estructura lógica del programa, pero por medio de estas, es posible indicar al compilador que considere las líneas de código indicadas, para efectuar diversas tareas. Por ejemplo, la siguiente línea invoca las funciones almacenadas en el archivo stdlib.h,

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

Comentarios: se visualizan en color azul oscuro. Un comentario, es una frase que no es tenida en cuenta al momento de compilar el programa, es decir es una frase que aclara lo que se está realizando, como un mensaje para quien revisa el código del programa, pero que no es ejecutada, y se puede utilizar en cualquier parte del código. Se inicia con los caracteres "/*" y se termina con los caracteres "*/", como ejemplo:

```
/* ********************************
 Declaración de variables
******************************** */
```

Cadenas de caracteres: se presentan en rojo. Son textos, que tienen que ser manipuladas como variables o mensajes, y por lo tanto tienen una presentación especial. Como ejemplo:

```
cout << "Este mensaje aparecerá en pantalla";
```

Números: se ven en azul claro. Los valores numéricos, usados por su valor, son presentados en color azul. Como ejemplos:

```
int a[100][100],SumaPares=0, SumaImpares=0;
```

Palabras Claves: Se presentan en color negro resaltado. Son palabras usadas por el lenguaje C++, y que tienen un significado exacto para la ejecución de los programas, presentamos aquí varios ejemplos:

```
void main()
if (i <=0 || j <=0)
```

```
for (m=0; m < j; m++)
```

Otras expresiones como los operadores matemáticos y lógicos (Symbol) o los identificadores de función, aparecerán como texto normal en color negro sin resaltar.

7 Compilación y generación de programas

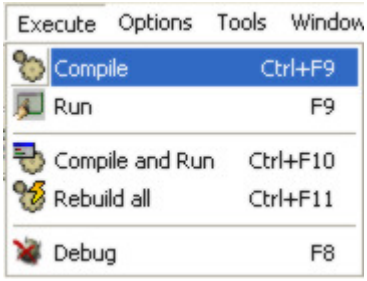
La compilación es el proceso mediante el cual el programa que tenemos en lenguaje de programación se traduce a lenguaje de máquina (ceros y unos) para poder ser ejecutado por el computador. La compilación genera un archivo ejecutable con extensión **.**exe.

El siguiente programa, que escribe los números de 1 a 10, está escrito en lenguaje de programación C++

```
#include <iostream.h>
#include <stdlib.h>
main(){
int a=0;
      for(a=0;a<10;a++)
      {
       cout << a <<"\n" ;
      }

   system("PAUSE");
}
```

Para convertir este código o cualquiera otro, en un programa ejecutable, se utilizan las herramientas de ejecución del programa: **"menú execute"**



Compilar (generar un programa)

Correr el programa

Compilar y correr

Reconstruir todo

Depurar

El proceso de compilación se hace por medio de la opción "Compile". Utilizando la opción "Run", se ejecuta la última compilación efectuada al código.

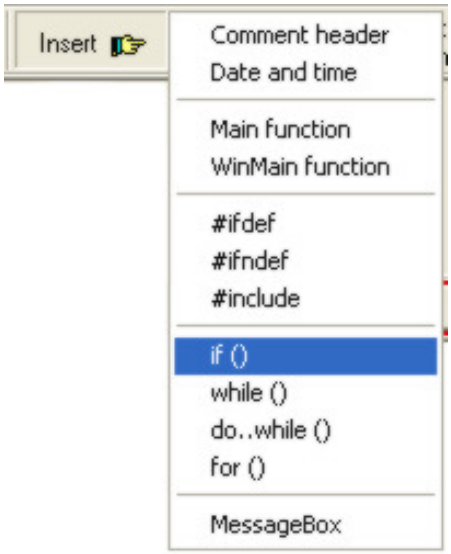
La opción "Compile and Run", hace consecutivamente las acciones de compilar y de correr el programa.

La opción "Rebuild all" sirve para volver a compilar todos los archivos de un proyecto, se usa para compilar más de un archivo, y tiene el mismo efecto de la opción "Compile".

8 Otras facilidades

Es posible insertar fracciones de código de frecuente uso, por medio de la opción INSERT del "Menu Edit", pues coloca en el editor estructuras completas del lenguaje, impidiendo que se omitan detalles que impiden la buena ejecución del programa.

Por Ejemplo, es recomendable iniciar a trabajar insertando "Comment header" que coloca en el punto donde se tenga el cursor sobre el editor, el siguiente código:



Por Ejemplo, es recomendable iniciar a trabajar insertando "Comment header" que presenta el siguiente código:

```
/*
   Name:
   Author:
   Description:
   Date:
   Copyright:
*/
```

Otro ejemplo, el resultado de insertar “for ()”, da como resultado:

```
for()
{
}
```

Se debe tener mucho cuidado, ya que como es posible ver, no se insertan todos los elementos básicos de la sentencia for, como por ejemplo el par de puntos y comas (;) necesarios para dividir los paramentos de entrada a esa sentencia.

9 Errores comunes en programación y como evitarlos con Dev-C++

9.1 Olvidar terminar un comentario con */

Incorrecto	Correcto

`#include <iostream.h>`
`#include <stdlib.h>`
 	
`int main()`
`{`
`/* inicio del programa`
`system("PAUSE");`
`return 0;`
`}`
	`#include <iostream.h>`
	`#include <stdlib.h>`
	
	`int main()`
	`{`
	`/* inicio del programa */`
	`system("PAUSE");`
	`return 0;`
	`}`

Si cuando escribe, nota que el texto no adquiere los formatos adecuados, muy posiblemente olvidó cerrar el comentario con los caracteres */	

9.2 Uso de letras mayúsculas cuando no es útil

Incorrecto	Correcto

`#include <iostream.h>`
`#include <stdlib.h>`
 	
`int Main()`
`{`
`/* inicio del programa */`
`system("PAUSE");`
`return 0;`
`}`
	`#include <iostream.h>`
	`#include <stdlib.h>`
	
	`int main()`
	`{`
	`/* inicio del programa */`
	`system("PAUSE");`
	`return 0;`
	`}`

No es evidente a simple vista, si se ignora el sentido de la instrucción, pero al momento de compilar (F9) muestra un resultado que indica la falta de una función o instrucción. El	

siguiere es el mensaje es mostrado en la ventana “Compiler and linker output” :

E:\DEV-C_~1\Lib\\libmingw32.a(main.o)(.text+0x8e): undefined reference to `WinMain@16'

9.3 Olvidar las dobles comillas de un texto o cadena de control.

Incorrecto	Correcto

```
1  #include <iostream.h>
2  #include <stdlib.h>
3  int main()
4  {
5  /* inicio del programa   */
6  cout<<Hola mundo \n;
7  system("PAUSE");
8  return 0;
9  }
```

```
1  #include <iostream.h>
2  #include <stdlib.h>
3  int main()
4  {
5  /* inicio del programa   */
6  cout<<"Hola mundo \n";
7  system("PAUSE");
8  return 0;
9  }
```

Este problema ocasiona múltiples errores al momento de crear un ejecutable.

El siguiere es el mensaje es mostrado en la ventana “Compiler and linker output” :

(nota, en este ejemplo el nombre del archivo es err3.cpp)

1. c:\err\err3.cpp: In function `int main()':
2. c:\err\err3.cpp:**7**: stray '\' in program
3. c:\err\err3.cpp:**7**: `Hola' undeclared (first use this function)
4. c:\err\err3.cpp:**7**: (Each undeclared identifier is reported only once for each function it appears in.)
5. c:\err\err3.cpp:**7**: parse error before `;'

Identifica la función en donde está el error (mensaje 1)

muestran la palabra ´Hola´ como algo no declarado (desconocido para el computador) (mensaje 3)

El número 7 indica la línea del código donde está el error.

9.4 Olvidar el signo ; después de cada instrucción.

Incorrecto

```
1
2  #include <iostream.h>
3  #include <stdlib.h>
4  int main()
5  {
6  /* inicio del programa    */
7  cout<<"Hola mundo \n"
8  system("PAUSE");
9  return 0;
10 }
```

Correcto

```
1
2  #include <iostream.h>
3  #include <stdlib.h>
4  int main()
5  {
6  /* inicio del programa    */
7  cout<<"Hola mundo \n";
8  system("PAUSE");
9  return 0;
10 }
```

Este inconveniente no indica en cual línea de código está el error, ya que generalmente indica la siguiente línea.
El siguiere es el mensaje es mostrado en la ventana “Compiler and linker output” :

(Nota, en este ejemplo el nombre del archivo es ejemp_4.cpp)

1. c:\err\ejemp_4.cpp: In function `int main()':
2. **c:\err\ejemp_4.cpp:8: parse error before `('**

En este ejemplo, hace falta terminar la instrucción 7 con **;** sin embargo la segunda línea del mensaje de salida indica que está en la línea numero 8.

9.5 Cambiar el signo ; por , después de alguna sentencia

Incorrecto

```
1
2  #include <iostream.h>
3  #include <stdlib.h>
4  int main()
5  {
6  /* inicio del programa    */
7  cout<<"Hola mundo \n";
8  system("PAUSE"),
9  return 0;
10 }
```

Correcto

```
1
2  #include <iostream.h>
3  #include <stdlib.h>
4  int main()
5  {
6  /* inicio del programa    */
7  cout<<"Hola mundo \n";
8  system("PAUSE");
9  return 0;
10 }
```

Este inconveniente no indica en cual línea de código está el error, ya que generalmente indica la siguiente línea.
El siguiere es el mensaje es mostrado en la ventana “Compiler and linker output” :

(Nota, en este ejemplo el nombre del archivo es ejemp_5.cpp)

```
1  c:\err\ejemp_5.cpp: In function `int main()':
2  c:\err\ejemp_5.cpp:9: parse error before `return'
```

En este ejemplo, hace falta terminar la instrucción 8 con **;** sin embargo la segunda línea del mensaje de salida indica que está en la línea numero 9.

9.6 Olvidar colocar llaves en una sentencia compuesta (aplica para las sentencias for, if, do, switch)

Incorrecto

```
1   #include <iostream.h>
2   #include <stdlib.h>
3   int main()
4   {
5   int a=0;
6   int b=0;
7   for (a = 0 ; a < 100 ; a++ )
8   {
9   b= a*a;
10  cout<<b<<" = "<<a<<"*"<< a<<"\n";
11
12  system("PAUSE");
13  return 0;
14  }
15
```

Correcto

```
1   #include <iostream.h>
2   #include <stdlib.h>
3   int main()
4   {
5   int a=0;
6   int b=0;
7   for (a = 0 ; a < 100 ; a++ )
8   {
9   b= a*a;
10  cout<<b<<" = "<<a<<"*"<< a<<"\n";
11  }
12  system("PAUSE");
13  return 0;
14  }
15
```

Este inconveniente no indica en cual línea de código está el error. Esto es debido a que realiza la comprobación completa de paréntesis que abren y cierran, solo el programador deberá determinar donde está el error.

El siguiere es el mensaje es mostrado en la ventana “Compiler and linker output” :

(Nota, en este ejemplo el nombre del archivo es ejemp_6.cpp)

```
1   c:\err\ejemp_6.cpp: In function `int main()':
2   c:\err\ejemp_6.cpp:15: parse error at end of input
```

En este ejemplo, hace falta cerrar llaves en la línea 11, el error es indicado después de toda la comprobación de llaves en la línea 15

9.7 Las palabras reservadas son:

auto
break
case
char
const
continue
default
do double
else
enum
extern
float
for
goto
if int
long
register
return
short
signed
sizeof
static
struct
switch
typedef
union
unsigned
void
volatile
while

Estas palabras no admiten modificaciones, es necesario escribirlas tal como son, sin mayúsculas, si no aparecen resaltadas en negrita resaltado en el área de edición, muy seguramente están mal escritas o fuera de contexto, y por lo tanto es necesario revisar el código.

Incorrecto	Correcto

```
1   #include <iostream.h>
2   #include <stdlib.h>
3   int main()
4   {
5   int a=0;
6   int b=0;
7   For (a = 0 ; a < 100 ; a++ )
8    {
9      b= a*a;
10    cout<<b<<" = "<< a <<"*"<<a<<"\n";
11   }
```

```
1   #include <iostream.h>
2   #include <stdlib.h>
3   int main()
4   {
5   int a=0;
6   int b=0;
7   for (a = 0 ; a < 100 ; a++ )
8    {
9      b= a*a;
10    cout<<b<<" = "<<a<<"*"<<a<<"\n";
11   }
```

```
12  system("PAUSE");
13  return 0;
14  }
```

```
12  system("PAUSE");
13  return 0;
14  }
```

Una palabra reservada denota una instrucción, si esta no es clara o no se encuentra, se pierde el sentido de las expresiones construidas en todas las líneas de código.
El siguiere es el mensaje es mostrado en la ventana “Compiler and linker output” :
(Nota, en este ejemplo el nombre del archivo es ejemp_7.cpp)

```
1  c:\err\ejemp_7.cpp: In function `int main()':
2  c:\err\ejemp_7.cpp:7: parse error before `;'
3  c:\err\ejemp_7.cpp:7: parse error before `)'
4  c:\err\ejemp_7.cpp: At top level:
5  c:\err\ejemp_7.cpp:12: ANSI C++ forbids declaration `system' with no type
6  c:\err\ejemp_7.cpp:12: `int system' redeclared as different kind of symbol
7  E:\DEV-C_~1\Include\stdlib.h:283: previous declaration of `int system(const char *)'
8  c:\err\ejemp_7.cpp:12: initialization to `int' from `const char *' lacks a cast
9  c:\err\ejemp_7.cpp:13: parse error before `return'
```

En este ejemplo, la sentencia for de la línea 7, tiene una letra en mayúsculas, el editor de texto no la detecta como una palabra reservada de C++, y por lo tanto, la sentencias propias de esta orden (`a = 0 ; a < 100 ; a++ `), pierden sentido, y ocurre un error general en la aplicación, es aconsejable en estos casos, revisar cada una de las líneas indicadas en la ventana de resultados de compilación para verificar su sintaxis.

9.8 Olvidar definir una variable.

Incorrecto

```
1   #include <iostream.h>
2   #include <stdlib.h>
3   int main()
4   {
5   int a=0;
6
7   for (a = 0 ; a < 100 ; a++ )
8   {
9   b= a*a;
10  cout <<b<<" = "<< a << "*" <<a<<"\n";
11  }
12  system("PAUSE");
13  return 0;
14  }
```

Correcto

```
1   #include <iostream.h>
2   #include <stdlib.h>
3   int main()
4   {
5   int a=0;
6   int b=0;
7   for (a = 0 ; a < 100 ; a++ )
8   {
9   b= a*a;
10  cout <<b<<" = "<< a << "*" <<a<<"\n";
11  }
12  system("PAUSE");
13  return 0;
14  }
```

El siguiere es el mensaje es mostrado en la ventana “Compiler and linker output” :

(Nota, en este ejemplo el nombre del archivo es ejemp_8.cpp)

```
1   c:\err\ejemp_8.cpp:9: `b' undeclared (first use this function)
2   c:\err\ejemp_8.cpp:9: (Each undeclared identifier is reported only once
3   c:\err\ejemp_8.cpp:9: for each function it appears in.)
```

Hace falta definir la variable b, en la línea 6, este error es detectado fácilmente por el

compilador como lo indica el primer mensaje del compilador, sin embargo, en la línea 9 es llamada esta variable por primera vez, por lo tanto es hay donde se muestra el error.

9.9 Usar “,” en vez de “;” dentro de una sentencia for

Incorrecto

```
1   #include <iostream.h>
2   #include <stdlib.h>
3   int main()
4   {
5   int a=0;
6   int b=0;
7   for (a = 0 , a < 100 , a++ )
8   {
9   b= a*a;
10  cout <<b<<" = "<< a << "*" <<a<<"\n";
11  }
12  system("PAUSE");
13  return 0;
14  }
```

Correcto

```
1   #include <iostream.h>
2   #include <stdlib.h>
3   int main()
4   {
5   int a=0;
6   int b=0;
7   for (a = 0 ; a < 100 ; a++ )
8   {
9   b= a*a;
10  cout <<b<<" = "<< a << "*" <<a<<"\n";
11  }
12  system("PAUSE");
13  return 0;
14  }
```

El siguiere es el mensaje es mostrado en la ventana “Compiler and linker output” :

(Nota, en este ejemplo el nombre del archivo es ejemp_9.cpp)

```
c:\err\ejemp_9.cpp:7: parse error before `)'
c:\err\ejemp_9.cpp:11: parse error before `}'
```

El uso de “**,**“ en ves de “**;**“dentro de una sentencia for, genera error en 2 partes, en la línea donde se ha declarado la sentencia, y en la línea donde finaliza, no necesariamente es indicada la falta del signo “**;**”

9.10 Olvidar incluir una librería

Incorrecto

```
1   #include <iostream.h>
2   #include <stdlib.h>
3
4   int main()
5   {
6   float a=0;
7   float b=0;
8   for (a = 0 ; a < 100 ; a++ )
9   {
10  b = atan(a);
11  cout  <<b<<"  =  atan("<<  a  <<  ")"
    <<"\n";
12  }
13  system("PAUSE");
14  return 0;
15  }
```

Correcto

```
1   #include <iostream.h>
2   #include <stdlib.h>
3   #include <math.h>
4   int main()
5   {
6   float a=0;
7   float b=0;
8   for (a = 0 ; a < 100 ; a++ )
9   {
10  b = atan(a);
11  cout  <<b<<"  =  atan("<<  a  <<  ")"
    <<"\n";
12  }
13  system("PAUSE");
14  return 0;
15  }
```

El siguiere es el mensaje es mostrado en la ventana “Compiler and linker output” :

(Nota, en este ejemplo el nombre del archivo es ejemp_10.cpp)

c:\err\ejemp_10.cpp:10: implicit declaration of function `int atan(...)'

En la línea 10 la variable b está recibiendo el valor del arcotangente de a, pero arcotangente es una función definida dentro de la librería matemática, por lo tanto el error indicado por el compilador, es no poder resolver el significado de atan(...);

Una librería es un archivo externo, que se llama para incluir funciones que están definidas dentro de este, es común encontrar un conjunto estándar de librerías, estas son:

ALLOC.H	ASSERT.H	BCD.H	BIOS.H	COMPLEX.H
CONIO.H	CTYPE.H	DIR.H	DIRENT.H	DOS.H
ERRNO.H	FCNTL.H	FLOAT.H	FSTREAM.H	GENERIC.H
GRAPHICS.H	IO.H	IOMANIP.H	IOSTREAM.H	LIMITS.H
LOCALE.H	MALLOC.H	MATH.H	MEM.H	PROCESS.H
SETJMP.H	SHARE.H	SIGNAL.H	STDARG.H	STDDEF.H
STDIO.H	STDIOSTR.H	STDLIB.H	STREAM.H	STRING.H
STRSTREA.H	SYS\STAT.H	SYS\TIMEB.H	SYS\TYPES.H	TIME.H

En cada una de estas librerías encontrará diversas funciones útiles, que simplifican la programación, por ejemplo en MATH.H, encontrará muchas funciones matemáticas, como la expuesta en el ejemplo anterior, también es posible (y muy recomendable), escribir librerías personalizadas, definidas para cada necesidad, ya que esto facilita la utilización de líneas de código que pueden ser útiles en uno o más programas.