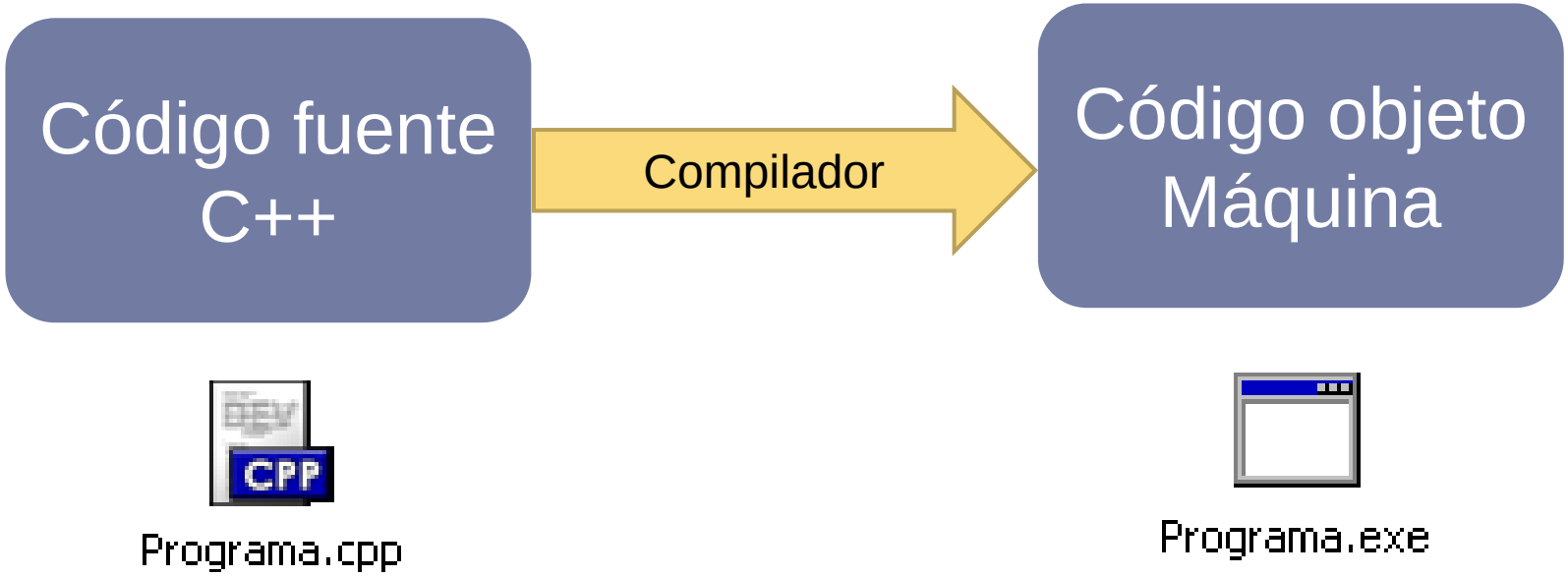
Introducción al laboratorio de Programación I

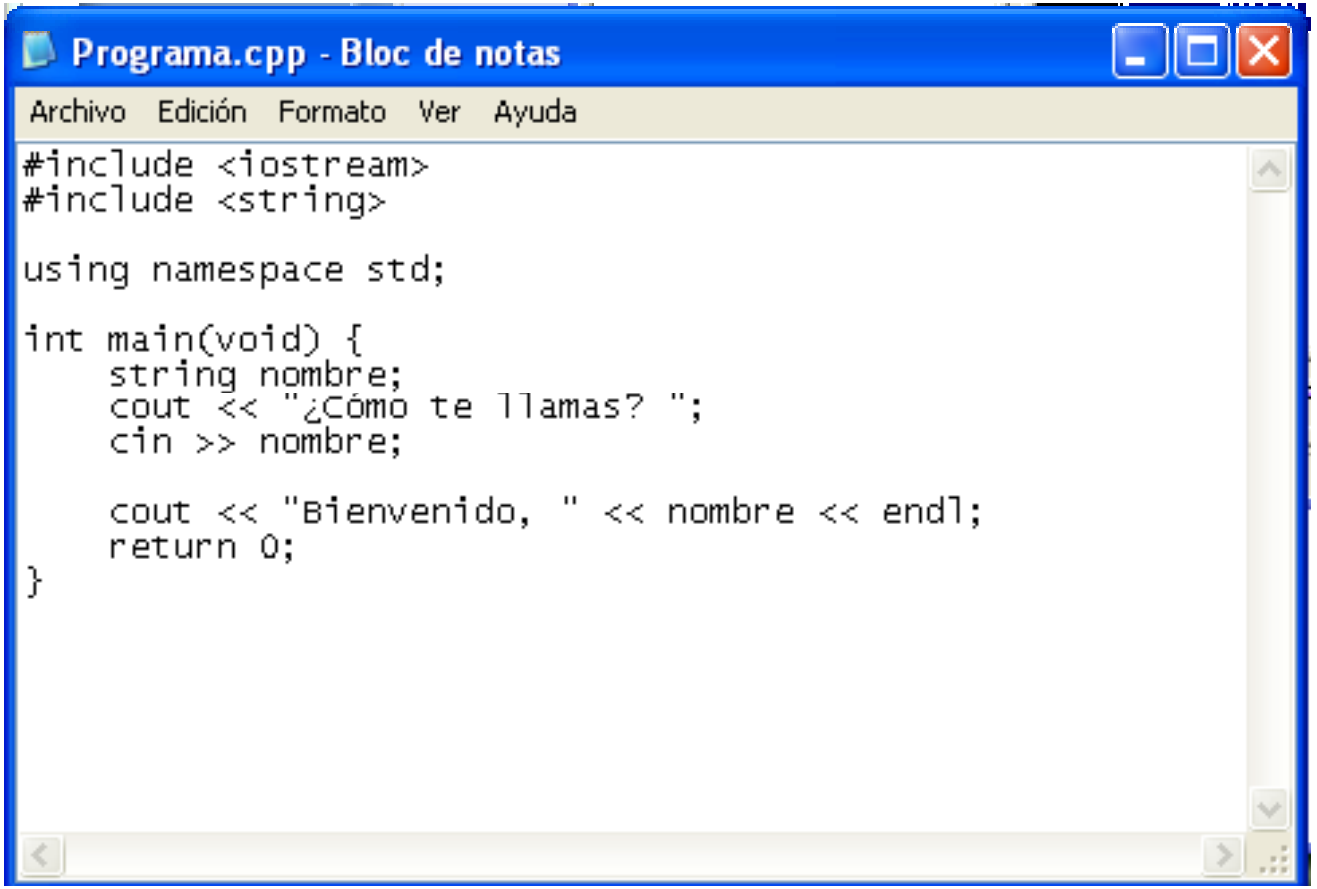
Manuel Montenegro
montenegro@fdi.ucm.es
Despacho B12

Compilación

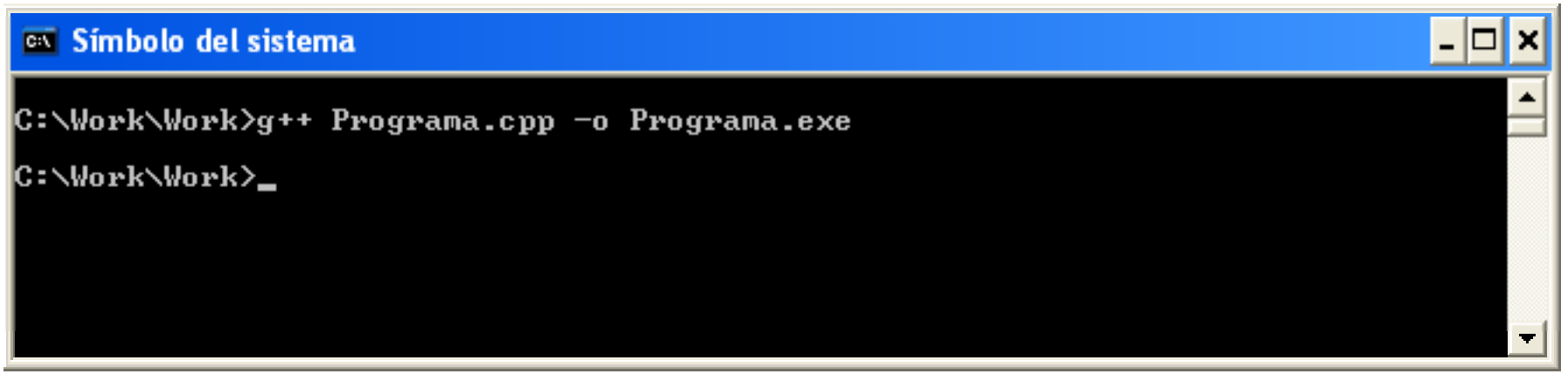
- Es el proceso de transformación de un programa escrito en un lenguaje de alto nivel (en nuestro caso C++) a un lenguaje inteligible por el ordenador (código máquina).

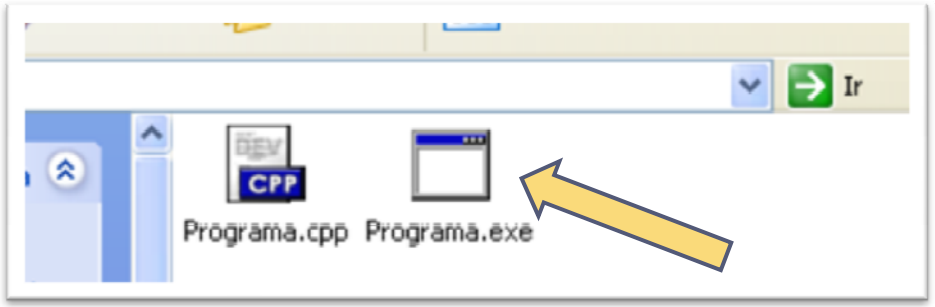


Paso I – Edición del programa

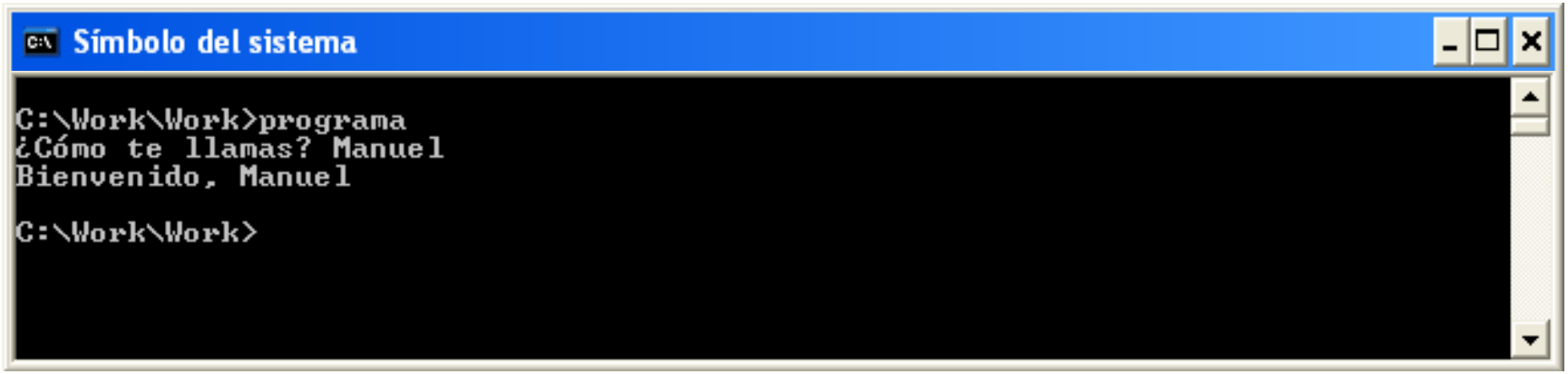


Paso II - Compilación





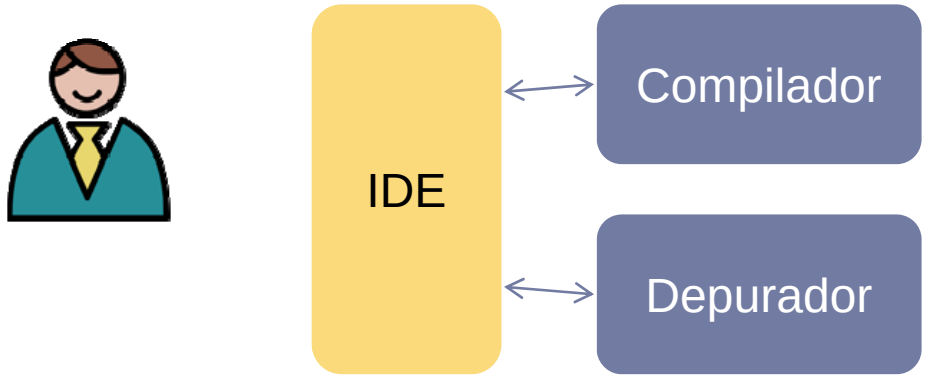
Paso III - Ejecución



- Procedimiento engorroso en programas grandes.

IDE

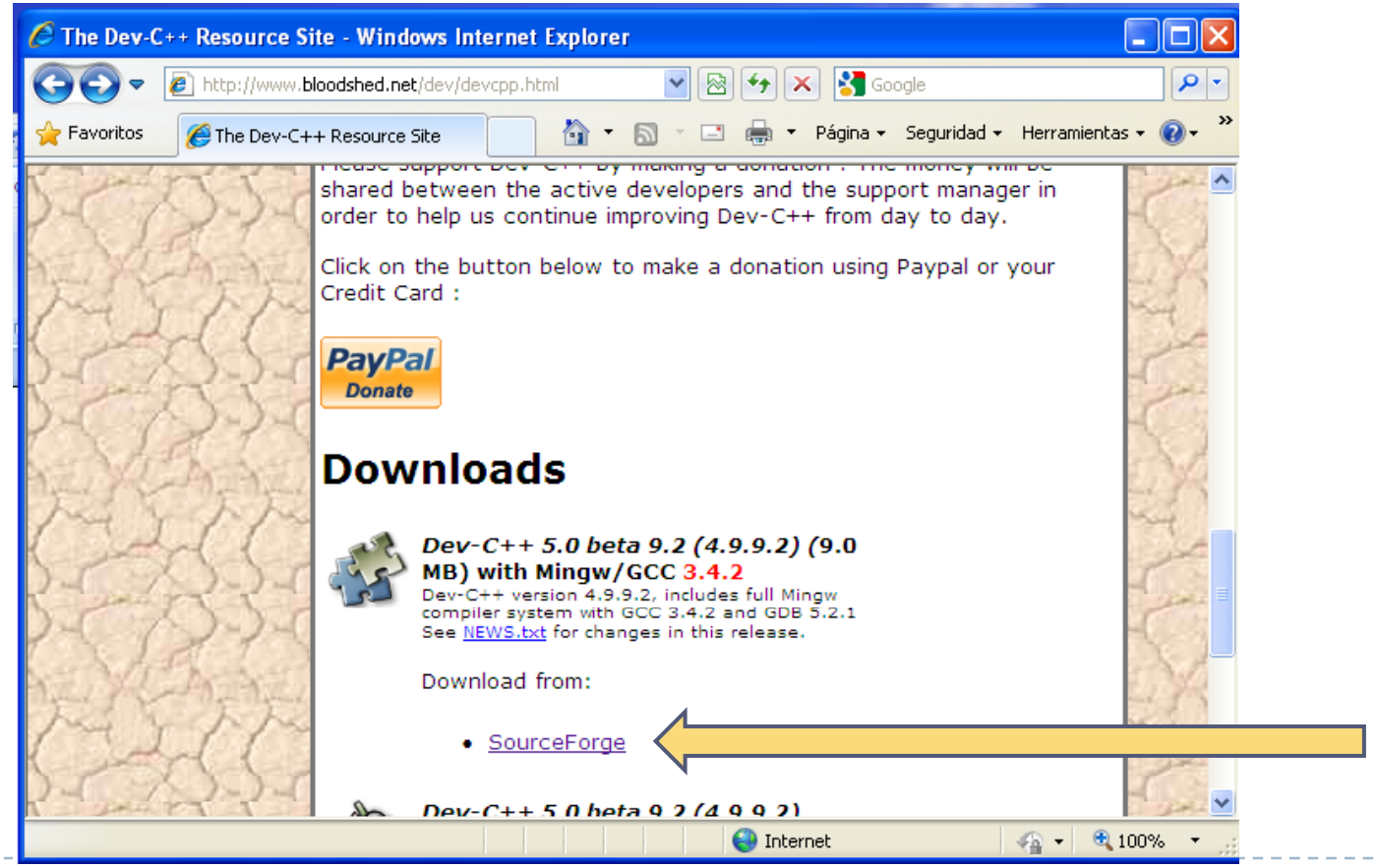
- Un entorno integrado de desarrollo (IDE) es un conjunto de herramientas para un programador.
- Proporciona varias utilidades de edición, así como una interfaz para el compilador.



- En el curso utilizaremos el entorno Dev-C++

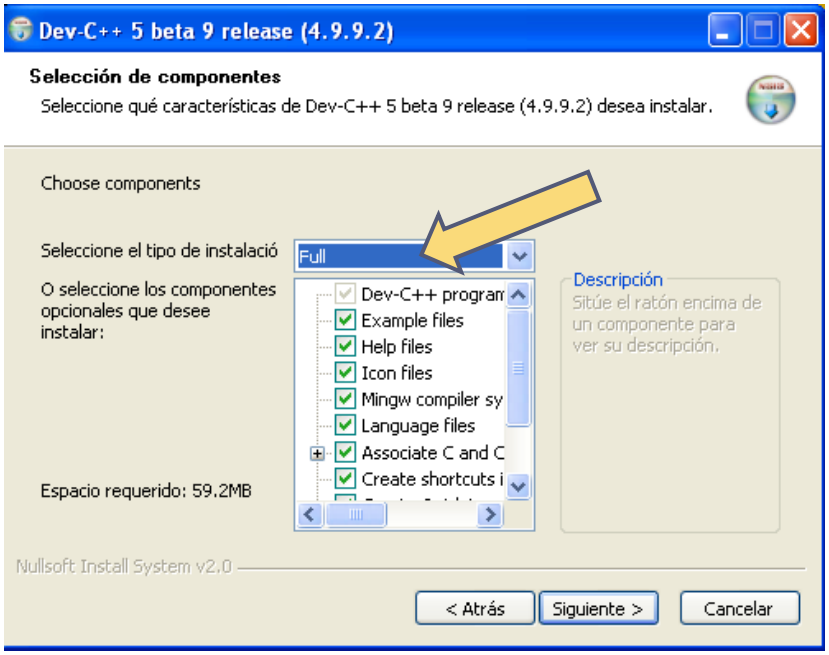
Entorno Dev-C++

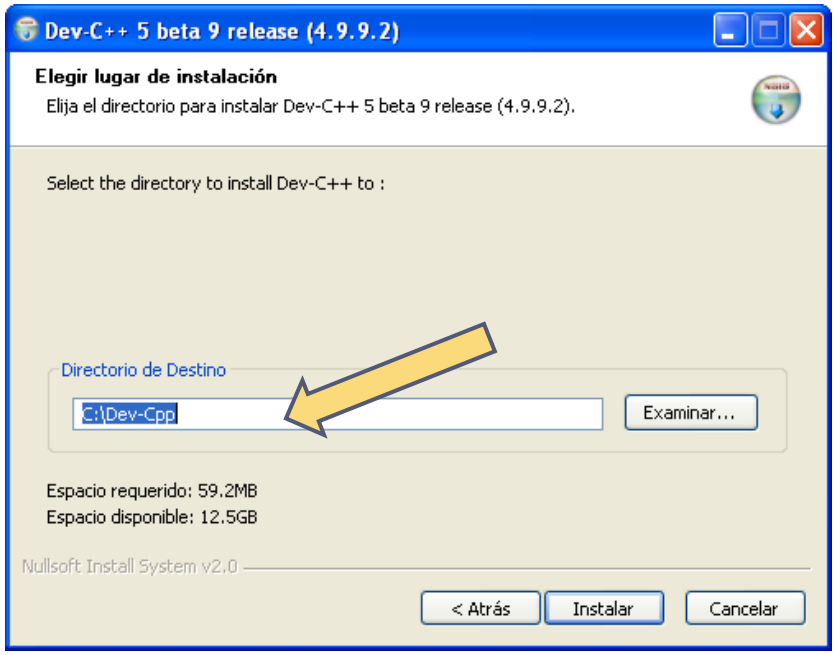
- Puede obtenerse en la dirección:
  http://www.bloodshed.net/dev/devcpp.html



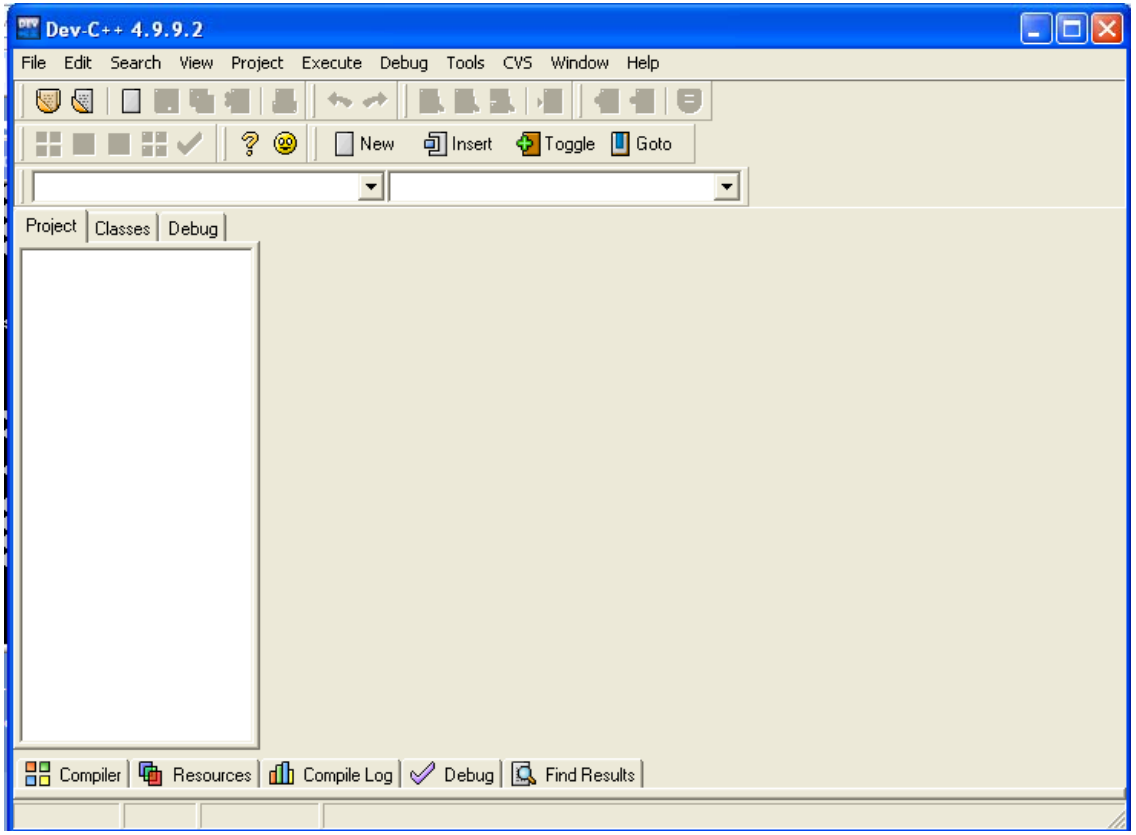
Instalación de Dev-C++

- Opciones en el programa de instalación:
  - Tipo de instalación: completa (Full)
  - Es muy recomendable instalarlo en la carpeta por defecto.





Ejecución de Dev-C++



- Para crear un programa:

  File → New → Source File

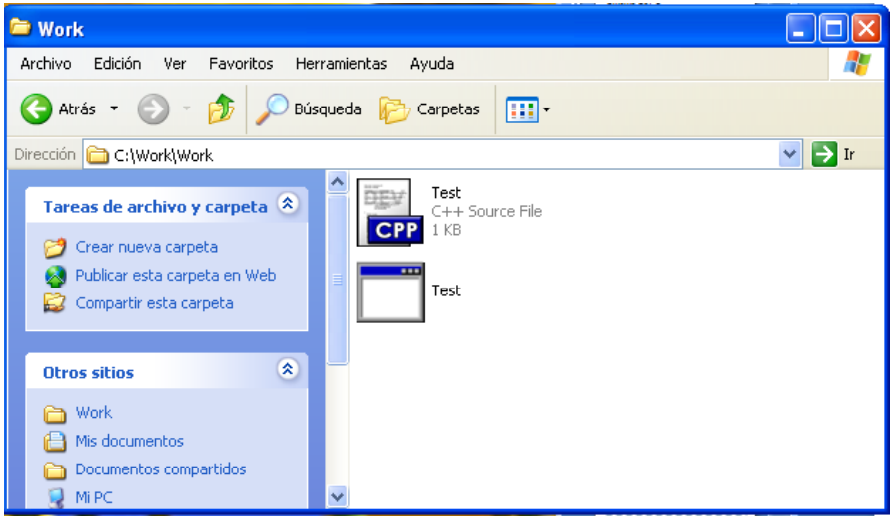
Programa “Hola, mundo”

```
#include <iostream>

using namespace std;

int main() {
    cout << "Hola, mundo" << endl;

    return 0;
}
```

Programa “Hola, mundo”

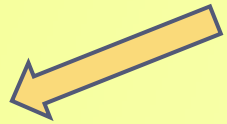
- Guardar el archivo:
  - File → Save *(Ctrl + S)*
  - Es necesario que el nombre de archivo tenga la extensión .cpp

- Compilar con:
  - Execute → Compile *(Ctrl + F9)*



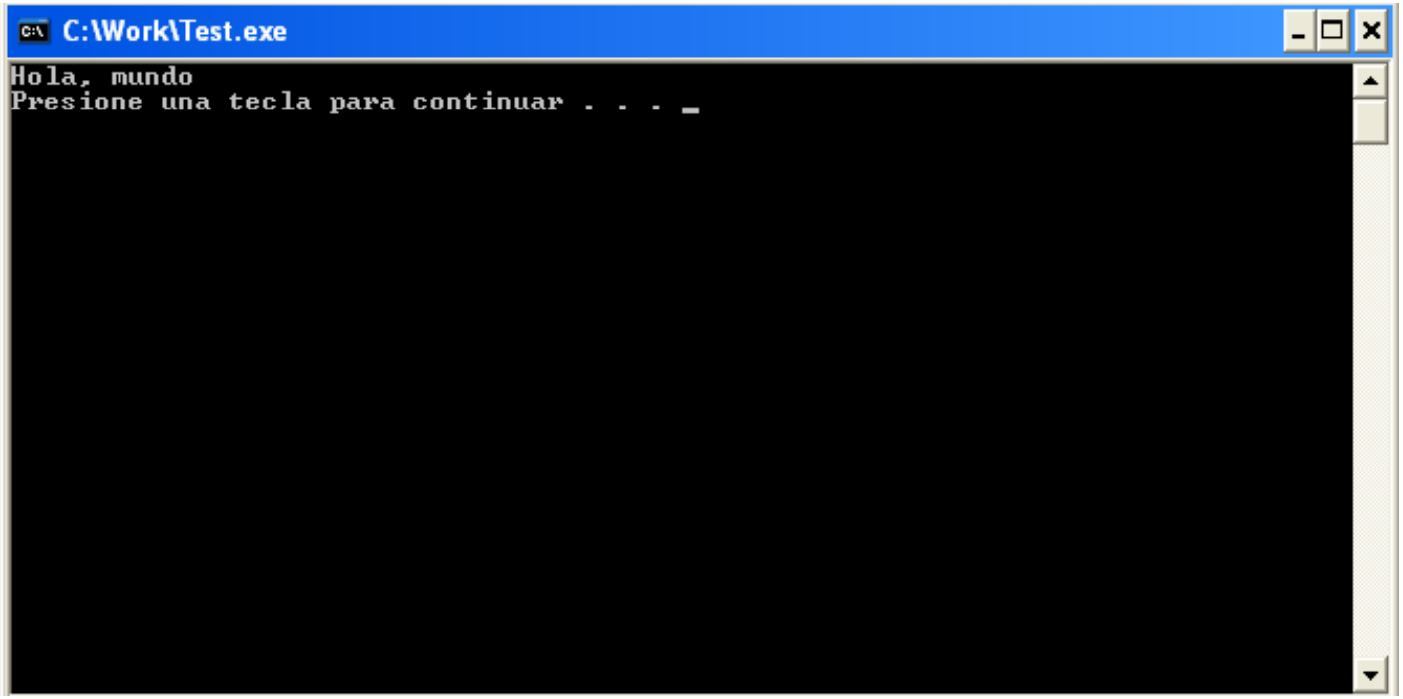
Programa “Hola, mundo” corregido

```
#include <iostream>

using namespace std;

int main() {
    cout << "Hola, mundo" << endl;

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



Ejecución del programa

- Podemos ejecutar el programa mediante:
  - Execute → Run *(Ctrl+F10)*
  - Execute → Compile & Run *(F9)*



Breve explicación del programa (I)

- `int main()`
  - Indica que el siguiente bloque de sentencias (delimitado por { ... }) es el contenido de la función principal del programa.
  - La ejecución de un programa en C++ siempre comienza con la ejecución de la función principal, independientemente de su situación en el programa.

- En nuestro ejemplo, la función main se compone de tres sentencias:

```
cout << "Hola, mundo" << endl;
system("pause");
return 0;
```

- Cada sentencia siempre va finalizada por el símbolo (;).

Breve explicación del programa (II)

- `cout << "…"`
  - `cout` es una variable que representa la salida estándar (en nuestro caso, la pantalla).
  - Esta sentencia envía el texto delimitado entre comillas dobles (“”) a la pantalla.
- Las distintas salidas pueden encadenarse:

```
cout << "Hola," << "mundo";
```

- Mediante `endl` indicamos que las salidas posteriores se impriman en la siguiente línea.

```
cout << "Linea 1" << endl << "Linea 2";
cout << endl << "Linea 3";
```

Variables

```
#include <iostream>

using namespace std;

int main() {
    int edad;

    cout << "Hola, mundo" << endl;
    edad = 34;
    cout << "Mi edad es" << edad << endl;

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

Declaración de variable

Asignación

Variables

- El resultado de ejecutar este programa es:

```
Hola, mundo
Mi edad es34
```

- Corrección:

```
cout << "Mi edad es " << edad << endl;
```

- ¿Qué ocurre si cambiamos esta sentencia por la siguiente?

```
cout << "Mi edad es " << "edad" << endl;
```

Directiva `#include`

- Toda identificador que se utilice ha de ser declarado previamente.
  - `int edad, dias;`
- ¿Dónde se encuentran declarados `cout` y `endl`?
- La directiva `#include <iostream>` indica que se incluya el fichero `iostream.h`, donde se encuentran las definiciones de estas dos variables, junto con otras definiciones relacionadas con la entrada y salida.
- El archivo `iostream.h` forma parte de la librería estándar de C++

Librerías estándar de C/C++

Nombre	Contenido
`iostream`	Entrada/Salida estándar
`fstream`	Entrada/Salida mediante ficheros
`cmath`	Funciones matemáticas: log, cos, tan,…
`ctime`	Funciones para obtener la hora y fecha actuales
`cctype`	Funciones relacionadas con caracteres
`string`	Manejo de cadenas de caracteres
…	…

http://www.cplusplus.com/reference/

Operaciones aritméticas

```
#include <iostream>

using namespace std;

int main() {
    int edad, dias;

    cout << "Hola, mundo" << endl;
    edad = 34;
    dias = edad * 365;
    cout << "Mi edad es" << edad << endl;
    cout << "Esto son " << dias << " dias" << endl;

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

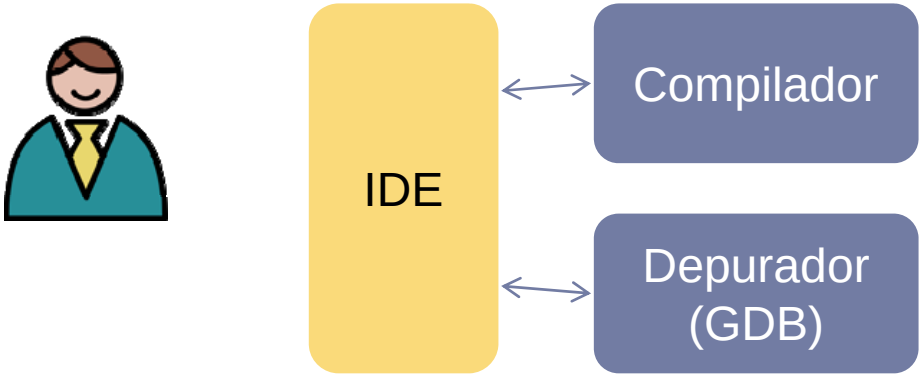
Op	Significado
+	Suma
-	Resta
*	Multiplicación
/	División
%	Módulo (resto de una división)

Tipos de errores

- Errores de compilación
  - Ejemplos: sintaxis incorrecta, uso de variables no declaradas, …
  - Se detectan antes de la ejecución del programa.
- Errores en tiempo de ejecución
  - Ejemplos: divisiones por cero, …
  - Aparece un error durante la ejecución del programa advirtiendo de tal circunstancia.
- Errores lógicos
  - El programa no funciona de acuerdo a lo esperado.
  - No se reflejan en ningún mensaje de error.

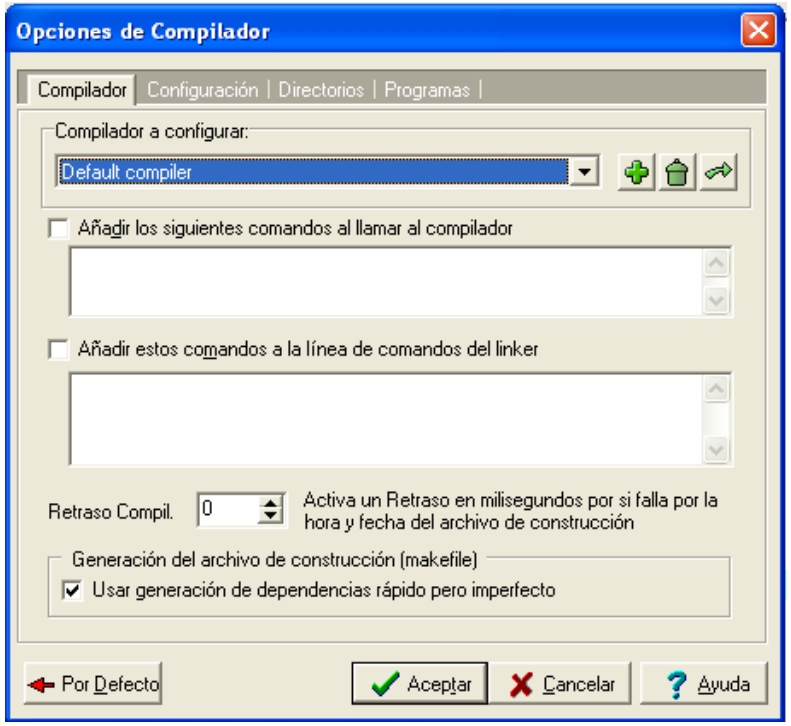
Depuración

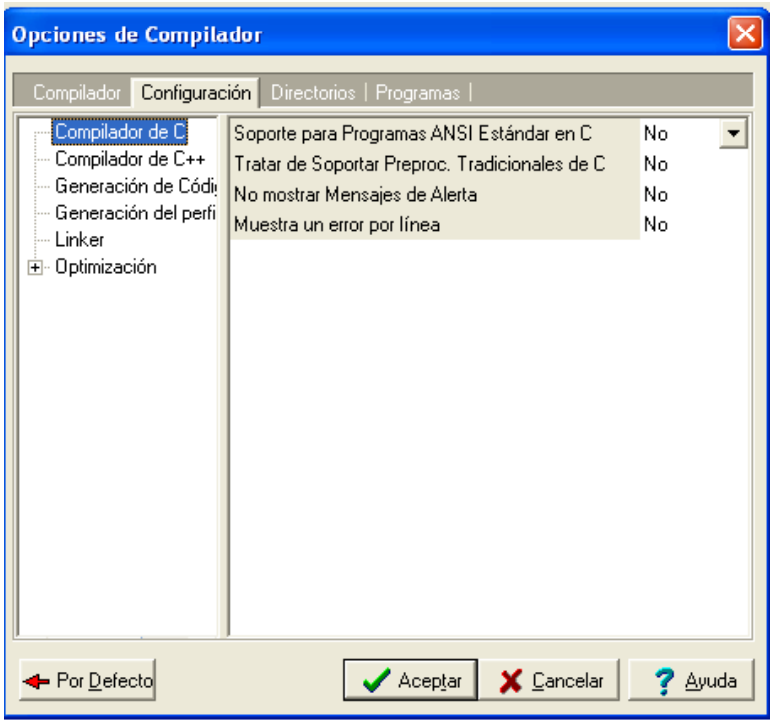
- Es el proceso de identificar y corregir los errores de programación.
- En C++ se realiza normalmente mediante la ejecución, instrucción a instrucción, del programa.
- Las herramientas que nos permiten hacer esto son los depuradores.



Configuración de Dev-C++

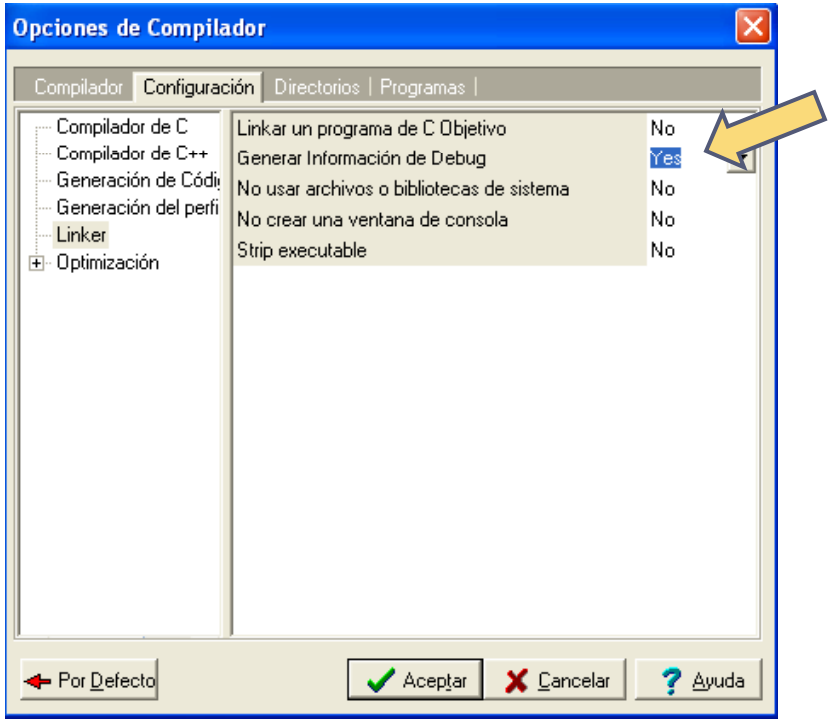
- Tools → Compiler Options
- Hacer click en Settings





Configuración de Dev-C++

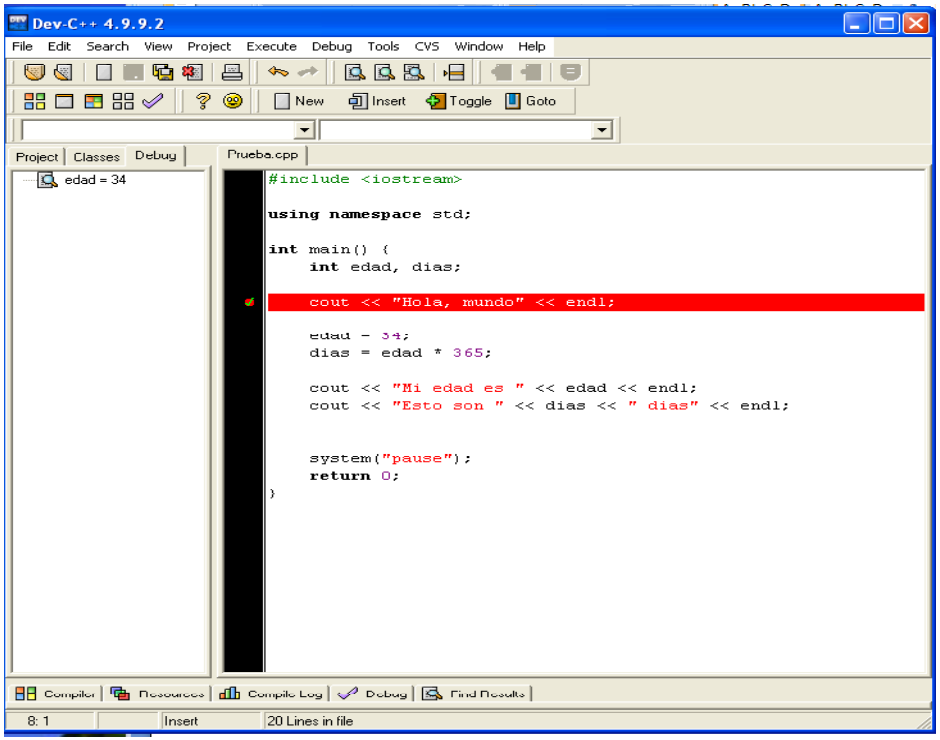
- En la sección Linker, activar la opción Generate debugging information.



- Volver a compilar mediante: Execute → Rebuild All *(Ctrl + F11)*

Puntos de ruptura (Breakpoints)

- Líneas del código en las que la ejecución del programa se detendrá en modo depuración.
  - Debug → Toggle breakpoint *(Ctrl+F5)*



Modo depuración

- Se activa mediante la opción Debug → Debug *(F8)*
- El programa se ejecutará normalmente hasta llegar a un punto de ruptura, en el cual se detendrá.
- Cuando el programa se detiene, puede ejecutarse instrucción por instrucción:
  - Debug → Next Step *(F7)*: Avanza hasta la siguiente sentencia.
  - Debug → Continue *(Ctrl+F7):* Avanza hasta el próximo punto de ruptura.
  - Debug → Run to cursor *(Mayus+F4):* Avanza hasta la posición actual del cursor.
  - Debug → Stop Execution *(Ctrl+Alt+F2)*: Aborta la ejecución del programa y abandona el modo depuración.

Inspección de variables

- Cuando el programa está detenido en modo depuración puede observarse el valor de una determinada variable mediante la opción:
  - Debug → Add Watch.