

LENGUAJES DE PROGRAMACIÓN

LENGUAJE C++

Lenguajes de Programación



Lenguajes de Bajo Nivel

Lenguaje de máquina

(programas ejecutables)



11000000 000000000001 000000000010
11110000 000000000010 000000000011

Lenguajes de Alto Nivel

Visual Basic, C,
C++, Java, Python

Necesitan traducirse a código de máquina, a través de otros programas "traductores" (**compiladores**)



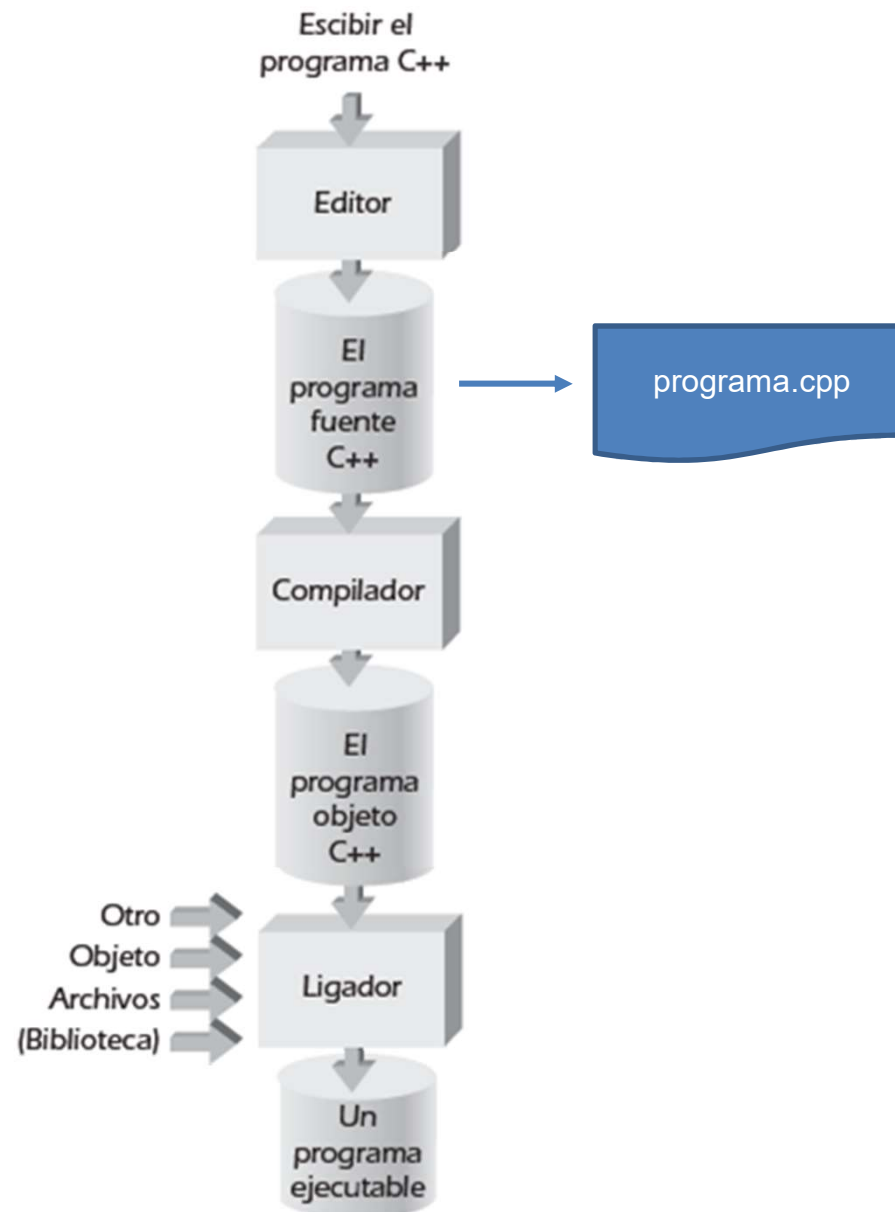
Lenguaje C++

- ▶ El **lenguaje C** es un lenguaje **de procedimiento estructurado** desarrollado en la década de los años 70.
- ▶ Se convirtió en el lenguaje dominante para aplicaciones de ingeniería de la década de los años 80.
- ▶ Tiene un amplio conjunto de capacidades que permite que se escriba como un lenguaje de nivel alto mientras conserva la capacidad de acceso directo a las características del nivel de máquina de una computadora.
- ▶ **El lenguaje C++** fue desarrollado a principios de la década de los años 80 para crear un lenguaje de programación **orientado a objetos**.
- ▶ C++ conservó el conjunto extenso de capacidades estructuradas de C, pero agregó su propia orientación a objetos para convertirse en un verdadero lenguaje de programación de uso general.

Compilación y Ejecución de un Programa

- El algoritmo debe escribirse mediante un programa editor y almacenarse en disco (archivo de programa), convirtiéndose en programa fuente.
- El programa fuente se traduce a lenguaje máquina. Esto lo realiza el compilador con el sistema operativo.
- Si en la compilación se encuentran errores (errores de compilación), se vuelve a editar el programa, se corrigen los errores y se compila de nuevo. Si no hay errores se obtiene el programa objeto (versión en lenguaje de máquina del código fuente).
- Luego hay que realizar un montaje o enlace (link), que es una combinación del programa objeto con las bibliotecas del compilador. Esto produce un programa ejecutable (programa completo en lenguaje de máquina).
- El ejecutable se puede ejecutar o correr (run). Si no hay errores de ejecución, se obtendrá la salida del programa.

Creación de un programa C++ ejecutable



IDEs

Para obtener el programa ejecutable necesitamos **dos herramientas básicas**:

- Un **editor** con el que crear un archivo con el texto del mismo (el código fuente)
- Y un **compilador** para traducir el código fuente a código ejecutable.

Para realizar estos pasos podemos seguir dos enfoques diferentes:

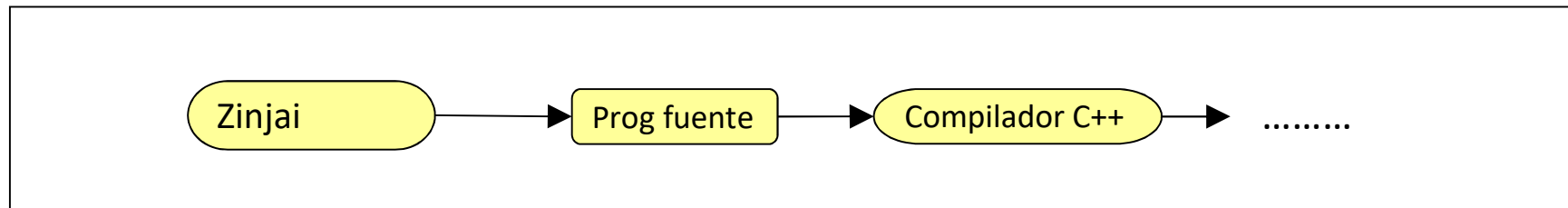
- Usar directamente la línea de comandos del sistema operativo
- O usar un **entorno de desarrollo integrado** (IDE).

En un entorno de desarrollo integrado disponemos de un conjunto de **ventanas y de botones** asociados a las diferentes herramientas mencionadas.



C++: Lenguaje compilado

- Antes de poder **ejecutar** el programa que escribimos, tenemos que **compilar** el código fuente (archivo **.cpp**) con un compilador de C++
- El compilador lee y analiza todo el programa. Si no hay errores lo traduce a código de máquina; sino muestra los errores detectados
- Los compiladores se pueden llamar habitualmente desde la línea de comandos, aunque nosotros usaremos un **entorno de programación (IDE)** que nos facilita la tarea



Lenguaje C++

En general, un programa C++ suele estar escrito en **varios archivos**.

Durante el proceso de compilación estos archivos serán combinados adecuadamente y traducidos a código objeto, obteniendo el programa ejecutable.

Para programas sencillos, bastará un único archivo cuya **extensión** será **“.cpp”**

El archivo suele comenzar con unas líneas para incluir las definiciones de los **módulos de biblioteca** que utilice nuestro programa, e irá seguido de **declaraciones y definiciones de tipos, de constantes y de subprogramas**.

El programa debe contener **un subprograma especial (la función main)** que indica dónde comienza la ejecución. Las instrucciones contenidas en dicha función main se ejecutarán una tras otra hasta llegar a su fin.

La función main devuelve un valor que indica si el programa ha sido ejecutado correctamente o, por el contrario, ha ocurrido un error.

```
#include <iostream>

using namespace std;

int main()
{
    cout << "Hola mundo" << endl;
    return 0;
}
```


Estructura de un Programa en C++

Directivas del preprocesador:

include <xxxx.h>

define

Declaraciones globales:

- Prototipos de funciones
- Variables comunes a todas las funciones del programa

Función principal:

int main ()

{

- Declaraciones locales

- Sentencias

}

Definiciones de otras funciones:

tipo-retorno funcion1 (.....)

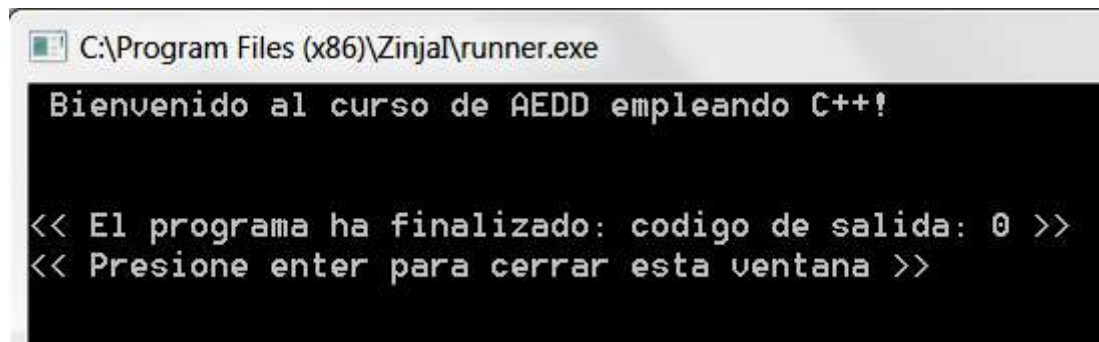
{

.....

}

Estructura de un Programa en C++

```
/* Programa de prueba para mostrar sintaxis de C++ */  
  
#include <iostream>  
using namespace std;  
  
/* Este programa muestra en pantalla un mensaje de bienvenida */  
  
int main() {  
    cout << "Bienvenido al curso de AEDD empleando C++! ";  
  
    return 0;    /* indica que el programa terminó exitosamente */  
} /* Fin de la función main */
```



The screenshot shows a Windows command prompt window titled "C:\Program Files (x86)\Zinja\runner.exe". The output of the program is displayed as follows:

```
Bienvenido al curso de AEDD empleando C++!  
  
<< El programa ha finalizado: código de salida: 0 >>  
<< Presione enter para cerrar esta ventana >>
```

Ejemplo de un Programa en C++

```
//- fichero: euros.cpp -----  
#include <iostream>  
using namespace std;  
const double EUR_PTS = 166.386;  
int main() {  
  
    cout << "Introduce la cantidad (En euros): ";  
    double euros;  
    cin >> euros;  
    double pesetas = euros * EUR_PTS;  
    cout << euros << " Euros equivalen a " << pesetas << " Pts" << endl;  
    return 0;  
} //- fin: euros.cpp -----
```

Se usa el espacio de nombres estandar

Bibliotecas

Constantes simbólicas y literales

Identificadores

Delimitadores

Operadores

Palabras claves

Comentarios

Variables

Entrada/salida

Asignación

Creación y Ejecución de un Programa

- Etapas de la creación de un programa:
 - Definir el programa
 - Definir directivas del preprocesador
 - Definir declaraciones globales
 - Crear main ()
 - Crear el cuerpo del programa
 - Crear las propias funciones definidas por el usuario
 - Utilizar comentarios adecuados
 - Almacenar en disco con nombre.cpp
 - Compilar, enlazar, ejecutar y comprobar el programa

Estas etapas pueden realizarse en un IDE (entorno de desarrollo integrado)

Directivas del Preprocesador

- **Preprocesador**: editor de texto inteligente que procesa **directivas** que son instrucciones al compilador antes que se compile el programa principal.

No son sentencias de C++, por ello no terminan en “;”. Generalmente se escriben al principio.

- **Las 2 directivas más usuales son:**

#include: indica al compilador que lea del disco el archivo fuente (de texto ASCII) que sigue, y su contenido lo inserte en esa posición.

Los archivos tienen código fuente diseñado por el usuario con el formato (<nombre.c>) o son archivos de sistema especiales, denominados **archivos de cabecera**, que residen en el compilador: **iostream.h**: para operaciones de E/S; **string.h**: para operaciones con cadenas (Ej: #include <iostream.h>).

El nombre del archivo puede darse con formato <nombre.h> (si se encuentra en el directorio por defecto) o “path nombre.h” si está en otro directorio (Ej: #include “D:\Mis programas\Prueba.h”).

#define: indica al compilador que defina un ítem de datos u operación (Ej: #define PI 3.141) sustituirá PI por el valor 3.141 cada vez que aparezca en el prog.

Elementos básicos de un Programa

Bibliotecas: El lenguaje C++ consta de un reducido número de instrucciones, pero ofrece un amplio repertorio de bibliotecas con herramientas que pueden ser importadas por los programas cuando son necesarias. Por este motivo, un programa suele comenzar por tantas líneas `#include` como bibliotecas se necesiten.

Constantes: Podemos emplear valores en su `sentido literal` (tal y como está escrito), pero también se puede definir un nombre para referirnos al mismo. Ello resulta más claro y permitirá que el `programa sea más legible y fácil de modificar en el futuro`.

Antes de utilizar una constante simbólica es necesario informar al compilador, indicando de qué tipo es, su nombre (`identificador`) y el `valor` que tiene asociado.

Elementos básicos de un Programa

Identificadores:

Para cada elemento que introduzcamos en nuestro programa debemos definir un **nombre (identificador)** con el que hacer referencia al mismo y disponer de algún mecanismo para informar al compilador de dicho nombre y de sus características

En C++ se considera que las letras minúsculas son caracteres diferentes de las letras mayúsculas.

Un identificador debe estar formado por una secuencia de letras y dígitos en la que el primer carácter debe ser una letra.

El carácter '_' puede utilizarse, sin embargo, los nombres no deben comenzar con dicho carácter.

Son sensibles a las mayúsculas.

Pueden ser extensos, pero son significativos los 32 primeros caracteres.

No pueden usarse palabras reservadas.

Ejemplos de identificadores:

nombre_cliente

dni

Mayor_item

fechaInicio

Elementos básicos de un Programa

Palabras reservadas:

Algunas palabras tienen un significado especial en el lenguaje y no pueden ser utilizadas con otro sentido.

Por eso no pueden ser utilizados para designar elementos definidos por el programador.

Son palabras reservadas, por ejemplo: `using`, `namespace`, `const`, `double`, `int`, `char`, `bool`, `void`, `for`, `while`, `do`, `if`, `switch`, `case`, `default`, `return`, `typedef`, `struct`, entre otras.

Delimitadores:

Son símbolos que indican comienzo o fin de una entidad (`() { } ; , < >`). Por ejemplo, en nuestro programa `euros.cpp` usamos `{ }` para delimitar el comienzo y el final de la función `main`, y el símbolo `;` para delimitar el final de una sentencia.

Operadores:

Son símbolos con significado propio según el contexto en el que se utilicen. Ejemplo: `= << >> * / % + - < > <= >= == != ++ -- . ,`, etc.

Elementos básicos de un Programa

Comentarios y formato del programa:

El compilador necesita reconocer los diferentes elementos que forman el programa. Para ello, utiliza **delimitadores y caracteres** que permiten separar unos elementos con otros.

Además, el programador puede añadir caracteres como espacios en blanco, líneas en blanco o tabuladores para **mejorar la legibilidad del programa**. El compilador los ignora.

Además, el programador puede estar interesado en mejorar la legibilidad del programa incluyendo **comentarios dirigidos a otros programadores**, no al compilador.

Para ello, debe marcar dicho texto de alguna forma que permita ser identificado por el compilador. En C++ esto se puede hacer de dos formas diferentes:

- **Comentarios hasta fin de línea:** los símbolos // marcan el comienzo del comentario, que se extiende hasta el final de la línea.
- **Comentarios enmarcados:** los símbolos /* marcan el comienzo del comentario, que se extiende hasta los símbolos de fin del comentario */.

Elementos básicos de un Programa

Variables:

Los datos se almacenan en memoria en variables de un cierto tipo. El programador debe decidir qué variables va a utilizar, pensar en un identificador para referirse a ellas y comunicarlo al compilador

Entrada/Salida de datos:

En general, un programa necesitará tomar datos de entrada y mostrar resultados en algún dispositivo de salida.

En C++ disponemos de flujos de entrada (**cin**) y de salida (**cout**), que nos permiten efectuar entrada y salida de datos, respectivamente. El operador **>>** se usa para tomar un valor de la entrada y el operador **<<** se usa para sacar un valor por la salida.

Asignación:

Si queremos realizar el cálculo y almacenar el resultado de manera que pueda ser utilizado posteriormente, utilizamos una instrucción de asignación en la que el operador **=** separa el valor a almacenar, resultado de evaluar una expresión aritmética (a la derecha), de la variable en la que lo queremos almacenar (asignar).

Depuración de un Programa

Se denomina así al proceso de encontrar errores. Los errores pueden ser:

- **Errores de Sintaxis:** Se violan las reglas de gramática del lenguaje (Ej: punto y coma después de la cabecera, omisión de “;” al final de sentencia, olvido de símbolos apareados (corchetes, paréntesis, /*..)).
- **Errores Lógicos:** Errores del programador en el diseño del algoritmo y posterior programa. No los detecta el compilador. Se detectan con cuidadoso análisis de los resultados (testing). Se puede necesitar un depurador (debugger).
- **Errores de Ejecución** se presentan después que el programa fue compilado, y cuando está en ejecución.
- Los compiladores emiten **mensajes de error o de advertencia**. Los mensajes de error del compilador suelen deberse a errores de sintaxis y pueden ser:
 - ✓ Fatales: no puede realizar la compilación.
 - ✓ De Sintaxis: avisa y termina.
 - ✓ Advertencia (warning): indican condiciones legítimas pero sospechosas.

Pruebas

- **Los Errores de Ejecución** se presentan después que el programa fue compilado, y cuando está en ejecución.
- El compilador puede no emitir ningún mensaje de error durante la ejecución, y eso no garantiza que el programa sea correcto pues solo indica si un programa se escribió bien sintácticamente. No indica si el programa hace lo que se desea que haga (**eficaz**).
- Para determinar si un programa tiene un error lógico, se debe ejecutar utilizando **datos de muestra** y comprobar la salida (**prueba – testing**). Esta prueba se ejecuta con diferentes entradas preparadas y seleccionadas preferentemente por personas diferentes al programador.
- Si se ha determinado que el programa tiene un error lógico y no se lo encuentra fácilmente, para localizarlo se debe realizar la **ejecución paso a paso**, **seguir la traza**. Algunos compiladores tienen un depurador (debugger) incorporado con el editor.

Máxima de Dijkstra (científico holandés): *Las pruebas sólo muestran **la presencia** de errores, **no su ausencia**. No se puede probar que un programa es correcto (exacto), sólo se puede mostrar que es incorrecto.*

LEER:

Capítulo 2 del libro de Bronson, 2da Edición,

“C++ para ingeniería y ciencias”.