Notas Programación en C++

Cristo Daniel Alvarado

27 de enero de 2026

Índice general

Capítulo 1

Básicos

§1.1 Introducción

Un código simple de C++ debería verse así:

```
1 #include <iostream>
2 using namespace std;
3
4 int main() {
5   cout << "Hello World!";
6   return 0;
7 }
```

Listing 1.1: myfirstprogram.cpp

Varias cosas a analizar en este fragmento de código:

- La instrucción `#include iostream` es una **header file library**, que nos permite trabajar con objetos de entrada y salida (mediante la terminal). Todos los header files que se deseen incluir deberán ser de esta forma, cambiando dependiendo del nombre del archivo de cabecera.
- `using namespace std` significa que podemos usar nombres para objetos y variables de la librería estándar.
- La función `int main()` es esencial para todo código de C++, es ella la que se ejecuta inicialmente y dónde va toda la información de la primera ejecución del código.
- Finalmente, el `return 0;` es la salida del programa que indica que no hubo ningún problema en la ejecución.

Observación 1.1.1
La instrucción `cout` (en C++ es diferente `cout` de `Cout`) es un **objeto** usado junto con el **operador de inserción** para imprimir texto o dar una salida. En este caso le estamos diciendo que imprima `Hello World!`.

Observación 1.1.2
Puede que en algunos códigos de C++ se omita la línea `using namespace std` y sea reemplazada en su lugar declarando siempre `std::`, que es la palabra reservada `std` junto con el operador `::`.

```
#include <iostream>

int main() {
  std::cout << "Hello World!";
  return 0;
}
```

Listing 1.2: Código en C++ donde no se usa la instrucción `using namespace std`.

La salida de este programa es la misma que la de (1.1).

Toda línea en C++ debe ser terminada en `;`. Una línea de código es también llamada **instrucción**. Un código de C++ se conforma de muchas instrucciones. Estas son ejecutadas en el orden en que fueron escritas por el usuario.

1.1.1. Salida de programa

El objeto `cout` junto con el operador son usados para imprimir valores e imprimir texto como en el código (1.1).

Se pueden añadir cuantos quiera objetos `cout` como se desee, sin embargo, *no se añadirán en nuevas lineas*, solo se seguirán poniendo en la misma.

```
#include <iostream>
using namespace std;

int main() {
  cout << "Hello World!";
  cout << "I am learning C++";
  return 0;
}
```

Listing 1.3: Salida de múltiples `cout`

El código anterior tiene como salida `Hello World!I am Learning C++`. Con `cout` también es posible imprimir números y operaciones resultantes de números, como las siguientes:

```
cout << 3;
cout << 2+3;
cout << 2*3;
```

Listing 1.4: Operaciones dentro de `cout`

El programa tiene como salida `356`, que son respectivamente cada una las salidas línea por línea del programa después de su ejecución.

Para insertar nuevas líneas en el programa se usa al final de la línea que estamos escribiendo el caracter `\n`.

Por ejemplo, el código:

```cpp
#include <iostream>
using namespace std;

int main() {
  cout << "Hello World!" << "\n";
  cout << "I am learning C++";
  return 0;
}
```

tiene como salida:

```
Hello World!
I am learning C++
```

en vez de todo en una sola línea.

Observación 1.1.3
Con $\ll$ se pueden insertar más caracteres de texto en la salida estándar.

Otra forma de hacer un salto de línea es usar $\ll$ `endl` después de lo que sea que queramos poner, como en el siguiente código:

```cpp
#include <iostream>
using namespace std;

int main() {
    cout << "Hello World!" << endl;
    cout << "I am learning C++";
    return 0;
}
```

Definición 1.1.1
El caracter `\n` es llamado **secuencia de escape**, y fuerza el cursor a cambiar su posición al inicio de la siguiente línea en la pantalla. Esto resulta en una nueva línea.

Otros caracteres de escape son:

- `\t`: tabular.
- `\\`: inserta un \.
- `\''`: inserta comillas.

§1.2 Variables

En C++, una variable se declara de la siguiente manera:

```cpp
type variableName = value;
```

algunos tipos de variables son:

- `int`.

- `double`.
- `char`.
- `string`. Guarda texto, a diferencia de C, C++ si tiene este tipo de variable.
- `bool`. Guarda dos estados, verdadero y falso.

Como en muchos otros lenguajes, se puede crear la variable y asignar su valor después. Esto reescribe el valor anterior que tenía (si es que había un valor anterior).

Algunos ejemplos de declaraciones de variables:

```
int myNum = 5;               // Integer (whole number without
    decimals)
double myFloatNum = 5.99;    // Floating point number (with
    decimals)
char myLetter = 'D';         // Character
string myText = "Hello";     // String (text)
bool myBoolean = true;       // Boolean (true or false)
```

1.2.1. Impresion en la salida estándar

Las variables se imprimen como si fuesen texto, con el objeto `cout` y el operador $\ll$:

```
int myAge = 35;
cout << "I am " << myAge << " years old.";
```

Estas variables se pueden operar, por ejemplo con el operador binario +.

1.2.2. Múltiples Variables

Se pueden también declarar múltiples variables, como en los siguientes dos códigos:

```
int x = 5, y = 6, z = 50;
cout << x + y + z;
```

```
int x, y, z;
x = y = z = 50;
cout << x + y + z;
```

Todas las variables llevan un nombre único, denominado **identificador**.

§1.3 Operadores

El operador `<<` es el llamado **operador inserción (insertion operator)** y es usado para imprimir texto en la salida de comandos.

`C++` es una extensión de `C`, por lo que tiene exactamente las mismas variables que `C`.

§1.4 User Input

`cout` es para salida de texto en la consola de comandos, `cin` es para la entrada de texto.

Definición 1.4.1 (cir y ≪)
`cin` es una variable predefinida que lee información del teclado con el **operador extracción (extraction operator) >>**

Con este ejemplo imprimimos el valor de `x`, antes guardamos la entrada de texto en esta variable.

Información adicional

Yo para compilar uso en VSCode la extensión de gdc (me parece que así se encuentra en la tienda de extensiones). Todo lo compilo de esa manera, aparte de que uso algunas extensiones adicionales para ayudarme con la parte de programación para que la escritura del código sea más eficiente de lo normal.