PRIMEROS PASOS EN C++

CADA APARTADO PRINCIPAL CORRESPONDE CON EL NOMBRE DEL PROYECTO CON EJEMPLOS DE LO APUNTADO AQUÍ

1. PRIMER PROGRAMA EN C++

- **LOCALIZAR EL LENGUAJE A UTF8**

```
#include <locale.h>
setlocale(LC_ALL, ".UTF8");
```

- **IMPRIMIR POR PANTALLA**
  - Característica de C++ que permite cargar librerías por defecto que podemos usar, en concreto iostream nos permite imprimir por pantalla: `#include <iostream>`
  - Imprimir por pantalla: `std::cout << "Lo que quiera imprimir"`
  - Imprime un salto de línea, tiene que ir en la misma línea del `cout`: `std::endl`
  - Ambos requieren de `<iostream>`
- **GENERAL**
  - A veces se incluye `return 0` al final del *main* ya que al ejecutarse toda la función de forma secuencial es una buena forma de ver si hay errores o no.

2. COMENTARIOS

- **COMENTARIOS**
  - Comentario de línea: `//Línea comentada`
  - Comentario de bloque: `/*`

```
Bloque comentado
*/
```

3. ERRORES Y AVISOS

- **ERRORES Y AVISOS**
  - Error de compilación: El programa no se compila correctamente por un error en nuestro código. Por ejemplo al no poner el ; cuando es debido.

  - Error de ejecución: El programa se compila correctamente, pero a la hora de la ejecución el programa no hace nada o crashea esto se debe a un error en la lógica de nuestro programa. Por ejemplo, al dividir entre 0.

  - Aviso (Warning): Avisos del compilador que no llegan a detener la compilación pero que es importante revisar y tener en cuenta.

4. SENTENCIAS Y FUNCIONES

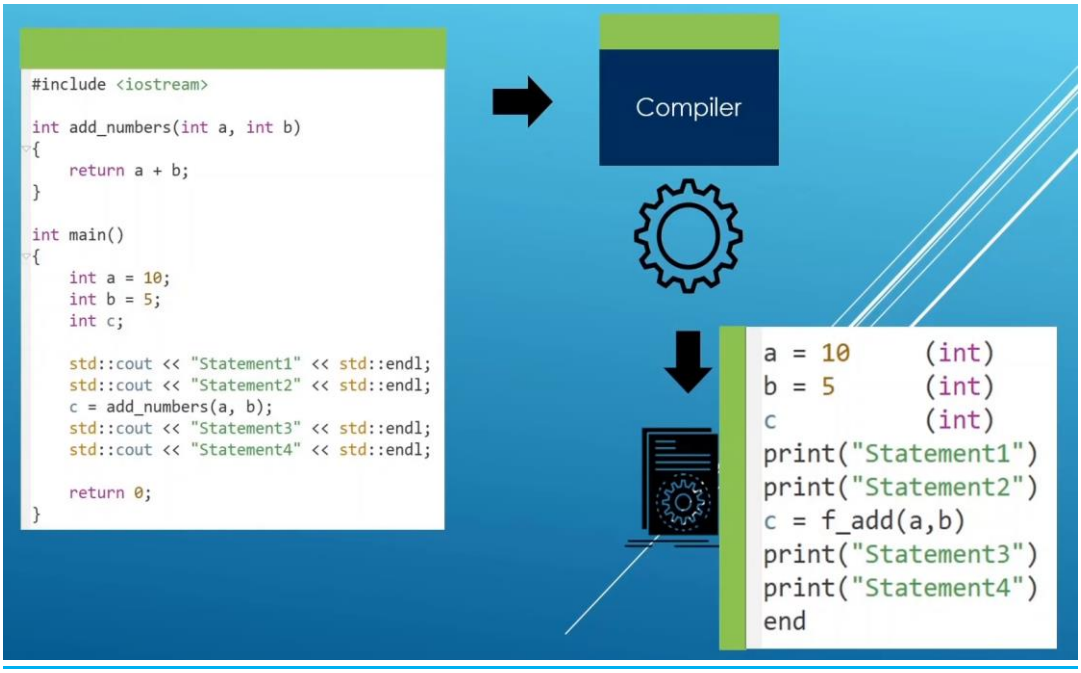
- Una sentencia es la unidad más básica de computación en un programa de C++, todo programa de C++ es una colección de sentencias organizadas para conseguir lo que sea que queramos, las sentencias terminan con ; en C++.
- Las sentencias se ejecutan secuencialmente (desde arriba hasta abajo) y la ejecución continúa hasta que el programa acabe o se ejecute otra secuencia de sentencias (una función).
- La sintaxis de las funciones es: `tipoDeDato nombreFunción (param1, param2) { }`
- Una función debe ser definida antes de ser llamada (obviamente).

5. ENTRADA Y SALIDA DE DATOS

- En la sentencia `std::cout << textoPorPantalla` hay que pensar que el `textoPorPantalla` va hacia `std::cout` y desde `std::cout` hacia la consola.
- Hay más flujos de entrada además de `std::cout`, como:
  - `std::cout` → Imprime datos en la consola.
  - `std::cerr` → Imprime los errores en la consola.
  - `std::clog` → Imprime logs en la consola.

- `std::cin` → Lee datos desde el terminal. En este caso el orden de los datos es a la inversa de como es con `std::cout` . Aquí los datos van de la consola, a `std::cin` y a el programa.
  Para capturar un `string` que contenga espacios (un nombre completo por ejemplo) podemos usar `std::getline(std::cin, std::string variableNombreCompleto)`. Puntualizar que para poder usar `string` hay que incluirlo en el header con `#include<string>` y para declarar un `string` hay que usar `std::string` en vez de `string` a secas, más información al respecto en el proyecto correspondiente de este apartado.

6. MODELO DE EJECUCIÓN DE PROGRAMAS EN C++



a. El dibujo negro debajo de la rueda del compilador, representa un archivo ejecutable, es decir binario y por tanto no legible por un humano, pero si por la CPU.
b. https://youtu.be/8jLOx1hD3_o?t=10269

7. C++ CORE LANGUAGE VS STANDARD LIBRARY VS STL

- **CORE LANGUAGE**

- Las opciones básicas de C++, como se definen las variables y las funciones, las reglas que dicen lo que puedes hacer y lo que no, la sintaxis, etc.

- **STANDARD LIBRARY**
  - Las librerías, componentes muy especializados directamente preparados para ser usados en nuestros programas de C++

- **STL**
  - Forma parte de la Standard Library y es una colección muy especializada de contenedores de tipos.