1.2 Tipos de datos simples

1.2 Tipos de datos simples

1. Introducción
2. Declaración vs definición
3. Tipos de datos simples
4. Enteros (int)
5. Decimales (float y double)
6. Booleanos (bool)
7. Carácter (char)
8. Tabla Resumen

---

1. Introducción

Un programa trabaja con datos con unas determinadas características. No es lo mismo procesar números naturales, números reales o nombres (cadenas de caracteres). En cada caso tratamos con datos de tipo diferente. C++ dispone de un número reducido de tipos predefinidos y es importante elegir en cada caso aquel que mejor se adapta a los datos que deseamos manipular.

La elección de un determinado tipo u otro para una determinada entidad del programa proporciona información acerca de:

- El rango de posibles valores que puede tomar.
- El conjunto de operaciones y manipulaciones aplicables.
- El espacio de almacenamiento necesario para almacenar dichos valores.
- La interpretación del valor almacenado.

2. Declaración vs definición

Una **definición** es crear una variable pero sin darle ningún tipo de valor. Esto se suele realizar cuando sabemos 100% que se le va a dar un valor a dicha variable posteriormente.

Una **declaración** es crear la variable ya con una valor de comienzo por defecto. Esa variable puede cambiar de valor, a través de asignaciones posteriores.

```
int suma; // Defino la variable suma
suma = 10; // Le asigno una valor
```

```
int suma = 0; // Declaro la variable suma
suma = 10; // Le asigna un valor después de crearla
```

CUIDADADO!!!

Si definimos o declaramos una variable y luego no se da un uso, el compilador nos da un **Warning**, que es un aviso de que hemos ocupado un espacio en memoria que no estamos aprovechnando. Por eso, es conveniente tener cuidado cuando realizamos la declaración o definición de variables, y en caso de no usarlas, se borren.

3. Tipos de datos simples

El lenguaje de programación C++ proporciona los siguientes tipos simples predefinidos:

```
int float double bool char
```

4. Enteros (int)

El tipo ***int*** se utiliza para trabajar con números enteros. Su representación suele coincidir con la definida por el tamaño de palabra del procesador sobre el que va a ser ejecutado, hoy día suele ser de 4 bytes (32 bits), aunque en determinados ordenadores puede ser de 8 bytes (64 bits).

Puede ser modificado para representar un rango de valores menor, mediante el modificador ***short*** (normalmente 2 bytes [16 bits]) o para representar un rango de valores mayor, mediante el modificador ***long*** (normalmente 4 bytes [32 bits] u 8 bytes [64 bits]) y ***long long*** (normalmente 8 bytes [64 bits]). También puede ser modificado para representar solamente números naturales (enteros positivos) utilizando el modificador ***unsigned***.

```
int num;  // numero entero
short int num; // numero entero de menor rango
long int num; // numero entero de mayor rango
long long int num; // numero entero de mayor rango
unsigned int num; // numero enteros positivos
```

5. Decimales (float y double)

Tanto el tipo ***float*** como el ***double*** se utilizan para representar números reales en formato de punto flotante. Se diferencian en el rango de valores que se utiliza para su representación interna. El tipo double (“doble precisión”) se suele representar utilizando 8 bytes ([64 bits]), mientras que el tipo float (“simple precisión”) se suele representar utilizando 4 bytes [32 bits]). El tipo double también puede ser modificado con long para representar “cuádruple precisión” (normalmente 12 bytes [96 bits]).

```
float d; // numero decimal
double d; // numero decimal
long double d; // decimal más rango
```

6. Booleanos (bool)

El tipo ***bool*** se utiliza para representar valores lógicos (o booleanos), es decir, los valores "Verdadero" o "Falso" o las constantes lógicas ***true*** y ***false***. Suele almacenarse en el tamaño de palabra más pequeño posible direccionable (normalmente 1 byte).

```
bool p = true; // booleano
```

7. Carácter (char)

El tipo char se utiliza para representar caracteres, es decir, símbolos alfanuméricos (dígitos y letras mayúsculas y minúsculas), de puntuación, espacios, control, etc. Normalmente utiliza un espacio de almacenamiento de 1 byte (8 bits) y puede representar 256 valores diferentes.

```
char i = 'c';
```

8. Tabla Resumen

Tipo	Bytes	Bits	Min.Valor	Max.Valor
`bool`	1	8	`false`	`true`
`char`	1	8	-128	127
`short`	2	16	-32768	32767
`int`	4	32	-2147483648	2147483647
`long`	4	32	-2147483648	2147483647
`long long`	8	64	-9223372036854775808	9223372036854775807
`unsigned char`	1	8	0	255
`unsigned short`	2	16	0	65535
`unsigned`	4	32	0	4294967295
`unsigned long`	4	32	0	4294967295
`unsigned long long`	8	64	0	18446744073709551615
`float`	4	32	1.17549435e-38	3.40282347e+38
`double`	8	64	2.2250738585072014e-308	1.7976931348623157e+308
`long double`	12	96	3.362103143112093506 26e-4932	1.189731495357231765 02e+4932