

Interacción con un programa

Recordemos que un programa de computadora nos ayuda a resolver problemas. En general, aunque no de manera obligatoria, un programa puede cumplir el siguiente esquema:



En nuestros primeros programas, la **entrada** y la **salida** se darán por **consola**, es decir, utilizando el **teclado** y el **monitor**.

Se implementan mecanismos de **entrada por consola** utilizando los comandos:

cin >> que se encuentra en la librería **iostream** (a partir de C++).

scanf(), que se encuentra en la librería **stdio.h** (desde inicios de C).

Se implementan mecanismos de **salida por consola** utilizando los comandos:

cout << que se encuentra en la librería **iostream** (a partir de C++).

printf(), que se encuentra en la librería **stdio.h** (desde inicios de C).

Más recursos para la salida por consola

Otros comandos y símbolos que se pueden utilizar al desplegar en pantalla:

endl que se encuentra en la librería **iostream** (a partir de C++).

El carácter **\n**, equivalente a **endl**, es su representación simbólica (código 10 de la tabla ASCII). Está disponible desde las primeras versiones de C.

El carácter **\t**, representación simbólica de tabulación (código 9 de la tabla ASCII). Está disponible desde las primeras versiones de C.

El carácter **\a**, representación simbólica de pitido de bocina (código 7 de la tabla ASCII). Está disponible desde las primeras versiones de C.

Y otros más.

Ejemplos:

```
#include <iostream>
```

```
using namespace std;
```

```
int main(void)
```

```
{
```

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

```

    cout << endl;
    cout << "Soy tu primer programa";

    return 0;
}

```

El mismo efecto en la salida tiene:

```
cout << "Hola mundo" << endl << "Soy tu primer programa";
```

El símbolo de redirección de flujo, `<<`, funciona como separador entre los diferentes elementos de salida.

```
cout << "Hola mundo" << "\n" << "Soy tu primer programa";
```

Notar que `\n` es un carácter, por tanto va encerrado entre comillas simples. Estrictamente hablando, los caracteres van encerrados entre comillas simples y las cadenas van encerradas entre comillas dobles.

El símbolo `\n`, es un carácter especial, tiene significado para el programa. Lo indica la plica invertida que le antecede. Lo mismo se puede decir de los demás caracteres especiales. Su representación en el programa consta de dos caracteres tipográficos, siendo el primero `\`. Pero el proceso de compilación lo traduce a un solo carácter.

```
cout << "Hola mundo\n" << "Soy tu primer programa";
```

Siendo un carácter, puede pasar a formar parte de una cadena.

```
cout << "Hola mundo\nSoy tu primer programa";
```

Puede colocarse en cualquier lado de la cadena.

Operaciones a la salida

Podemos desplegar resultados numéricos, que se dan a partir de la implementación de operaciones.

Por ejemplo:

```
cout << 5;
```

```
cout << 5 << endl;
```

```
cout << 5 << "\n";
```

Estas dos últimas realizan un cambio de línea después de desplegar el valor numérico.

```
cout << 5 + 3 << endl;
```

Realiza la operación y luego despliega el resultado.

```
cout << 5 + 3 * 6 << endl;
```

```
cout << (5 + 3) * 6 << endl;
```

El uso de paréntesis cambia la jerarquía de operación, tal y como se espera.

```
cout << sqrt(2) << endl;
```

```
cout << pow(3, 2) << endl;
```

Para obtener raíz cuadrada y elevar a potencia se requiere de las funciones `sqrt(valor)` y `pow(base, exponente)`, que se encuentran en la librería `cmath` de C++ (`math.h` en C). Ambas librerías pueden declararse.

```
cout << "La raíz cuadrada de 2 es: " << sqrt(2) << endl;
```

```
cout << "3 elevado a 2 es: " << pow(3, 2) << endl;
```

Se puede combinar cualquier cantidad de elementos y de diferente naturaleza y tipo en un comando de salida.

La división tiene una forma particular de comportarse:

Operación	Resultado
Entero / Entero	Entero (parte entera del resultado)
En cualquier otro caso: Entero / Real Real / Entero Real / Real	Real (número con parte entera y parte fraccionaria)

Otro operador de división:

Podemos calcular el residuo de la división entera con el operador `%`:

```
cout << 10 % 3 << endl;
```

Esta operación desplegará 1 en pantalla.

Ingreso de datos a nuestros programas

Para efectuar el ingreso de datos a un programa, no importando el lenguaje de que se trate, el programa utiliza la memoria RAM de la computadora para almacenar temporalmente los datos, mientras estos son procesados.

Esto se logra a través del uso de `variables`, que son espacios de memoria con nombre. Nuestro programa podrá hacer uso del dato colocado en ese espacio de memoria a través de enunciar su nombre.

Las variables son utilizadas para la manipulación de datos. En general, se utilizan en las operaciones aritméticas, de comparación y de cualquier otro tipo. También

deben ser utilizadas para capturar los datos que se envían desde el teclado hacia el programa.

Toda variable debe ser definida, o declarada, antes de ser utilizada. La declaración consiste en indicar el tipo del dato que va a contener, y el nombre asignado a la variable.

Para declarar una variable se utiliza la sintaxis:

`<tipo> <lista de variables>;`

La lista de variables puede constar de una o más variables, separadas por coma.

Los tipos de datos que maneja el lenguaje C++ se muestran a continuación.

Algunos tipos de datos del lenguaje C++

Tipos enteros

Tipo	Descripción	Cantidad de bytes que ocupa	Intervalo de valores
<i>char</i>	Carácter	1	-128 a 127
<i>short</i>	Entero corto	2	-32768 a 32767
<i>int</i>	Entero	4	-2,147,483,648 a 2,147,483,647
<i>long</i>	Entero largo	4	-2,147,483,648 a 2,147,483,647

Estos tipos pueden ser precedidos de las palabras reservadas *signed* y *unsigned*.

Por defecto, estos tipos consideran el signo, así que la palabra reservada *signed* es solo por compatibilidad.

Al preceder uno de estos tipos con la palabra reservada *unsigned*, el intervalo de valores representables cambia. Por ejemplo:

Para *unsigned char* el intervalo de representación es de 0 a 255.

Para *unsigned short*, el intervalo de valores es de 0 a 65535.

Ojo, con el tipo *char* se manipulan los caracteres individuales y lo hace manejando el valor numérico que le corresponde en la tabla ASCII. Así que con una variable de este tipo se pueden hacer operaciones aritméticas.

Tipos reales

Tipo	Descripción	Cantidad de bytes que ocupa	Intervalo de valores
<i>float</i>	Número en punto flotante en precisión simple.	4	Positivos: 3.4×10^{-38} a 3.4×10^{38} Negativos: -3.4×10^{-38} a -3.4×10^{38}
<i>double</i>	Número en punto flotante en doble precisión.	8	Positivos: 1.7×10^{-308} a 1.7×10^{308} Negativos: -1.7×10^{-308} a -1.7×10^{308}

<i>long double</i>	Real largo	doble	10	Positivos: $3.4 \cdot 10^{-4932}$ a $1.1 \cdot 10^{4932}$ Negativos: $-3.4 \cdot 10^{-4932}$ a $-1.1 \cdot 10^{4932}$
--------------------	---------------	-------	----	--

Tipo booleano

Tipo	Descripción	Cantidad de bytes que ocupa	Intervalo de valores
<i>bool</i>	Datos de tipo lógico.	1	1 (<i>true</i>) ó 0 (<i>false</i>)

Este tipo no existe en C, fue implementado cuando surgió C++.

En C, ya que no manejaban el tipo *bool*, como palabra reservada, resolvían en valor booleano de las variables de acuerdo a su contenido, así: toda variable que contenga un valor diferente de cero, se considera de valor booleano verdadero (*true*) ; toda variable que contenga un valor de cero, se considera de valor booleano falso (*false*).

Esto mismo sigue siendo válido en C++, aunque cuente con este nuevo tipo.

Tipo cadena

Tipo	Descripción	Cantidad de bytes que ocupa	Intervalo de valores
<i>string</i>	Hileras o cadenas de caracteres.	Varía dependiendo del tamaño de la cadena	- -

Para declarar variables de tipo *string* se necesita declarar la librería *string*: `#include <string>`

Ejemplos de uso de variables:

```
int a;
```

Esta instrucción declara una sola variable, llamada a, que podrá almacenar datos de tipo entero.

```
int a, b;
```

Las variables a y b, siendo ambas del mismo tipo, pueden ser declaradas en la misma instrucción.

```
int a;
```

```
float x;
```

Variables de diferente tipo requieren ser declaradas, cada una, en una instrucción que indica su tipo.

```
int a, b;
```

```
float x, y;
```

Ejemplos del uso de variables para capturar datos de entrada al programa:

```
int a;
```

```
cout << "Ingese un valor entero: ";
```

```
cin >> a;
```

Observar:

```
int a, b, c;
```

```
cout << "Ingrese tres enteros: ";
```

```
cin >> a >> b >> c;
```

Este ejemplo captura tres datos enteros de entrada y los almacena en tres variables, en la misma instrucción.

```
int a;
```

```
float estatura;
```

```
cout << "Ingrese su edad y su estatura en metros: ";
```

```
cin >> a >> estatura;
```

Los lenguajes de programación de computadora permiten definir variables cuyo nombre es de mayor longitud de un carácter, con el propósito de ser didácticos en los programas.

```
int n;
```

```
float precio, costo;
```

```
cout << "¿Cuántas pupusas vas a comerte? ";
```

```
cin >> n;
```

```
cout << "¿Cuál es el precio de las pupusa? ";
```

```
costo = n * precio;
```

```
cout << "El precio a pagar es: " << costo << endl;
```

La última instrucción de despliegue es equivalente a :

```
cout << "El precio a pagar es: " << n * precio << endl;
```

Pero en la mayoría de los casos es conveniente manipular los resultados de cálculos por medio de variables.