Definiciones Preliminares

- **Algoritmo**: Conjunto de instrucciones precisas, ordenadas y finitas que permiten llevar a cabo una actividad mediante pasos sucesivos.
- Un algoritmo debe ser
  - preciso
  - finito
  - correcto
  - independiente del lenguaje de programación
- **Estructura de Datos**: Forma particular de organizar datos en una computadora para que pueda ser utilizado de manera eficiente.

Lenguaje C++

- Creado por el danés Bjarne Stroustrup
- Comenzó a trabajar en 1979 con "C con clases", cambió el nombre a C++ en 1983

Año	Estándar	Nombre
1998	ISO/IEC 14882:1998	C++98
2003	ISO/IEC 14882:2003	C++03
2011	ISO/IEC 14882:2011	C++11
2014	ISO/IEC 14882:2014	C++14
2017	ISO/IEC 14882:2017	C++17
2020	ISO/IEC 14882:2020	C++20

- Próximo estándar previsto para 2023
- https://en.cppreference.com/w/cpp/links

Objetivos al programar

- Nuestro código debe ser:
  - **Correcto**: de nada sirve que sea rápido si el resultado no es el esperado
  - **Simple** (y claro): hace más fácil de entender el código y por lo tanto de mantenerlo.
  - **Eficiente**: aprovechar lo mejor posible los recursos disponibles.
- El orden es importante, salvo casos particulares donde el segundo y tercer punto pueden invertirse.

Ejemplo Simple

```
#include <iostream>
/* Comentario en más
   de una línea */
int main()
{
    std::cout << "Hola Mundo!" << std::endl;
    // cout y endl están declaradas en iostream
    return 0;
}
```

Estructura general (muy simplificada) de un programa C++:

- Directivas del preprocesador
- Definiciones de tipos de datos
- Implementación de funciones

Datos

- Para poder manipular datos hay que “tenerlos” en algún lado. Por supuesto se usa la memoria RAM de la computadora.
- Ese “algún lugar” donde ponemos un dato en la memoria lo llamamos un objeto. Un objeto es una región de memoria con un tipo que especifica qué clase de información se puede colocar en ella.
  - **Tipo**: define un conjunto de valores posibles y un conjunto de operaciones (para un objeto).
  - **Objeto**: zona de memoria que contiene un valor de un tipo dado.
  - **Valor**: es un conjunto de bits en la memoria interpretados según un tipo (codificación del valor abstracto).
  - **Variable**: es un objeto con nombre.
  - **Declaración**: es una sentencia que da un nombre a un objeto.
  - **Definición**: es una declaración que reserva memoria para un objeto.

Tipos de Datos Básicos

- Numéricos Enteros
  - **char** : Caracteres
  - **int** : Números enteros
    - de distintos tamaños: short, long, long long
    - Con o sin signo: signed, unsigned
- Numéricos “Reales”
  - **float** : Punto flotante de 32 bits
  - **double** : Punto flotante de 64 bits
  - **long double** : Punto flotante de mayor precisión 80 a 128 bits
- Lógicos
  - **bool**: true, false

Definiciones

tipo_dato nombre_variable;

```
char c;
unsigned long cantidad;
int i;
```

tipo_dato nombre_variable = valor_inicial;
tipo_dato nombre_variable {valor_inicial}; (desde C++11)

```
short int dato = 52000; /* No controla tamaños, convierte
                           a -13536 */
short int dato {52000}; /* error: narrowing conversion of
                ‘52000’ from ‘int’ to ‘short int’ inside { } */
char letra = 'A';
unsigned long total {0xA01D7F};  // 10493311UL
bool flag = true;
```

Cada variable debe inicializarse tan pronto como sea posible

Definiciones

NO se recomienda hacer varias definiciones en una misma línea, si bien el lenguaje lo permite.

tipo_dato nombre_variable1 , nombre_variable2;

```
float fuerza, gramos;
short vueltas, envios;
```

tipo_dato nombre_variable1= valor_inic_1 [,nombre_variable2= valor_inic_2] … ;

```
unsigned char c = 'A' , letra = 65;
int i = 2, j = 3;
double aceleracion = 3.0, masa = -5.44;
```

Constantes y literales

Modo C

```
#define PI 3.14159
perimetro = 2 * PI * radio;
```

Modo C++

```
const double pi = 3.14159;
perimetro = 2 * pi * radio;
pi = 4; // ERROR al compilar !!
```

Literales

```
3               //entero
true, false     //booleanos
'a'             //caracter
“Hola”          //String → en realidad “arreglo de 5 const char”
3.5             // punto flotante
2e-4            // punto flotante
'\n'            // line feed (10 o 0xA)
```

Cadenas de caracteres

- En su modo básico C++ no tiene un tipo de dato que permita manejar una cadena de caracteres (strings).
- Sin embargo C++ provee en su biblioteca estándar la **clase string**, la que usaremos como un tipo de datos más.

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

int main()
{
        cout << "Por favor ingrese su nombre: ";
        string nombre;
        cin >> nombre;
        cout << "Hola " << nombre << endl;
        return 0;
}
```

Secuencias de escape de caracteres

Código	Significado	Valor ASCII (Decimal)	Valor ASCII (Hexadecimal)
'\n'	Nueva línea (dependiente SO)	10	0x0A
'\r'	Retorno de carro	13	0x0D
'\t'	Tabulador (horizontal)	09	0x09
'\f'	Nueva página	12	0x0C
'\a'	Alerta (campana)	07	0x07
'\b'	Retroceder un caracter	08	0x08
'\v'	Tabulador (vertical)	11	0xB
'\\'	Barra invertida	92	0x5C
'\''	Comilla simple	39	0x27
'\"'	Comilla doble	34	0x22
'\ddd'	El caracter ASCII cuyo código sea ddd en octal		
'\xhh'	El caracter ASCII cuyo código sea nn en hexadecimal		
Nota: Este listado no es completo			

Operadores

- Aritméticos
  - Suma, resta → + , -
    - Con string el operador + concatena.
  - Multiplicación, división → * , /
  - Módulo → %
    - Solo para enteros.
- Relacionales
  - Menor, mayor → < , >
  - Menor o igual, mayor o igual → <= , >=
  - Igual, distinto → == , !=

Operadores

- Lógicos
  - No (Negación) → !
  - y → &&
  - o → ||
- Bits
  - y → &
  - o (inclusivo) → |
  - o (exclusivo) → ^
  - Desplazamiento a derecha → >>
  - Desplazamiento a izquierda → <<
  - Complemento a uno → ~

Operadores

- Incremento y decremento → ++ , --
  - Post: i++ (uso, luego incremento)

```
b = 2;
c = 3;
a = b++ * c;
//a vale 6 y b vale 3
```

  - Pre: ++i (incremento, luego uso)

```
b = 2;
c = 3;
a = ++b * c;
//a vale 9 y b vale 3
```

Operadores

- Asignación → `=`
  - Asocia de derecha a izquierda:
    - `a = b = c = 8;`
- Operar y asignar
  - Los operadores aritméticos y de manejo de bits pueden combinarse con la asignación
  - Ejemplo de un sumador
    - `total = total + dato;`
    - `total += dato;`
  - Genéricamente si el operador es X entonces
    - `a X= b;` ≡ `a = a X (b);`

Precedencia de los operadores

Precedencia	Operador	Asociatividad
1	`::`	Izquierda a Derecha

2	`++ --` *(postfijo)*
	`() [] . ->`
	`type() type{}`	

3	`++ --` *(prefijo)* `+ – ! ~ (type)`	Derecha a Izquierda
	`*` *(desreferenciar)* `&` *(dirección de)*	

	`sizeof`
	`new delete`
	`new[] delete[]`	

4	`.* ->*`	Izquierda a Derecha
5	`* / %`	
6	`+ –`	Izquierda a Derecha
7	`<< >>`	
8	`< <= > >=`	
9	`== !=`	
10	`&`	
11	`^`	
12	`|`	
13	`&&`	
14	`||`	

15	`?: = += –=`
	`*= /= %= <<=`
	`>>= &= ^= |=`
		Derecha a Izquierda

16	`throw`	
17	`,`	Izquierda a Derecha

Tomado de:
https://en.cppreference.com/w/cpp/language/operator_precedence

Licencia