



LENGUAJE DE PROGRAMACIÓN: C++

Autores:

Colectivo de Entrenadores ACM-ICPC UM

Enero / 2019

ENTRENAMIENTO PARA CONCURSANTES
ACM-ICPC DE LA UNIVERSIDAD DE
MATANZAS

Objetivos

- ▶ Familiarizar a los estudiantes con los lenguajes de programación más utilizados en los concursos ACM-ICPC.

Contenidos

Objetivos

Contenidos

Bibliografía

Lenguaje de programación

Lenguaje de programación: C++

Estructura de solución

Captura y salida de datos

Conclusiones

Bibliografía

- ▶ *Manual de preparación para concursantes ACM-ICPC de la Universidad de Matanzas.* Publicado en el Entorno Virtual de Aprendizaje de la Universidad de Matanzas (eva.umcc.cu). Sección *Bibliografía*.

Lenguajes de programación

- ▶ Un lenguaje de programación es un idioma artificial diseñado para expresar computaciones que pueden ser llevadas a cabo por máquinas como las computadoras.
- ▶ Es un conjunto de reglas, notaciones, símbolos y/o caracteres que permiten a un programador poder expresar el procesamiento de datos y sus estructuras en la computadora.

Lenguajes de programación: C++

- ▶ C++ es un lenguaje de programación diseñado a mediados de los años 1980 por Bjarne Stroustrup. La intención de su creación fue el extender al exitoso lenguaje de programación C con mecanismos que permitan la manipulación de objetos.
- ▶ Para desarrollar C++ podemos utilizar *IDE* como *CodeBlocks* y *Eclipse*.

Estructura de una solución

- ▶ La primera sección está destinada para la incluir todas bibliotecas propias del lenguaje que serán utilizadas en la solución del problema.

```
#include <iostream>
#include <math>
#include <algorithm>
```

Estructura de una solución

- ▶ La segunda sección de nuestra estructura va estar destinada a definir nuestras propias macros y definiciones de elementos muy útiles en pos de agilizar y mejorar la comprensión de nuestro código. Ejemplos de lo anterior expuesto son los siguientes fragmentos.

```
#define PI 3.14  
#define LL long long  
#define ENDL '\n'
```


Estructura de una solución

- ▶ La tercera sección la destinaremos a la declaración de variables y métodos auxiliares que utilizaremos en la solución de nuestro problema. Tener en cuenta que si se tiene dos métodos A, B y dentro de este último es invocado el método A la implementación de A tiene que estar por encima o primero que la implementación del método B.

```
int a,b;  
int suma(int _a ,int _b){  
    return _a+_b;  
}
```

Estructura de una solución

- En la última y cuarta sección va estar enmarcada dentro del programa principal.

```
int main(){  
    cout.setf(ios::fixed,ios::floatfield);  
    cout.precision(0);  
    ios_base::sync_with_stdio(0);  
    std::cin.tie(0);  
  
    cin>>a>>b;  
    cout<<suma(a,b)<<ENDL;  
  
    return 0;  
}
```

Estructura de una solución

```
#include <iostream>
#define ENDL '\n'

using namespace std;

int a,b;
int suma(int _a ,int  _b){
    return _a+_b;
}

int main(){
    cin>>a>>b;
    cout<<suma(a,b)<<ENDL;
    return 0;
}
```

Captura y salida de datos

- ▶ En múltiples ocasiones la diferencia entre una solución correcta y una incorrecta se puede definir en la captura o salida de datos por eso es importante conocer que recursos nos brinda C++ para realizar dichas acciones.
- ▶ C++ dispone de dos jerarquías de clases para las operaciones de entrada/salida: una de bajo nivel, y otra de alto nivel, con las clases: `istream`, `ostream` e `iostream`, que derivan de la clase `ios`.
- ▶ En nuestro caso utilizaremos la clase `iostream` de la siguiente manera:

```
#include <iostream>
```

Captura de datos

- ▶ Para capturar los datos en el lenguaje de programación C++ se puede utilizar dos variantes. La primera es la que propone su lenguaje de programación antecesor C (*Ver en el Manual*) y la segunda es la que propone el lenguaje de programación en sí.
- ▶ Al ejecutarse un programa en C++ uno de los flujos que se abren automáticamente es el de entrada.
- ▶ **cin**: entrada estándar (teclado).

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

Captura de datos

- ▶ **cin** no lee los espacios en una línea de hecho los usas al igual que el salto de línea y tabulaciones para delimitar los datos. En caso que te haga falta leer toda una línea entera incluyendo espacio se puede hacer de la siguiente manera.

```
string line  
getline(cin,line)
```

Tener en cuenta que el salto de línea no lo captura y si vuelves a leer otra cadena seguido solo vas a leer el salto de línea.

Salida de datos

- ▶ Para capturar los datos en el lenguaje de programación C++ se puede utilizar dos variantes. La primera es la que propone su lenguaje de programación antecesor C (*Ver en el Manual*) y la segunda es la que propone el lenguaje de programación en sí.
- ▶ Al ejecutarse un programa en C++ uno de los flujos que se abren automáticamente es el de salida.
- ▶ **cout**: salida estándar (pantalla).

```
cout << suma(a, b) << ENDL ;
```

Captura y salida de datos

- ▶ Cada flujo de C++ tiene asociados unos indicadores, que son unas variables miembro enum de tipo long que controlan el formato al activarse o desactivarse alguno de sus bits. Más detalles en el Manual de preparación.

```
cout.setf(ios::fixed,ios::floatfield);  
cout.precision(0);  
ios_base::sync_with_stdio(0);  
std::cin.tie(0);
```


Captura de datos

- ▶ El otro aspecto a tener en cuenta es el formato de entrada el cual puede presentar varias variantes para las cuales se debe estar preparado, a continuación se detallan los más comunes.
- ▶ Cuando me dan la cantidad de casos previamente.

```
int cases;  
cin>>cases;  
while(cases--)  
{  
    /* Desarrollo de la solucion */  
}
```

Captura de datos

- ▶ Cuando me dan la cantidad de casos previamente y necesito saber el caso que estoy analizando.

```
int cases;  
cin>>cases;  
for ( int i =1; i <= cases ; i ++)  
{  
    /* Desarrollo de la solucion */  
}
```

Captura de datos

- Se lee hasta que en la entrada se cumpla una determinada condición en cuanto a sus valores.

```
bool exits = false ;
int entrada ;
while(!exits)
{
    cin>>entrada ;
    if(entrada==0)
        exits = true ;
    else{
        /* Desarrollo de la solucion */
    }
}
```

Captura de datos

- ▶ Se lee hasta fin de fichero o se sabe en que momento termina la entrada.

```
int entrada2 ;  
while(cin>>entrada2)  
{  
    /* Desarrollo de la solucion */  
}
```

Conclusiones

- ▶ C++ independientemente de la versión que se use es el lenguaje más utilizado en la codificación de soluciones a problemas en concursos ACM-ICPC.
- ▶ Pero no significa que sea el perfecto, el mismo presenta problemas o dificultad con respecto a otros lenguajes en el trabajo de números grandes y en determinados ejercicio donde la complejidad está en la captura de los datos de entrada y las cadenas.
- ▶ A favor de C++ está su velocidad y uso de memoria.

UNIVERSIDAD DE MATANZAS

cosechando el saber

FIN