# Club de programación competitiva UNAM

Pu++

21 de marzo del 2020

# Conceptos básicos de c++

# Programando en c++ i

En programación competitiva, el diseño general de un programa es de la siguiente forma.

```
#include <bits/stdc++.h>
using namespace std;

int main()
{
    //aquí va su código
    return 0;
}
```

Nota: // indica un comentario y todo lo que sea seguido en esa misma línea no afectará en la ejecución del código. Puede ser omitido y su carácter es meramente informativo.

# Programando en c++ ii

El `include` al inicio del programa le indica a el compilador g++ que quieremos incluir toda la biblioteca estándar.

La línea del `using` declara que las clases y funciones de la biblioteca estándar las podemos usar directamente en nuestro código.

El método `main` es el punto de entrada de cualquier programa en C++. Es el punto en el que la ejecución del programa empieza.

# Entrada y Salida

# Entrada y Salida i

`Problema:` Dados dos números enteros a y b, imprimir b seguido de a.

`Input:` Se darán dos números enteros a, b.

`Output:` Debes imprimir los números en el siguiente formato: *b a* (nótese el espacio entre ellos).

**Ejemplo de entrada**

```
5 10

-200 52342
```

**Ejemplo de salida**

```
10 5

52342 -200
```

# Entrada y Salida ii

Lo primero que tenemos que hacer es leer los enteros a y b. Para esto, primero tenemos que **declararlos** (ésto es para asignarles una posición en memoria).

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int a, b;   //utilizamos int para declarar
                //que los números son enteros
    return 0;
}
```

# Entrada y Salida iii

Ahora vamos a **leer** las variables a y b de la entrada estándar (desde la terminal). Para leer, utilizamos la función **cin**.

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int a, b;
    cin >> a >> b;  //¡Es importante el órden en el
                    //que leemos las variables!
    return 0;
}
```

# Entrada y Salida iv

Finalmente, tenemos que imprimir en la salida estándar el resultado que nos piden. Para esto usamos la función cout.

```
#include <bits/stdc++.h>
using namespace std;

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

Estamos agregando **endl** para imprimir un salto de línea al final de la respuesta.

`¡A probar el programa!`

Recuerda que tienes que `compilar` el programa y luego correr el archivo `ejecutable`.

Posibles errores de compilación pueden venir de escribir los diamentes de forma incorrecta (« y »), o de haber olvidado un punto y coma.

# Trabajando con números

# Trabajando con números i

**Problema:** Dados dos números enteros a y b, aplicar su suma, resta, multiplicación, división y módulo.

**Input:** Se darán dos números enteros positivos a, b.

**Output:** Deberás imprimir el resultado de cada una de las operaciones en el formato indicado en el ejemplo de salida.

**Ejemplo de entrada**

```
151 5
```

**Ejemplo de salida**

```
SUMA: 156
RESTA : 146
MULTIPLICACION : 755
DIVISION : 30
MODULO: 1
```

# Trabajando con números ii

**Operación módulo:** a modulo b es el **residuo** de hacer la división $\frac{a}{b}$.

En particular, si: $a \pmod b = 0$, entonces $a$ es múltiplo de $b$.

**Ejemplos:**

- $16 \pmod 2 = 0$
- $17 \pmod 3 = 2$
- $29 \pmod{10} = 9$
- $30 \pmod 5 = 0$

# Trabajando con números iii

Como ya sabemos, lo primero es leer los enteros a y b.

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int a, b;
    cin >> a >> b;
    return 0;
}
```

# Trabajando con números iv

Ahora aplicar las operaciones correspondientes en el formato.

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int a, b;
    cin >> a >> b;
    cout << "SUMA: " << a+b << endl;
    cout << "RESTA: " << a-b << endl;
    cout << "MULTIPLICACION: " << a*b <<endl;
    cout << "DIVISION: " << a/b << endl;
    cout << "MODULO: " << a%b << endl;
    return 0;
}
```

# Trabajando con números v

`Consideraciones importantes:`

- ¡Tengan cuidado de **no** aplicar **`divisiones o módulos`** con el **`número 0`**! Éstos errores son comunes, y un juez rechazará el problema dando el veredicto **`Runtime Error`**.
- En el ejemplo anterior, el resultado $\frac{151}{5} = 30$ es correcto, a pesar de que $30 * 5 \neq 151$. Esto es porque al hacer operaciones entre números enteros, también el resultado será un número entero, y se descartará la fracción decimal. En otras palabaras, la **`división entre enteros`** es equivalente a la **`división con función piso`**.

# Otros tipos de números

# Tipos de números i

Además de `int`, contamos con otros tipos de datos para representar números. Veamos cuándo utilizar cada uno.

- `int:` es un entero de `32 bits`. Esto significa que puede guardar valores entre $-2^{31}$ y $2^{31} - 1$ (que es más o menos $2 * 10^9$). ¿Qué pasa si al programa de la sección anterior le damos como entrada $a = b = 10^9$?
- `long long int:` es un entero de `64 bits`. Esto significa que puede guardar valores entre $-2^{63}$ y $2^{63} - 1$ (que es más o menos $9 * 10^{18}$).
- `double:` es un número de precisión que utilizamos para representar `números con parte fraccionaria`.

# Tipos de números ii

`Problema:` Dados dos números a y b, elevarlos al cuadrado.

`Input:` Se darán dos números a y b. a es un número entero tal que $0 \leq a \leq 10^9$ y b un número real tal que $0.0 \leq b \leq 100.0$.

`Output:` Deberás imprimir el resultado de elevar a al cuadrado y b al cuadrado, separados por un espacio.

| Ejemplo de entrada | Ejemplo de salida |
| --- | --- |
| `1000000000 5.5` | `1000000000000000000 30.25` |

# Tipos de números iii

En este caso, un int ya no es suficiente para guardar la información.

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    long long a;
    double b;
    cin >> a >> b;
    cout << a*a << " " << b*b << endl;
    return 0;
}
```

# Estructuras de Control

# Ciclos i

**Problema:** Dado un entero positivo n, escribir la oración *'Amo la programacion competitiva'* n veces.

**Input:** Se dará un número entero positivo n.

**Output:** Deberás imprimir la oración *'Amo la programacion competitiva'* n veces.

**Ejemplo de entrada**

```
5
```

**Ejemplo de salida**

```
Amo la programacion competitiva
Amo la programacion competitiva
Amo la programacion competitiva
Amo la programacion competitiva
Amo la programacion competitiva
```

# Ciclos ii

Para resolver éste tipo de problemas necesitaremos de ciclos.
Veremos dos estructuras para eso:

- While
- For

Amabas pueden ser usadas de forma equivalente, pero dependiendo de la situación muchas veces una es más cómoda de usar que la otra.

# While i

El While es una estructura de control que nos permite ejecutar instrucciones mientras una condición dada se siga cumpliendo.

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;
    while (n > 0){
       //instrucciones a ejecutar
    }
    return 0;
}
```

No ejecutar código, se entrará en un bucle infinito si n es mayor a 0.

# While ii

En el caso anterior nuestra condición a cumplirse es `n>0.`

Nótese que si la condición llega a cumplirse, y no se modifica las variables involucradas, se seguirá cumpliendo la condición infinitamente, quedando atrapado en el while.

Por ello es importante modificar las variables de la condición puesta `dentro del while`.

# While iii

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;
    while (n > 0){
        cout << "Amo la programacion competitiva" << endl;
        n-- //modificacion de variable de condicion
    }
    return 0;
}
```

`n--` es una forma corta de escribir `n=n-1;`
El código dentro del While se ejecutará n veces y se dentendrá cuando n valga 0.

# For i

El For es una estructura de control que nos sirve para repetir procesos de una forma más puntual.

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;
    for (int i = 0 ; i < n; i++){
       //instrucciones a ejecutar
    }
    return 0;
}
```

# For ii

Entre los parentesis posteriores a la palabra reservada `for` se encuntran los elementos que indican cuantas veces se repetirá las intrucciones entre las llaves. Éstos elementos son 3 y se separan con un punto y coma:

- `Variable de control (int i = 0):` Declaramos la variable de control, en este caso llamada i que es de tipo entero (int). La variable puede ser del tipo que queramos y llevar el nombre que deseemos. Se le asigna un valor inicial que puede ser cualquier valor correspondiente al tipo de dato.
- `Valor final (i < n):` El valor final especifica hasta que valor llegará nuestra variable de control. Cuando la clausula especificada no sea cierta saldremos del ciclo. En nuestro ejemplo, cuando i deje de ser menor a n se dejara de ejecutar las instrucciones en el For.

# For iii

- `Tamaño del paso (i++):` El tamaño del paso especifica la cantidad deseada del aumento de la variable de control. i++ es un forma corta de escribir: i = i+1;
  En nuestro caso hacemos un incremento de uno en nuestra variable de control.

Ya sabiendo lo anterior, podemos deducir que nuestro for se ejecutará n veces.

# For iv

¿Puedes deducir cuántas veces se ejecutará el siguiente for?

```
for (int i = 20 ; i < 70; i = i+10){
    cout << "Hola" << endl;
}
```

¡Así es, se ejecutará 5 veces!
El valor de i pasará por los valores: 20, 30, 40, 50, 60 y 70.
Cunado i valga 70, dado que 70 no es menor a 70, ya no se seguirá ejecuntando el for. Por lo tanto, solo se habrá ejecutado el for con los primeros 5 valores.

# For v

Solución a problema:

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;
    for (int i = 0 ; i < n; i++){
       cout << "Amo la programacion competitiva" << endl;
    }
    return 0;
}
```

# If Else i

En ocasiones, queremos procesar un conjunto de datos de una forma particular si éstos cumplen cierto requisito. Por ejemplo, si queremos saber cuáles números entre 1 y n son pares o impares.

# If Else ii

`Problema:` Dado un entero positivo n, imprimir los números pares entre el 1 y n.

`Input:` Se dará un número entero positivo n.

`Output:` Deberás imprimir n/2 líneas, donde cada una indica un numero par entre el 1 y n, siguiendo el formato del ejemplo.

**Ejemplo de entrada**

```
5
```

**Ejemplo de salida**

```
El numero 2 es par
El numero 4 es par
```

# If Else iii

En este problema, debemos leer el entero n. Luego, recorrer los números del 1 al n e imprimirlo si es par.

```
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;
    for(int i = 1; i <= n; i++){
        //aquí va su código
    }
    return 0;
}
```

# If Else iv

```
if(expresion){
    //procedimiento
}
```

La sentencia if evalúa la expresión dentro del paréntesis. Si expresion se evalúa como verdadera, el procedimiento dentro del cuerpo del if se ejecuta. Si se evalúa como falsa, el procedimiento dentro del cuerpo del if no se ejecuta.

¿Cómo podemos ver si un número es par?

¡Si su residuo módulo dos es cero!

# If Else v

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;
    for(int i = 1; i <= n; i++){
        if(i%2 == 0){
            cout <<"El numero "<< i <<" es par"<< endl;
        }
    }
    return 0;
}
```

# If Else vi

Nota: en la expresión del if utilizamos == en lugar de =. Esto es porque el operador = hace una asignación de variable con valor, mientras que == verifica que ambos valores sean iguales.

# If Else vii

`Problema:` Dado un entero positivo n, imprimir la paridad de los números entre el 1 y n.

`Input:` Se dará un número entero positivo n.

`Output:` Deberás imprimir n líneas, donde se indican las paridades de los números entre el 1 y n.

**Ejemplo de entrada**

```
5
```

**Ejemplo de salida**

```
El numero 1 es impar
El numero 2 es par
El numero 3 es impar
El numero 4 es par
El numero 5 es impar
```

# If Else viii

```
if(expresion){
    //procedimiento
}
else{
    //otro procedimiento
}
```

Cuando utilizamos if...else, se ejecuta alguno de dos distintos bloques de código, dependiendo de la evaluación de expresion. Si se evaluó como verdadera, se ejecuta el cuerpo del if. En otro caso, se ejecuta el cuerpo del else.

# If Else ix

```
if(expresion1){
    //procedimiento1
}
else if(expresion2){
    //procedimiento2
}
.
.

else{
    //otro procedimiento
}
```

# If Else x

También es posible anidar varios if...else. En estos casos, se evalúan las distintas expresiones en orden para ver cual se evalúa como verdadera, y ejecutar el procedimiento correspondiente.

# If Else xi

```cpp
#include <bits/stdc++.h>
using namespace std;

int main(){
    int n;
    cin >> n;
    for(int i = 1 ; i <= n; i++){
        if(i%2 == 0){
            cout << "El numero " << i << " es par" << endl;
        }else{
            cout << "El numero " << i << " es impar" << endl;
        }
    }
    return 0;
}
```

# Problemas

# Problemas i

Ahora... ¡a practicar!

- Ingresa a omegaup.com
- Crea una cuenta
- Practica con los problemas de la siguiente liga https://omegaup.com/arena/pumasmas/problems

Hasta la próxima ☺