Primer programa en C++

Introduciremos el siguiente texto en Geany .

```
1   // Escribir tres textos en C++
2   #include <iostream>
3   int main ()
4   {
5       std::cout << "Hola";
6       std::cout << " ";
7       std::cout << "Alumno";
8       return 0;
9   }
10
```

Con la orden **#include <iostream>** , estamos diciéndole al compilador que utilice la librería de utilidades de entrada / salida por pantalla , esto permite poder utilizar la instrucción **cout** (salida por pantalla).

El siguiente programa es realizar la división entre dos números enteros :

int x,y;

Con la orden **cin >>** , pedimos los números al usuario y con **cout <<** mostramos la información.

A partir de ahora pondremos siempre **(using namespace std)** con lo que nos evitaremos escribir el std::

```
1   // Division de dos enteros
2
3   #include <iostream>
4   using namespace std;
5
6   int main ()
7   {
8       int x, y;
9       cout << "Introduce un numero: ";
10      cin >> x;
11      cout << "Introduce otro: ";
12      cin >> y;
13      cout << "Su division es " << x/y;
14      return 0;
15  }
```

Actividades :

1. Utilizando como base el ejemplo anterior , modifícalo para realizar un producto entre dos números.

2. Utiliza la librería <cmath> y obtener la raiz cuadrada de un numero facilitado por el usuario .
Sqrt()

3. Crea un programa que pida al usuario una cantidad de millas terrestres y calcule su equivalente en KM (1 milla = 1,609 km) Necesitaras usar datos float o doble.

4. Realiza un programa que pida 5 números reales al usuario y calcule su media.

5. Si se ingresan E euros en un banco , con un interés R durante N periodos de tiempo, el dinero que se obtendrá finalmente vendrá dado por la formula : $E*(1+R)^N$. Crea un programa que lo calcule a partir de los datos introducidos por el usuario.

Necesitaras las variables :

E , euros
R , el interés
N , los años

Ten en cuenta que R lo tendrás que dividir por 100 antes de utilizar la formula.

Toma de decisiones : (IF….ELSE)

```cpp
// Número par?

#include <iostream>

using namespace std;

int main ()
{
    int n;
    cout << "Dime un numero: ";
    cin >> n;
    if (n % 2 == 0)
        cout << "Es par" << endl;
    else
        cout << "Es impar" << endl;
    return 0;
}

```

Disponemos de una instrucción **if .. else** , que evalua mediante la operación módulo si el numero es par o no. Las comparaciones siempre se realizan con doble = .

La operación % (modulo) calcula el resto de la división.

Operador	Operación
<	Menor que
>	Mayor que
<=	Menor o igual que
>=	Mayor o igual que
!=	No igual a (distinto de)

Tabla 2. Relación de operadores y sus significados.

```cpp
// Comprobar el valor de una variable
#include <iostream>
using namespace std;

int main()
{
  int numero;

  cout << "Escribe un numero: ";
  cin >> numero;
  if (numero == 0)
       cout << "Has escrito cero.";

  return 0;
}
```

Actividades :

1. Realiza un programa que pida dos números enteros y diga cual es mayor.

2. Elabora un programa que pida dos números al usuario. Si el segundo no es cero , mostrará la división entre ambos; en caso contrario debe aparecer un mensaje de error.

3. Crea un programa que pida dos números e indique si el primero es múltiplo del segundo.

Encadenar condiciones

Las condiciones se pueden encadenar con «y», «o», «no», etc., tal como se indica en la tabla 3. Así, se pueden escribir expresiones como:

```
// Condiciones múltiples
#include <iostream>
using namespace std;

int main()
{
  int numero;

  cout << "Escribe un numero: ";
  cin >> numero;
  if (!(numero == 0))
    cout << "No es cero.";
  if ((numero == 2)||(numero == 3))
    cout << "Es dos o tres.";
  if ((numero >= 2) && (numero <= 7))
    cout << "Esta entre 2 y 7 (incluidos).";

    return 0;

}
```

Operador	Significado
&&	Y
||	O
!	No

Tabla 3. Relación de operadores y sus significados.

```
Escribe un numero: 3
No es cero. Es dos o tres. Esta entre 2 y 7
(incluidos).
```

Actividades :

1.Crea un programa que pida al usuario dos números , los compruebe y diga :
“Uno de ellos es positivo”, “Los dos son positivos” o “Ninguno lo es”.

2. Crea un programa que pida tres números reales e indique cual de los 3 es mayor y muestre su valor.

3. Crea un programa que indique dados dos números si uno es múltiplo del otro.

SWITCH

Se utiliza para evaluar de forma rápida varios valores de una variable que se van recogiendo en los **CASE** , cada uno de los cuales debe de terminar con un break.

```
// Condiciones con switch
#include <iostream>
using namespace std;

int main()
{
  int numero;

  cout << "Introduce un numero del 1 al 5: ";
  cin >> numero;
  switch (numero)
  {
    case 1: cout << "Uno";
      break;
    case 2: cout << "Dos";
      break;
    case 3: cout << "Tres";
      break;
    case 4: cout << "Cuatro";
      break;
    case 5: cout << "Cinco";
      break;
    default: cout << "Valor incorrecto!";
  }
    return 0;
}
```

```
Introduce un numero del 1 al 5: 4
Cuatro
```

Cuando varios casos tengan que mostrar un mismo resultado, se puede dejar que el control «caiga» de uno a otro, dejando vacío el primero de ellos. Por ejemplo, para el sistema de calificaciones escolares en España, en el que tanto un 7 como un 8 son un notable, se podría hacer lo siguiente:

```
case 7:
  case 8:
    cout << "Notable";
      break;
```

Actividades:

1.Elabora un programa que pida números del 1 al 10 y muestre la nota correspondiente.

2.Crea un programa que pida un número del 1 al 100 y diga si es múltiplo de 3 o no .
(Utiliza un Switch y agrupa casos).

BUCLES

A veces una condición se debe de comprobar de forma repetitiva hasta que se cumpla una determinada condición. Para ello utilizaremos el **while**

```
// Repetición con while: pedir un número positivo
#include <iostream>
using namespace std;

int main()
{
  int numero;

  cout << "Escribe un numero positivo: ";
  cin >> numero;
  while (numero <= 0)
  {
    cout << "Ese numero no es positivo." << endl;
    cout << "Escribe otro numero positivo: ";
    cin >> numero;
  }
    return 0;

}
```

Si queremos comprobar la condición tras dar una primera pasada utilizaremos **do-while**

```
// Repetición con do-while: pedir un número positivo
#include <iostream>
using namespace std;

int main()
{
  int numero;

  do
  {
    cout << "Escribe un numero positivo: ";
    cin >> numero;
    if (numero <= 0)
      cout << "Ese numero no es positivo." << endl;
  }
  while (numero <= 0);

  return 0;

}
```

Actividades :

1. Crea un programa que pida al usuario su password tantas veces como sea necesario hasta que introduzca una valida. (Ej pass=7890)

2. Modifica el programa anterior para añadirle también la comprobación de usuario.

3. Crea un programa que pida números positivos al usuario y que calcule su suma , el programa acabará cuando introduzcamos un 0.

4. Elabora un programa donde el usuario intente adivinar un número del 1 al 100 , dispondrá de 6 intentos y habrá que indicarle si el numero introducido es mayor o menor que el numero a adivinar. El programa finalizará cuando se superen los 6 intentos o se adivine el número.

FOR

El bucle for se utiliza para iteraciones ya determinadas.

```
// Contador con for: operación incremento
#include <iostream>
using namespace std;
int main()
{
  int numero;
  for(numero = 1; numero <= 10; numero++)
    cout << numero << " ";
    return 0;
}
```

```
#include <iostream>
using namespace std;
int main()
{
  int tabla, numero;
  for (tabla=1; tabla<=5; tabla++)
  {
    for (numero=1; numero<=10; numero++)
      cout << tabla <<  " por " <<  numero
        << " es " << tabla*numero << endl;
      cout << endl;
  }
  return 0;
}
```

Crea un programa donde le muestres al usuario la tabla de multiplicar que el te indique con un número.