## Grado en Ingeniería Información

# PROGRAMACIÓN I

## Sesión 3

**Curso 2022-2023**

Marta N. Gómez

UNIVERSIDAD NEBRIJA

UNIVERSIDAD NEBRIJA

Programación en C++11.

- Estructura de un programa en C++11.
- **Instrucciones de Entrada y Salida: "Hola Mundo".**
- Tipos de datos y variables.
- Manipulación de datos:
  - Sentencias condicionales: if, switch-case.
  - Control de flujo: for, while.


**ÍNDICE**

# Entrada y Salida de datos

- La biblioteca estándar que se debe incluir es:

```
#include  <iostream>
```

- Incluye los flujos y operadores para manejar teclado y pantalla:

```
//  operador de extracción desde el flujo cin hacia variables
std::cin >>             //  Leer datos de teclado

//  operador de inserción al flujo cout
std::cout <<            //  Mostrar datos en pantalla
```

# Entrada de datos (std::cin)

- Permite **solicitar por teclado** uno o varios datos, del mismo tipo o de tipos diferentes, y asociarlos a una o varias variables.
- Ejemplo:

```
std::cin >> nom_variable;

std::cin >> nom_var1 >> nom_var2 >> …;
```

# Entrada de datos (std::cout)

- Permite **mostrar por pantalla** uno o varios mensajes/datos del mismo tipo o de tipos diferentes.

- Ejemplo:

```
std::cout  <<   "\n\n\tTexto " << nom_var1 << " " <<
nom_var2 << endl;
```

| Código | Descripción | Código | Descripción |
|---|---|---|---|
| **\n** | Salto de línea | \” | Escribir doble comilla |
| **\t** | Tabulación | \\ | Escribir barra inclinada |
| **endl** | Salto de línea | | |

# Espacio de nombres o *namespace*

- Zonas del programa donde son válidos los identificadores/etiquetas de variables / tipos de datos / funciones / estructuras / etc.
- Resuelven problemas de redefinición que se pueden presentar en desarrollos grandes.
- Ejemplo:

```
#include <iostream>
using namespace std;
int main()
{  int aa {2022};                    //   int i=2022;
   cout << "El valor de aa es: " << aa << endl;
```

UNIVERSIDAD NEBRIJA

Programación en C++11.

- Estructura de un programa en C++11.
- Instrucciones de Entrada y Salida: “Hola Mundo”.
- **Tipos de datos y variables.**
- Manipulación de datos:
  - Sentencias condicionales: if, switch-case.
  - Control de flujo: for, while.


ÍNDICE

# Tipos de datos: variables

- Es una posición de memoria a la que se puede acceder a través de un nombre (***identificador***).
- Se caracteriza por: **posición de memoria, tipo de dato y nombre**.
- Está obligatoriamente asociada a un **tipo de dato**.
- Tipos:
  - **Var. Globales:** declaradas para tener acceso a ellas desde el ámbito de cualquier **bloque** del programa.
  - **Var. Locales:** declaradas para tener acceso a ellas dentro del ámbito de un determinado **bloque** (método, función, bucle, etc.). Existen durante la ejecución de dicho bloque.

# Tipos de datos: variables

- Declaración de variables sin inicializar:

  ***tipo_dato*** nombre_var;

- Definir **varias variables** del mismo tipo a la vez:

  ***tipo_dato*** nom_var1 [, nom_var2,…, nom_varN];

- Definir e **inicializar** una variable:

  ***tipo_dato*** nom_var1 = valor; //C++ “antiguo”

  ***tipo_dato*** nom_var1 {valor}; //C++ “moderno”

```
#include <iostream>

int main()

{   std::cout << "Hola Mundo" << std::endl;

    //  Esto es un comentario

    int i {666};            //   int i=666;

    return 0;

}
```

```
#include <iostream>

int main()

{   std::cout << "Hola Mundo" << std::endl;

    //  Esto es un comentario

    int i {2.5};  //   int i=2.5;     Error de compilación

    return 0;

}
```

```
#include <iostream>

int main()

{   //  Mostrar el año actual

    int aa {2022};        //   int i=2022;

    std::cout << “El valor de aa es: " << aa << std::endl;

    return 0;

}
```

```
#include <iostream>

int main()

{   //  Mostrar el año actual

    int aa {2022};        //   int i=2022;

    std::cout << “El valor de aa es: " << aa << std::endl;

    aa = 2023; //  Actualizamos al año que viene

    std::cout << “Ahora el valor de aa es: " << aa << std::endl;

    return 0;

}
```

# Tipos de datos: variables

No se puede usar una variable sin declarar:

```
std::cout << aa << std::endl;        // Error de compilación

int aa = 2022;
```

# Tipos de datos simples

- Numéricos:
  - Enteros
  - Reales
- Otros:
  - Lógicos
  - Caracteres

# Tipos de datos simples

- Numéricos:

  Enteros

  - Datos numéricos sin parte decimal.
  - Modificables por: ***signed*** o ***unsigned, long*****.**

| Tipo | Descripción | Valor mínimo | Valor máximo |
|---|---|---|---|
| **short** | nº entero 16 bits | -32768 | 32767 |
| **int** | nº entero 32 bits | -2147483648 | 2147483647 |

# Tipos de datos simples

- Numéricos:

  Reales

  - Datos numéricos con parte decimal.
  - ***double*** puede ser modificado por ***long*** para ampliar el **rango de valores.**

| Tipo | Descripción | Valor mínimo | Valor máximo |
|---|---|---|---|
| **float** | nº decimal 32 bits | 1.2E-38 | 3.4E+38 |
| **double** | nº decimal 64 bits | 2.3E-308 | 1.7E+308 |

# Tipos de datos simples

- Otros:

## Lógicos

| Operador | Significado |
|---|---|
| ! | Not |
| && | And |
| \|\| | Or |

| A | B | !A | A && B | A \|\| B |
|---|---|---|---|---|
| *true* | *true* | *false* | *true* | *true* |
| *true* | *false* | *false* | *false* | *true* |
| *false* | *true* | *true* | *false* | *true* |
| *false* | *false* | *true* | *false* | *false* |

## Caracteres

- Los **caracteres** individuales se encierran entre **comillas simples** ('a').

| Tipo | Descripción |
|---|---|
| **bool** | Almacena 2 valores true o false (0 o 1), pero ocupa 8 bits |
| **char** | Almacena caracteres (ocupa 8 bits) [código ASCII] |

# Tipos de datos simples

- Modificadores:

***signed:*** tipo con signo (no vale para *float*, *double* y *bool*).

***unsigned***: tipo sin signo (no vale para *float*, *double* y *bool*).

***long***: tipo largo (vale para *int*, *double y long* (*long long*)

***modificador*** *tipo_dato* nom_var;

# Tipos de datos - estructuras

Se crean tipos de datos a partir de otros tipos:

- Datos simples
- Anidando otras estructuras

# Tipos de datos - estructuras

Se crean tipos de datos a partir de otros tipos:

- Datos simples

```
struct Persona {
    int edad;
    float altura;
    float peso;
};
```

# Tipos de datos - estructuras

Se crean tipos de datos a partir de otros tipos:

- Declaración de una variable ***Persona***

```
Persona   unapersona;
```

# Tipos de datos - estructuras

Se crean tipos de datos a partir de otros tipos:

- Asociar valores a los **campos** de la variable ***Persona***

```
unapersona.edad = 18;          // campo int

unapersona.altura = 180.8;     // campo float

unapersona.peso = 75.9;        // campo float
```

# Tipos de datos - estructuras

Se crean tipos de datos a partir de otros tipos:

- Inicializar la variable ***Persona***

```
Persona   unapersona {18, 185.3, 78.6};
```

No recomendable porque no se explicita a qué **campo** se está asociando cada valor.

# Constantes

Las constantes se declaran anteponiendo la palabra ***const***

```
const  tipo_dato   nom_constante = valor;   //C++ "antiguo"

const  tipo_dato   nom_constante {valor};   //C++ "moderno"
```

Ejemplos:

```
const float  PI = 3.1416;              //C++ "antiguo"

const char  VOCAL_a {'a'};             //C++ "moderno"
```

# Constantes

Errores de compilación:

```
const float PI;
```

```
const char VOCAL-a {'a‘};
```

```
VOCAL_a = ‘A';
```

# Asignación/Asociación de valor a una variable

Siempre se realiza a través del operador **=**

Asigna el valor de la **expresión derecha** a la **variable situada a la izquierda** de la instrucción.

nom_variable = expresión;

**expresión** puede ser otra variable, una constante o una operación entre variables y constantes.

Ejemplos:

```
float pi {3.14};
pi = 3.1416;        //  se cambia el valor de pi, ahora vale 3.1416
```

# Asignación/Asociación de valor a una variable

Errores de compilación, variable sin inicializar:

```
float pi;

std::cout << pi << std::endl;        // valor impredecible

pi = 3.1416;
```

# Asignación/Asociación de valor a una variable

**Operadores:**

=  +=  -=  *=  /=  %=

| Instrucción Abreviada | Instrucción Equivalente | Significado |
|---|---|---|
| | m = n; | Asigna el valor de **n** a **m** |
| m += n; | m = m + n; | Suma **m** y **n** y se asigna a **m** |
| m -= n; | m = m – n; | Resta **n** a **m** y se asigna a **m** |
| m *= n; | m = m * n; | Multiplica **m** por **n** y se asigna a **m** |
| m /= n; | m = m / n; | Divide **m** entre **n** y se asigna a **m** |
| m %= n; | m = m % n; | Calcula el resto de la división entera y se asigna a **m** |

# *Incremento/Decremento*

| | |
|---|---|
| **A++**<br>**++A** | **Incrementa en 1 el valor de A (A= A+1)** |
| **A--**<br>**--A** | **Disminuye en 1 el valor de A (A= A-1)** |

```
x=1;
A = ++x; // preincremento: A valdrá 2, x valdrá 2
```

```
x=1;
A = x++; // postincremento: A valdrá 1, x valdrá 2
```

UNIVERSIDAD NEBRIJA

Programación en C++11.

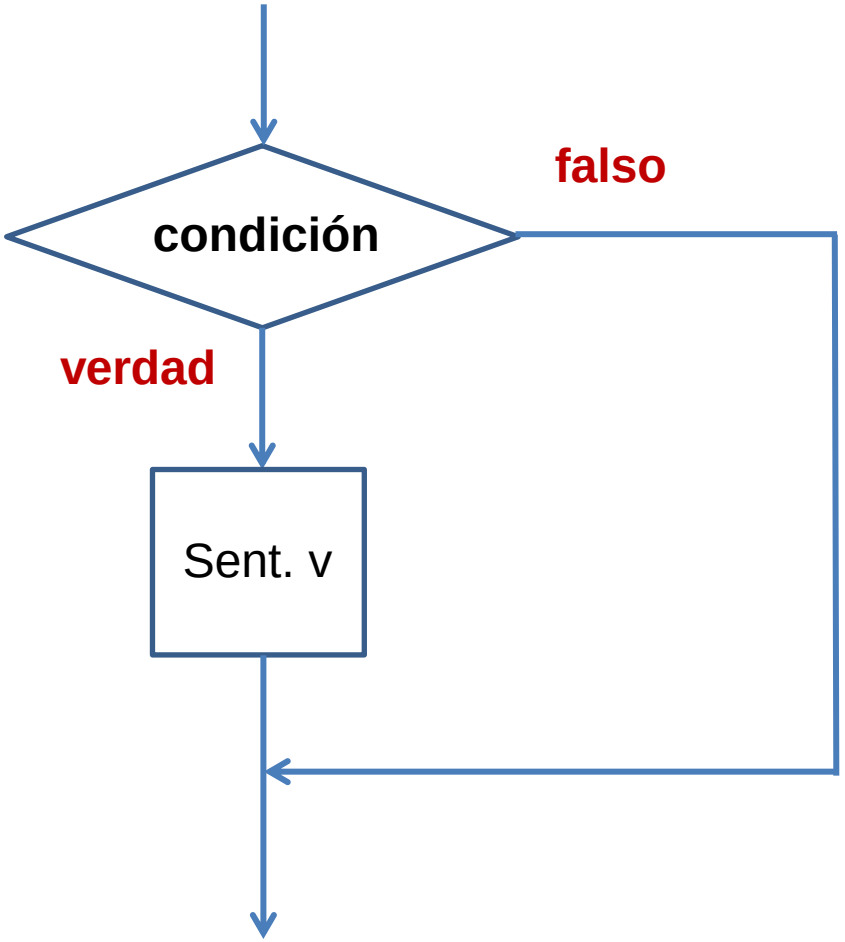
- Estructura de un programa en C++11.
- Instrucciones de Entrada y Salida: “Hola Mundo”.
- Tipos de datos y variables.
- **Manipulación de datos:**
  - **Sentencias condicionales: if, switch-case.**
  - Control de flujo: for, while.


**ÍNDICE**

# Sentencias Condicionales - if

```
if (condición)
   sentencia;
```

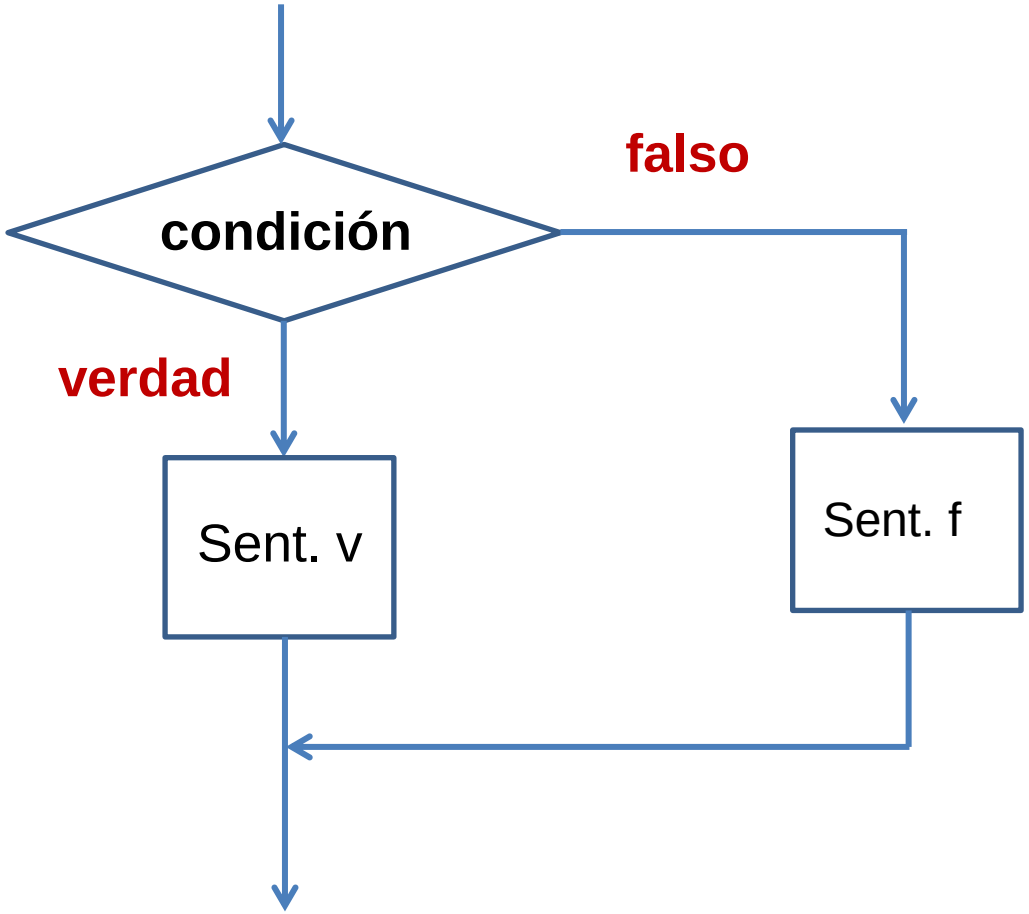
```
if (condición)
{   sentencia_1;
    sentencia_2;
    ...
    sentencia_n;    }
```



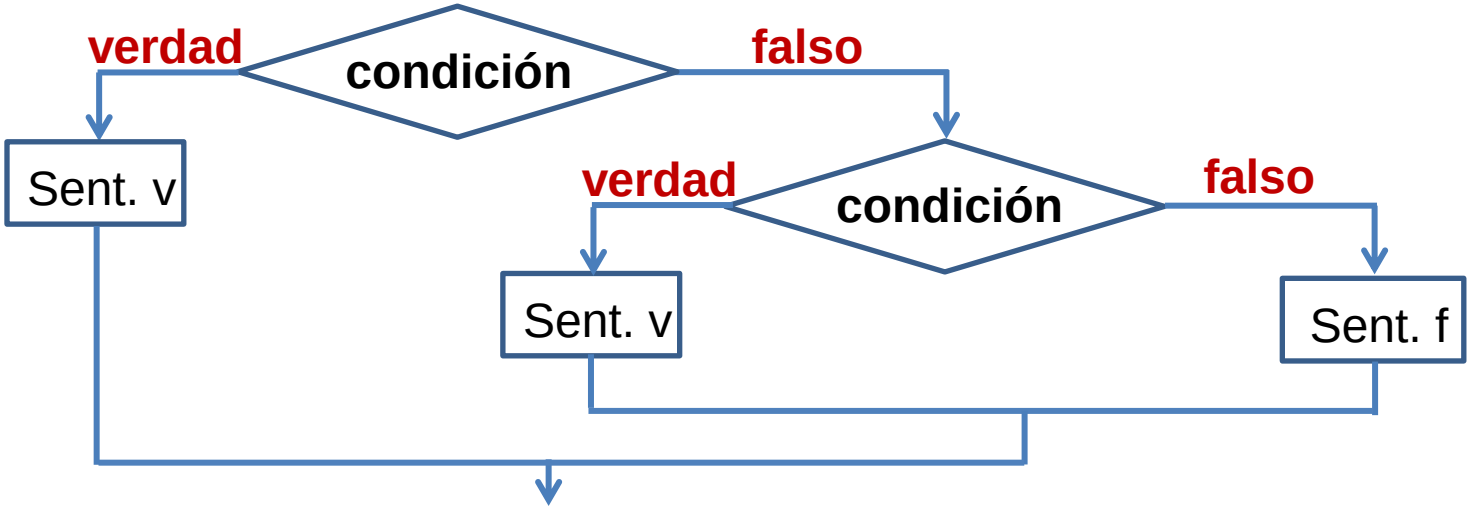
# Sentencias Condicionales - if

```
if (condición)
   sentencia1;
else
    sentencia2;
```

```
if (condición)
{   sentencia11;
    sentencia12;...  }
else
{   sentencia21;
    sentencia22;...  }
```



# Sentencias Condicionales - if




```
if (condición1)
   sentencia1;
else if (condición2)
        sentencia2;
     else sentencia3;
```

```
if (condición1)
{  sentencia11;
   sentencia12;... }
else if (condición2)
        {  sentencia21;
           sentencia22;... }
else {  sentencia31;
        sentencia32;... }
```

UNIVERSIDAD NEBRIJA

# Sentencias Condicionales - operadores de comparación

| Operador | Significado |
|---|---|
| == | Igualdad |
| != | Distinto |
| > | Mayor que |
| < | Menor que |
| >= | Mayor o igual que |
| <= | Menor o igual que |

# Sentencias Condicionales - operadores lógicos

| c1 | c2 | !c1 | c1 && c2 | c1 \|\| c2 |
|---|---|---|---|---|
| *true* | *true* | *false* | *true* | *true* |
| *true* | *false* | *false* | *false* | *true* |
| *false* | *true* | *true* | *false* | *true* |
| *false* | *false* | *true* | *false* | *false* |

# Sentencias Condicionales - if

```
#include <iostream>
using namespace std;
int main()
{  int i;
    cout << "Escriba un numero y pulse 'intro'" << endl;
    cin >> i;
    if (i > 5)   cout << "El numero es mayor que 5" << endl;
   else  if (i < 5)   cout << " El numero es menor que 5" << endl;
          else  cout << " El numero es igual a 5" << endl;

   cout << "Escriba un numero y pulse 'intro'" << endl;
   cin >> i;
   if (i < 10)
       if (i > 5)  cout << " El numero esta entre 5 y 10" << endl;
       else  cout << " El numero es menor o igual que 5" << endl;
   else  cout << " El numero es mayor o igual que 10" << endl;
 system ("pause");
 return 0;
}
```

## Grado en Ingeniería Información

# PROGRAMACIÓN I

## Sesión 4

**Curso 2022-2023**

Marta N. Gómez

UNIVERSIDAD NEBRIJA

UNIVERSIDAD NEBRIJA

Programación en C++11.

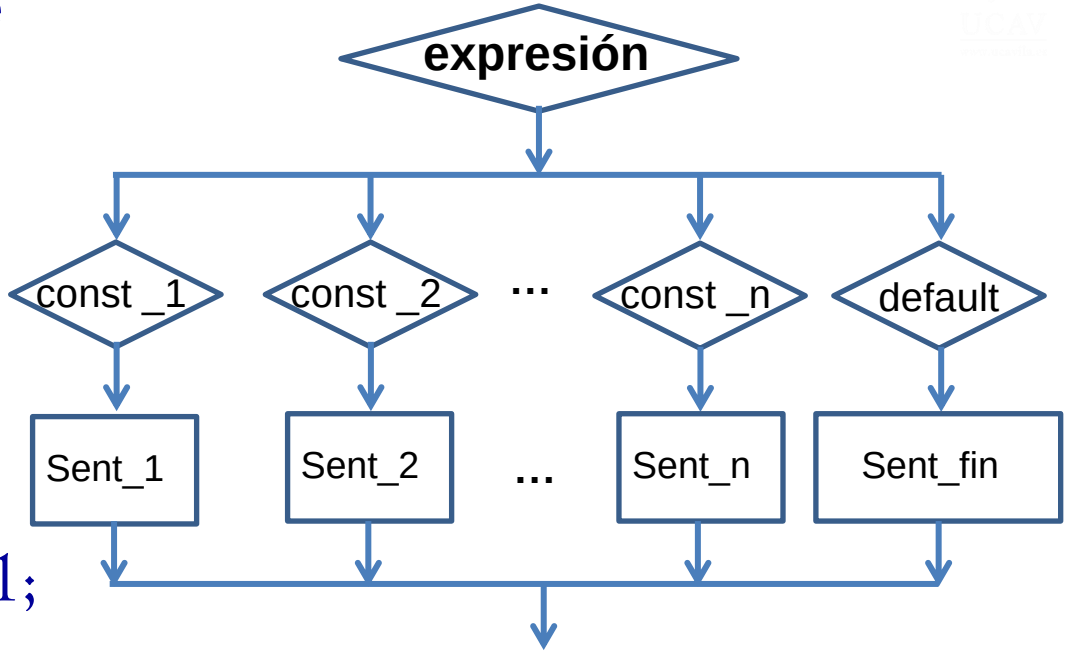
- Estructura de un programa en C++11.
- Instrucciones de Entrada y Salida: “Hola Mundo”.
- Tipos de datos y variables.
- **Manipulación de datos:**
  - **Sentencias condicionales: if, switch-case.**
  - Control de flujo: for, while.


**ÍNDICE**

# Sentencias Condicionales - switch-case

El resultado de la expresión debe ser de ***tipo ordinal***:

- tipo entero
- carácter
- lógico




```
switch (expresión)
{    case const_1:   sentencias_1;
                     break;
     case const_2:   sentencias_2;
                     break;
     case const_3:
     case const_4:   sentencias_3_4;
                     break;
     [default:   sentencias_fin;]
}
```

# Sentencias Condicionales - switch-case

El resultado de la expresión debe ser de ***tipo ordinal***:

- Entero (***int***)
- Carácter (***char***)
- Lógico (***bool***)

La sección ***default*** no es obligatoria. Sección donde se va cuando no hay un case donde saltar.

Los comandos ***break*** no son obligatorios, se podría necesitar agrupar varias condiciones.

Si el ***break*** no se incluye dentro del ***case***, se continua la ejecución del siguiente case sin salir del bloque ***switch-case***.

No se pueden declarar variables dentro del ***switch-case*** (error de compilación).

# Sentencias Condicionales - switch-case

```
#include <iostream>
using namespace std;
int main()
{  int num;
   cout  << " Introduzca un numero del 1 al 10: ";
   cin >> num;
  switch (num)
      {  case 1 :   case 3 :     case 5 :
              cout << " El numero es impar " << endl;
              break;
        case 2 :
        case 4 :
        case 6 :  cout << " El numero es par" << endl;
                  break;
        case 7 :    case 8 :
        case 9 :    case 10 : cout << " El numero es mayor que 6" << endl;
                              break;
        default : cout << " El numero no está entre 1 y 10" << endl;
      }
  return 0;
}
```

# Sentencias Condicionales - switch-case

```
#include <iostream>
using namespace std;
int main()
{  char letra;
   cout  << " Introduzca una letra: ";
   cin >> letra;
  switch (letra)
      {  case 'a' :          case 'e' :          case 'i' :
         case 'o' :          case 'u' :
               cout << " La letra es una vocal minuscula: " << letra << endl;
               break;
         case 'A' :          case 'E' :          case 'I' :
         case 'O' :          case 'U' :
               cout << " La letra es una vocal mayuscula: " << letra << endl;
               break;
         default : cout << " La letra no es una vocal:" << letra << endl;
      }

    return 0;
}
```

UNIVERSIDAD NEBRIJA

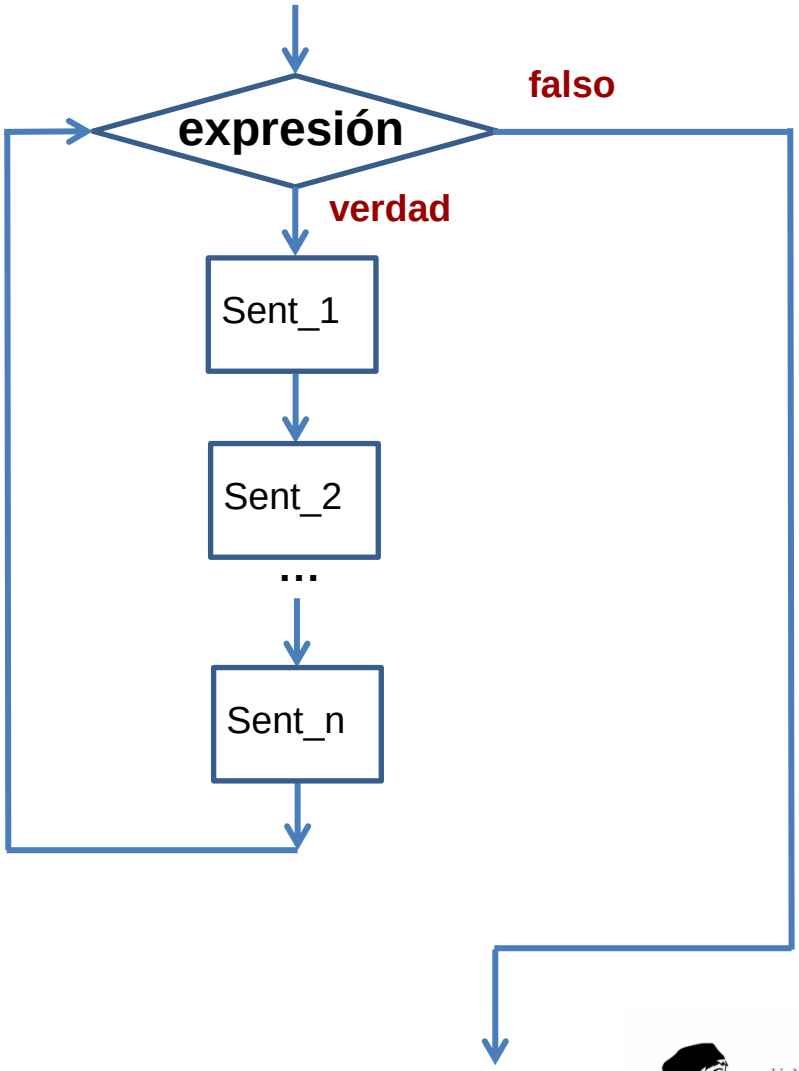
Programación en C++11.

- Estructura de un programa en C++11.
- Instrucciones de Entrada y Salida: “Hola Mundo”.
- Tipos de datos y variables.
- **Manipulación de datos:**
  - Sentencias condicionales: if, switch-case.
  - **Control de flujo: for, while.**


**ÍNDICE**

# Control de flujo - while

```
while (expresión)
    sentencia;

while (expresión)
 {  sentencia1;
    sentencia2;
    ...;
 }
```




**Se ejecuta 0 ó n veces**

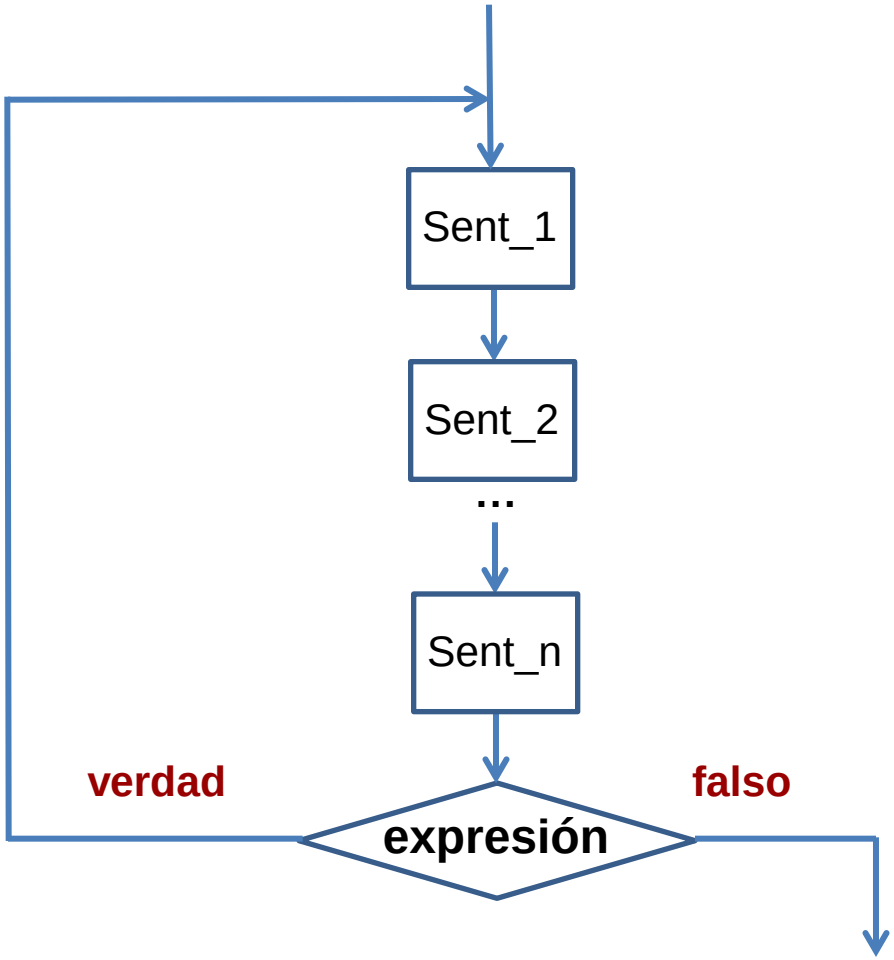
# Control de flujo - while

```
#include <iostream>
using namespace std;
int main()
{
   int secreto {15};
   int adivina {0};

   while (adivina != secreto)
   {
      cout << "Adivina el numero: ";
      cin >> adivina;
    }
    cout << "Lo has adivinado." << endl;

    return 0;
}
```

# Control de flujo - do-while

```
do
{  sentencia;
}  while (expresión);
```

```
do
{  sentencia1;
   sentencia2;
   ...;
}  while (expresión);
```




**Se ejecuta 1 ó n veces**

# Control de flujo - do-while

```
#include <iostream>
using namespace std;
int main()
{
int secreto {15};
int adivina;        //no hace falta inicializarla porque se solicita un valor
do
{
  cout << "Adivina el número: ";
  cin >> adivina;
} while (adivina != secreto);
cout << "Lo has adivinado." << endl;

return 0;
}
```

## Grado en Ingeniería Información

# PROGRAMACIÓN I

## Sesión 5

**Curso 2022-2023**

Marta N. Gómez

UNIVERSIDAD NEBRIJA

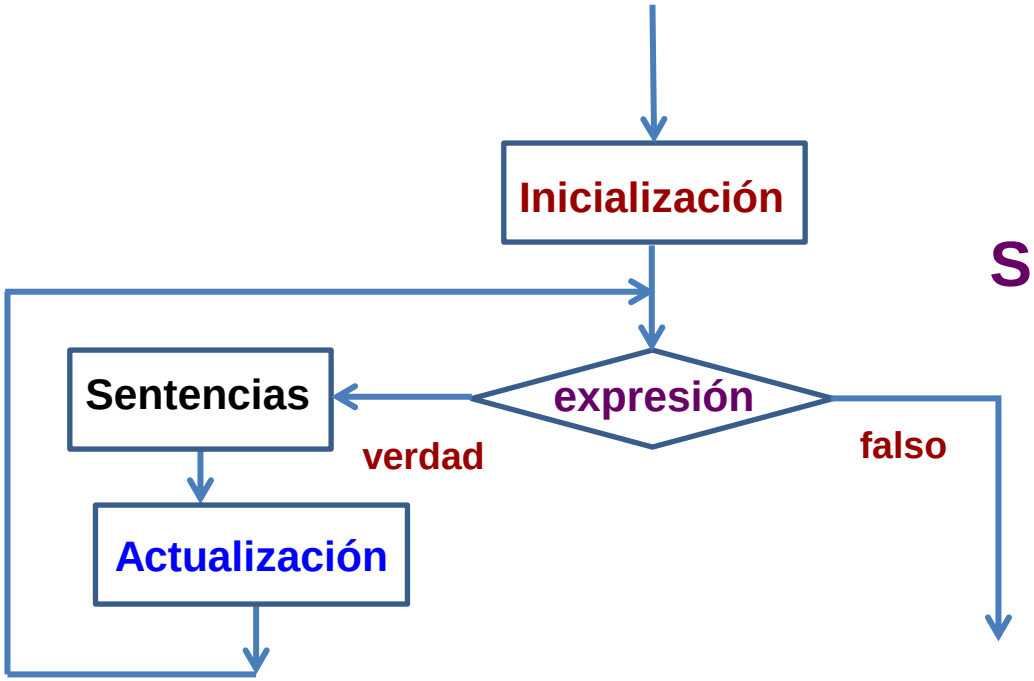
UNIVERSIDAD NEBRIJA

Programación en C++11.

- Estructura de un programa en C++11.
- Instrucciones de Entrada y Salida: “Hola Mundo”.
- Tipos de datos y variables.
- **Manipulación de datos:**
  - Sentencias condicionales: if, switch-case.
  - **Control de flujo: for, while.**


**ÍNDICE**

# Control de flujo - for (;;)




**Se ejecuta 0 ó n veces**

```
for (inicialización; expresión; actualización)
    sentencia;

for (inicialización; expresión; actualización)
{   sentencia1;
    sentencia2;
    ...;
}
```

# Control de flujo - for(;;)

```
#include <iostream>
using namespace std;
int main()
{   int factorial = 1, num;

    cout  << " Introduzca un numero positivo y mayor que 0: ";
    cin >> num;
    for (int i = 1; i <= num; i = i + 1)
        factorial = factorial * i;

    cout << num << "! = " << factorial << endl;

    system (" pause ");
    return 0;
}
```

# break:

- ***Salto incondicional***.
- Se puede utilizar dentro de los ***bucles*** y de los ***switch*** para forzar su salida.
- Provoca la ***salida inmediata*** de la ejecución de ese ***switch*** o del ***bucle*** (en caso de estar en bucles anidados, sólo se sale del ***bucle interno***).

```
#include <iostream>
using namespace std;
int main()
{    int num;
     cout  << " Introduzca un numero positivo, entre 0 y 10: ";
     cin >> num;
     cout  << " Tabla del " << num << endl;

     for (int multiplicador = 1; multiplicador <= 10; multiplicador++)
     {      if (multiplicador > 5)  break;  // La  tabla llega hasta el 5
            cout << num << " x " << multiplicador << " = ";
            cout << num * multiplicador << endl;
     }

     return 0;
}
```

## continue:

- ***Salto incondicional***.
- Se puede utilizar dentro de los bucles para ***detener su ejecución*** y ***forzar una nueva iteración*** volviendo a comprobar la **condición** del bucle.

# Sentencias de Control

```
#include <iostream>
using namespace std;
int main()
{   char c;
    while (true)
    {   cout << " MENU PRINCIPAL: " << endl;
        cout << "Seleccione  i: izquierda, d: derecha, f: fin -> ";
        cin >> c;
        if  (c == 'f')   break;                    // Fuera de while (true)
        if  (c == 'i')  {  cout << " MENU IZQUIERDA: " << endl;
                           cout << " Seleccione a o b: ";
                           cin >> c;
                           if (c == 'a')  { cout << " Ha elegido 'a' " << endl;
                                            continue;     }  // Regresa al menú principal
                           if (c == 'b')   cout << " Ha elegido 'b' " << endl;
                           else   cout << " No ha elegido ni a ni b " << endl;
                           continue;       // Regresa al menú principal
                        }    // Cierra el if (c == 'i')
```

```
    if (c == 'd')  {  cout << " MENU DERECHO: " << endl;
                      cout << " Seleccione c o d: ";
                      cin >> c;
                      if  (c == 'c')   cout << " Ha elegido ’c’ " << endl;
                      else if  (c == 'd')   cout << " Ha elegido ’d’ " << endl;
                      else    cout << " No ha elegido ni c ni d " << endl;
                      continue;                    // Regresa al menú principal
                    } // Cierra el if (c == 'd’)
    cout << " Debe elegir i o d o f " << endl;
  } // Cierra del while

  cout << " Saliendo del menu... " << endl;

  return 0;
}
```