# FUNDAMENTOS DE PROGRAMACIÓN

## TEORÍA Nº 3

## UNIDAD 2: INTRODUCCIÓN A LA PROGRAMACIÓN

**Facultad de Ingeniería**
**Universidad Nacional de Jujuy**


C++

# ÍNDICE

- Lenguaje de Programación C++
- Instaladores de C++
- Estructura General de un Programa
- Tipos de Datos en C++
- Operadores en C++
- Asignación, lectura y escritura en C++
- Paradigmas de programación
  - Programación estructurada
- Estructuras de programación Secuenciales, Condicionales o Selectivas en C++.

# C++ (1) - INSTALACIÓN

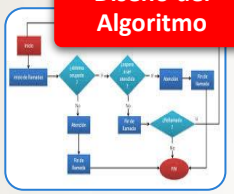
- **Dev-C++** es un entorno de desarrollo integrado para programar en lenguaje C/C++. Usa MinGW, una versión de GCC, como compilador. Dev-C++ puede además ser usado en combinación con Cygwin y cualquier otro compilador basado en GCC. El Entorno está desarrollado en el lenguaje Delphi de Borland. Fuentes y sitio de descarga https://www.bloodshed.net/index.html.
- **CodeBlocks** es un entorno de desarrollo integrado libre y multiplataforma para el desarrollo de programas en lenguaje C++. Está basado en la plataforma de interfaces gráficas WxWidgets, por lo que puede usarse libremente en diversos sistemas operativos, y está bajo GNU. Es una herramienta para desarrollar programas en C++ que ofrece una interfaz sencilla a los usuarios. Fuentes y sitio de descarga http://www.codeblocks.org/downloads/binaries.
- Opciones online: http://cpp.sh/, https://www.jdoodle.com/online-compiler-c++, https://www.onlinegdb.com/online_c++_compiler, entre otros.
- App:https://play.google.com/store/apps/details?id=ru.iiec.cxxdroid&hl=es&gl=US

# C++ (2) - ESTRUCTURA GENERAL DE PROGRAMA

- **Declaraciones**
  - Librerías (*include* indica al compilador qué librerías incluir en el programa objeto para generar el ejecutable)
  - Módulos (tipo y argumentos de los módulos)
  - Variables Globales (tipos e identificadores)
  - Constantes
- **Programa Principal**
  - La función *main* contiene declaraciones de variables e instrucciones necesarias para controlar las operaciones que ejecuta el programa.
- **Módulos**
  - Se especifica el código correspondiente a cada módulo o componente de programa.

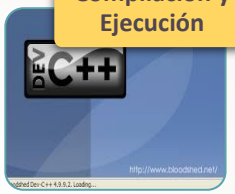
# FASES DE DESARROLLO

Análisis del Problema

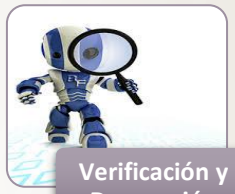


Diseño del Algoritmo


Codificación



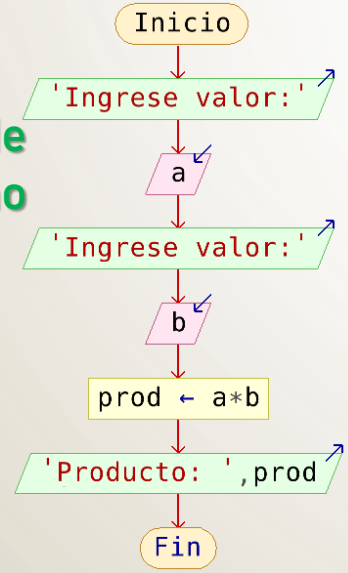

Compilación y Ejecución



Verificación y Depuración

Documentación y Mantenimiento

## Diseño de Algoritmo




CODIFICACIÓN

## Programa Fuente

```
#include <iostream>

using namespace std;

main()
{ int a,b,prod;
  cout << "Ingrese dato: ";
  cin >> a;
  cout << "Ingrese dato: ";
  cin >> b;
  prod=a*b;
  cout << "Producto: " << prod << endl;
  system("pause");
}
```

# ESTRUCTURA GRAL DE PROGRAMA (1)

- **Declaraciones**
  - Librerías (*include* indica al compilador qué librerías incluir en el programa objeto para generar el ejecutable)
  - Módulos (tipo y argumentos de los módulos)
  - Variables Globales (tipos e identificadores)
  - Constantes
- **Programa Principal**
  - La función *main* contiene declaraciones de variables e instrucciones necesarias para controlar las operaciones que ejecuta el programa.
- **Módulos**
  - Se especifica el código correspondiente a cada módulo o componente de programa.

# ESTRUCTURA GRAL DE PROGRAMA (2)

```
/* Archivos de cabecera */
#include <archivo_cabecera.h>
#include <archivo_cabecera.h>
```

Librerías del Lenguaje

```
/* Prototipos de funciones del programador */
tipo_dato función1 (argumentos);
tipo_dato función2 (argumentos);
```

Módulos definidos por el programador

```
/* Variables y constantes globales */
tipo_dato variable_global;
const tipo_dato constante_global=valor;
#define PI 3.14
```

# ESTRUCTURA GRAL DE PROGRAMA (3)

```
/* Algoritmo principal */
tipo_dato main(argumentos)
{
   /*Variables locales del algoritmo principal */
   tipo_dato nombre_variable1, nombre_variable2;
   tipo_dato nombre_variable3, nombre_variable4;
   ...
   /* Instrucciones del algoritmo principal */
   ...
   función1(argumentos);
   ...
   función2(argumentos);
   ...
   return valor;
```

# ESTRUCTURA GRAL DE PROGRAMA (4)

```
/* Código completo de las funciones del programador*/

tipo_dato función1 (argumentos)
{
    /* Variables locales e instrucciones del módulo */
}

tipo_dato función2 (argumentos)
{
    /* Variables locales e instrucciones del módulo */
}
```

# TIPOS DE DATOS EN C/C++

| Nombre | Descripción | Tamaño | Rango |
|---|---|---|---|
| char | Caracter (código ASCII) | 8 bits | Con signo: -128 ... 127<br>Sin signo: 0 ... 255 |
| short int (short) | Número entero corto | 16 bits | Con signo: -32768 ... 32767<br>Sin signo: 0 ... 65535 |
| int | Número entero | 32 bits | Con signo: -2147483648 ... 2147483647<br>Sin signo: 0 ... 4294967295 |
| long int (long) | Número entero largo | 64 bits | Con signo: -9223372036854775808, 9223372036854775807<br>Sin signo: 0 ... 18446744073709551615 |
| float | Número real | 32 bits | 3,4*10-38 ... 3,4*10+38 (6 decimales) |
| double | Número real en doble precisión | 64 bits | 1,7*10-308 ... 1,7*10+308 (15 decimales) |
| long double | Número real largo de doble precisión | 80 bits | 3,4*10-4932 ... 1,1*10+4932 |
| bool | Valor booleano | 1 bit | true (VERDADERO) o false (FALSO) |

# OPERADORES EN C/C++

| Tipo | Operadores |
|---|---|
| Asignación | **=** |
| Aritmético | Potencia: ***pow(x,y)*** (librería *math.h*); *x, y* valores numéricos<br>Producto: *****<br>Cociente: **/**<br>Módulo o resto: **%**<br>Sumar: **+**<br>Diferencia: **-** |
| Alfanuméricos | Operaciones con cadenas (librería *string.h*)<br>*strcat(s,t);* concatena *t* al final de *s*.<br>*strcmp(s,t);* compara *s* y *t*, retornando negativo, cero, o positivo para: *s<t*, *s==t*, *s > t*.<br>*strcpy(s,t);* copia *t* en *s*.<br>*strlen(s);* retorna la longitud de *s*.<br>donde *s* y *t* son variables de tipo cadena. |
| Lógicos | Negación (NO, NOT): **!**<br>Conjunción (Y, AND): **&&**<br>Disyunción (O, OR): **\|\|** |
| Relacionales | Igual: **==**<br>Distinto: **!=**<br>Mayor: **>**<br>Mayor o igual: **>=**<br>Menor: **<**<br>Menor o igual: **<=** |

# ESTRUCTURAS SECUENCIALES (1)

- Asignación

**variable ← expresión**

variable←expresión

```
variable=expresión;
```

- Lectura

**variables**

LEER variable

```
cin >> variable;
```

- Escritura

**“Mensaje”, variables**

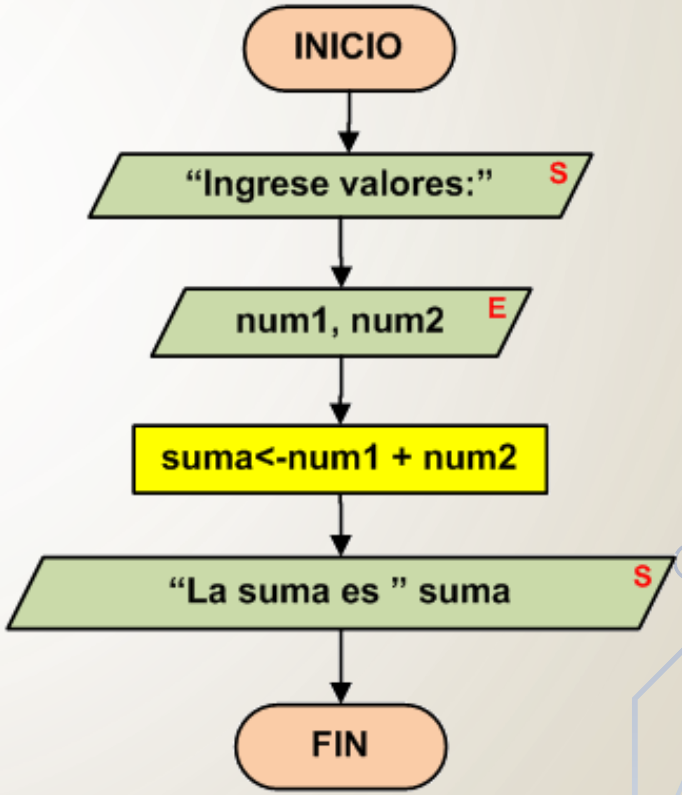
ESCRIBIR “mensaje”
ESCRIBIR variable
ESCRIBIR “texto”, variable

```
cout << "texto";
cout << variable;
cout << "texto" << variable;
```

# ESTRUCTURAS SECUENCIALES (2)

- Diseñe un algoritmo que sume 2 valores ingresados por el usuario.

```
PROGRAMA sumar_valores
VARIABLES
  num1, num2, suma: ENTERO
INICIO
  ESCRIBIR "Ingrese valor: "
  LEER num1
  ESCRIBIR "Ingrese valor: "
  LEER num2
  suma<-num1+num2
  ESCRIBIR "Resultado ",suma
FIN
```



# ESTRUCTURAS SECUENCIALES (3)

**Programa que suma 2 valores ingresados por el usuario.**

| ASIGNACIÓN | |
|---|---|
| suma ← suma + 10 | suma=suma + 10; |
| **LECTURA** | |
| leer valor | **cin** >> valor; |
| **ESCRITURA** | |
| escribir 'hola mundo!!!' | **cout** << "hola mundo!!!"; |

```
#include <iostream>

using namespace std;

main()
{ int num1,num2,suma;
  cout << "Ingrese valor: ";
  cin >> num1;
  cout << "Ingrese valor: ";
  cin >> num2;
  suma=num1+num2;
  cout << "Resultado" << suma << endl;
  system("pause");
}
```

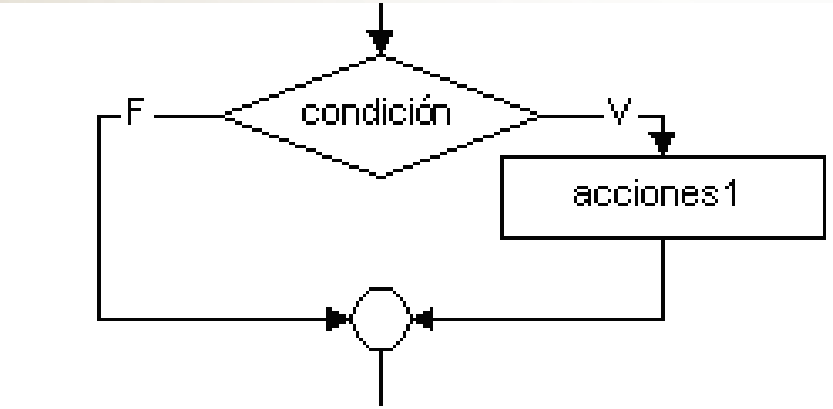
Operación de Lectura

Operación de Escritura

Operación de Asignación

# ESTRUCTURAS SELECTIVAS (1)

- Selectivas Simples



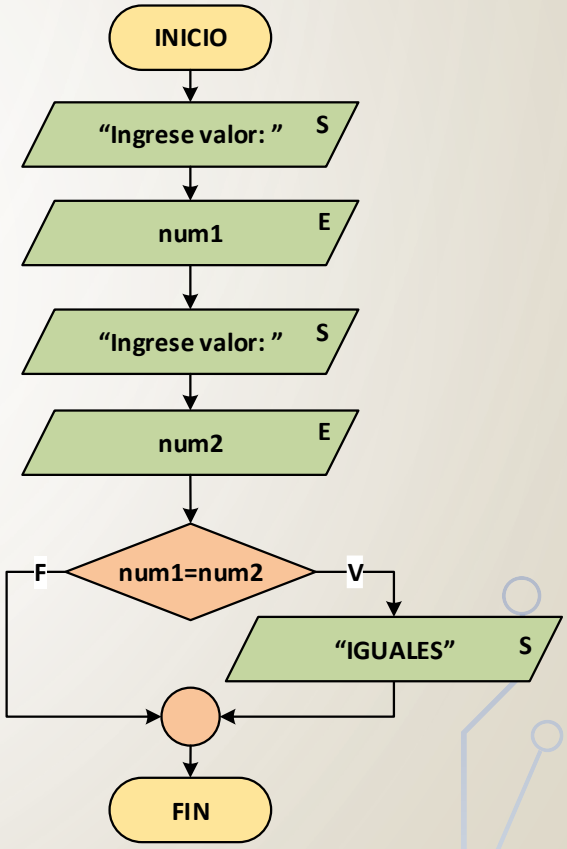

**SI condición ENTONCES**
**acciones**
**FIN_SI**

```
if (condición)
    acción_simple;

if (condición)
{
    bloque_acciones;
}
```

# ESTRUCTURAS SELECTIVAS (

- Diseñe un algoritmo que compare 2 valores ingresados por el usuario y determine si son iguales.

```
PROGRAMA comparar_valores
VARIABLES
  num1, num2: ENTERO
INICIO
  ESCRIBIR "Ingrese valor: "
  LEER num1
  ESCRIBIR "Ingrese valor: "
  LEER num2
  SI num1=num2 ENTONCES
        ESCRIBIR "IGUALES"
  FIN_SI
FIN
```



# ESTRUCTURAS SELECTIVAS (3)

Programa que compara 2 valores ingresados por el usuario y determina si son iguales o no.

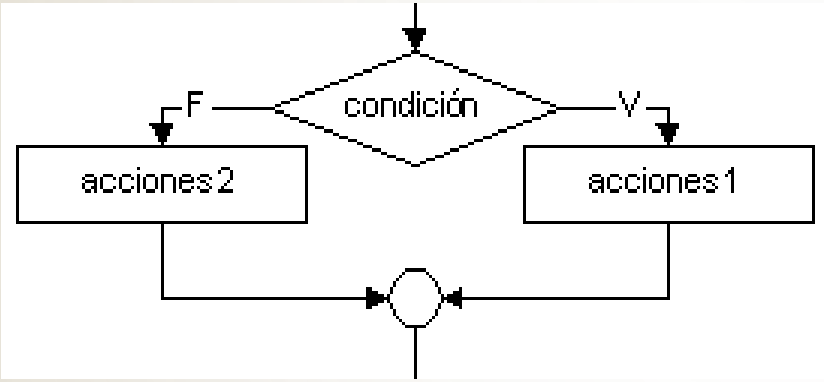
| CONDICIONALES O SELECTIVAS SIMPLES | |
|---|---|
| si condición entonces<br>acciones<br>fin_si | **if** (condición)<br>acciones; |

```
#include <iostream>
#include <stdlib.h>

using namespace std;

main()
{ int num1,num2;
  cout << "Ingrese valor: ";
  cin >> num1;
  cout << "Ingrese valor: ";
  cin >> num2;
  if (num1==num2)
    cout << "Iguales" << endl;
  system("pause");
}
```

Selectiva Simple

# ESTRUCTURAS SELECTIVAS (4)

- Selectivas Dobles



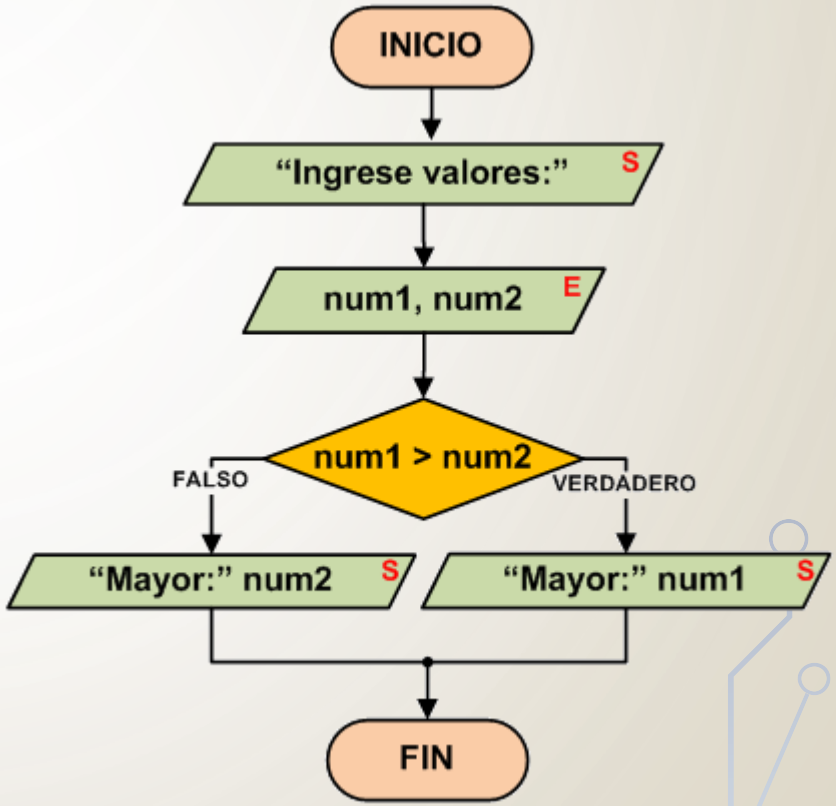

**SI condición ENTONCES**
**acciones_1**
**SINO**
**acciones_2**
**FIN_SI**

```
if (condición)
    acción_simple1;
else
    acción_simple2;


if (condición)
{
    bloque_acciones1;
}
else
{
    bloque_acciones2;
}
```

# ESTRUCTURAS SELECTIVAS (

- Diseñe un algoritmo que compare 2 valores ingresados por el usuario y determine cuál es el mayor.

```
PROGRAMA mostrar_mayor
VARIABLES
  num1, num2: ENTERO
INICIO
  ESCRIBIR "Ingrese valor: "
  LEER num1
  ESCRIBIR "Ingrese valor: "
  LEER num2
  SI num1>num2 ENTONCES
        ESCRIBIR "Mayor ", num1
  SINO
        ESCRIBIR "Mayor ", num2
  FIN_SI
FIN
```



# ESTRUCTURAS SELECTIVAS (6)

**Programa que compara 2 valores ingresados por el usuario y determina el mayor de ellos.**

**CONDICIONALES O SELECTIVAS DOBLES**

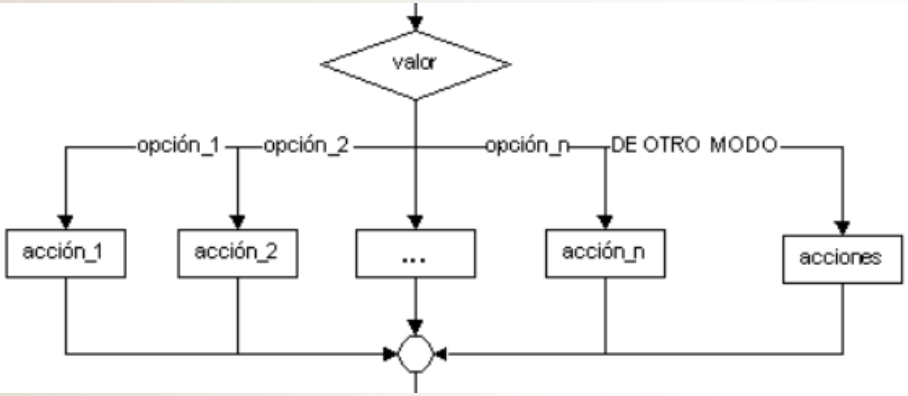
| | |
|---|---|
| si condición entonces<br>  acciones1<br>sino<br>  acciones2<br>fin_si | **if** (condición)<br>  acciones1;<br>**else**<br>  acciones2; |

```
#include <iostream>
#include <stdlib.h>

using namespace std;

main()
{ int num1,num2;
  cout << "Ingrese valor: ";
  cin >> num1;
  cout << "Ingrese valor: ";
  cin >> num2;
  if (num1>num2)
    cout << num1 << " es el mayor" << endl;
  else
    cout << num2 << " es el mayor" << endl;
  system("pause");
}
```

Selectiva Doble

# ESTRUCTURAS SELECTIVAS (7)

- Selectivas Múltiples



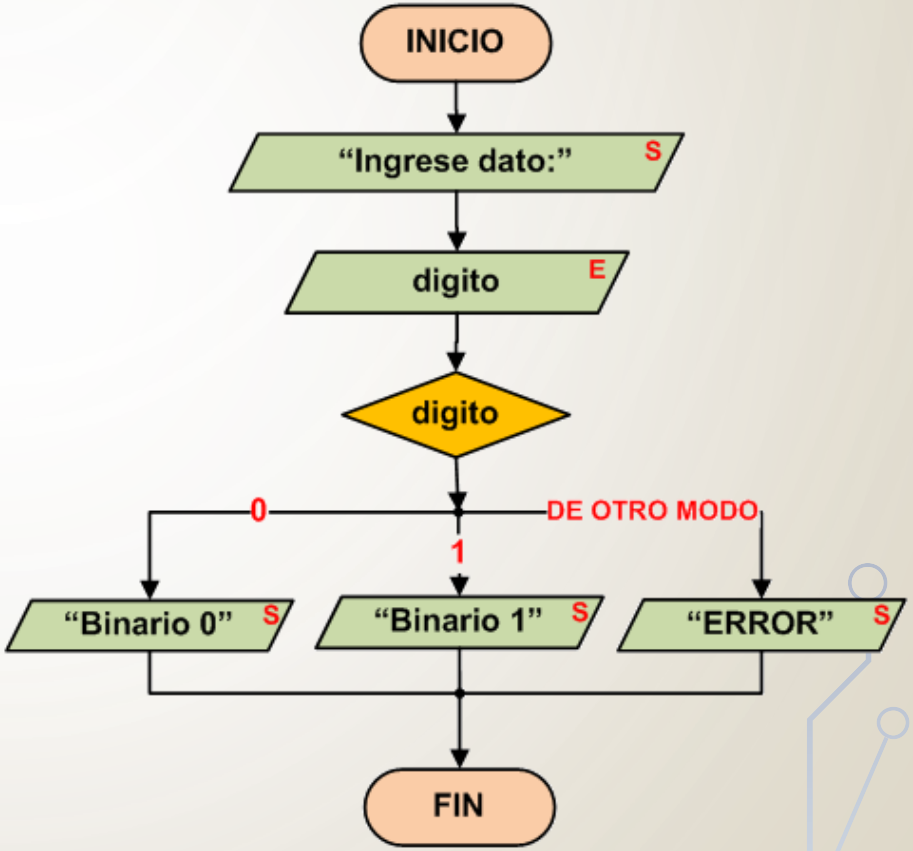

```
switch (valor)
{
 case 1: acción_1;
         break;
 case 2: acción_2;
         break;

 case n: acción_n;
         break;
 default: otra_acción;
}
```

**SEGÚN valor HACER**
    **op1: acción_1**
    **…**
    **opn: acción_n**
**DE OTRO MODO**
    **otra_acción**
**FIN_SI**

# ESTRUCTURAS SELECTIVAS (C++)

- Diseñe un algoritmo que determine si un dígito ingresado por el usuario es binario o no.

```
PROGRAMA binarios
VARIABLES
  digito: ENTERO
INICIO
  ESCRIBIR "Ingrese valor: "
  LEER digito
  SEGUN digito HACER
   0:ESCRIBIR "Binario 0"
   1:ESCRIBIR "Binario 1"
   DE OTRO MODO:
        ESCRIBIR "ERROR"
  FIN_SEGUN
FIN
```



# ESTRUCTURAS SELECTIVAS (9)

**Programa que indica si un valor ingresado por el usuario es un dígito binario.**

**CONDICIONALES O SELECTIVAS MÚLTIPLES**

| | |
|---|---|
| según opción hacer<br>  op1: acciones_1<br>  op2: acciones_2<br>  ...<br>  opn: acciones_n<br>de otro modo<br>  acciones<br>fin_segun | **switch** (opción)<br>{<br>  **case** op1: acciones_1; **break;**<br>  **case** op2: acciones_2; **break;**<br>  ...<br>  **case** opn: acciones_n; **break;**<br>  **default**: acciones;<br>} |

```
#include <iostream>
#include <stdlib.h>

using namespace std;

main()
{ int digito;
  cout << "Ingrese valor: ";
  cin >> digito;
  switch (digito)
  { case 0: cout << "Binario 0" << endl;
            break;
    case 1: cout << "Binario 1" << endl;
            break;
    default: cout << "ERROR" << endl;
  }
  system("pause");
}
```

Selectiva Múltiple