FUNDAMENTOS DE PROGRAMACIÓN

TEORÍA Nº 3

UNIDAD 2: INTRODUCCIÓN A LA PROGRAMACIÓN

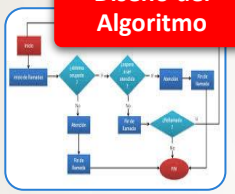
Facultad de Ingeniería
Universidad Nacional de Jujuy

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++.

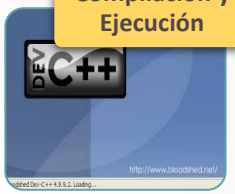
FASES DE DESARROLLO

Análisis del Problema

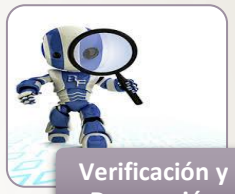


Diseño del Algoritmo

Codificación



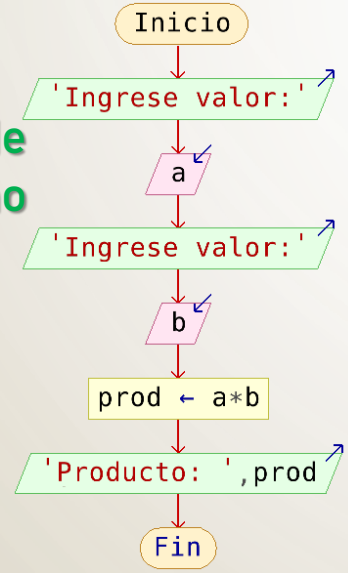
Compilación y Ejecución



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");
}
```

TIPOS DE DATOS EN C/C++

Nombre	Descripción	Tamaño	Rango
char	Caracter (código ASCII)	8 bits	Con signo: -128 ... 127 Sin signo: 0 ... 255
short int (short)	Número entero corto	16 bits	Con signo: -32768 ... 32767 Sin signo: 0 ... 65535
int	Número entero	32 bits	Con signo: -2147483648 ... 2147483647 Sin signo: 0 ... 4294967295
long int (long)	Número entero largo	64 bits	Con signo: -9223372036854775808, 9223372036854775807 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 Producto: ***** Cociente: **/** Módulo o resto: **%** Sumar: **+** Diferencia: **-**
Alfanuméricos	Operaciones con cadenas (librería *string.h*) *strcat(s,t);* concatena *t* al final de *s*. *strcmp(s,t);* compara *s* y *t*, retornando negativo, cero, o positivo para: *s<t*, *s==t*, *s > t*. *strcpy(s,t);* copia *t* en *s*. *strlen(s);* retorna la longitud de *s*. donde *s* y *t* son variables de tipo cadena.
Lógicos	Negación (NO, NOT): **!** Conjunción (Y, AND): **&&** Disyunción (O, OR): **||**
Relacionales	Igual: **==** Distinto: **!=** Mayor: **>** Mayor o igual: **>=** Menor: **<** Menor o igual: **<=**


C++

C++ (1) - INSTALACIÓN

- **Zinjal** es Software Libre y se distribuye gratuitamente bajo licencia GPL, por lo que está permitida su copia o distribución bajo los criterios de la misma. Ha sido desarrollado por Pablo Novara, inicialmente en el marco de una Beca de Iniciación a la Investigación Científica otorgada por la Universidad Nacional del Litoral (Argentina).**https://zinjai.sourceforge.net/**

- **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.

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 */
}
```

ESTRUCTURAS SECUENCIALES (1)

- Asignación

variable ← expresión

variable←expresión

```
variable=expresión;
```

- Lectura

variables

LEER variable

```
cin >> variable;
```

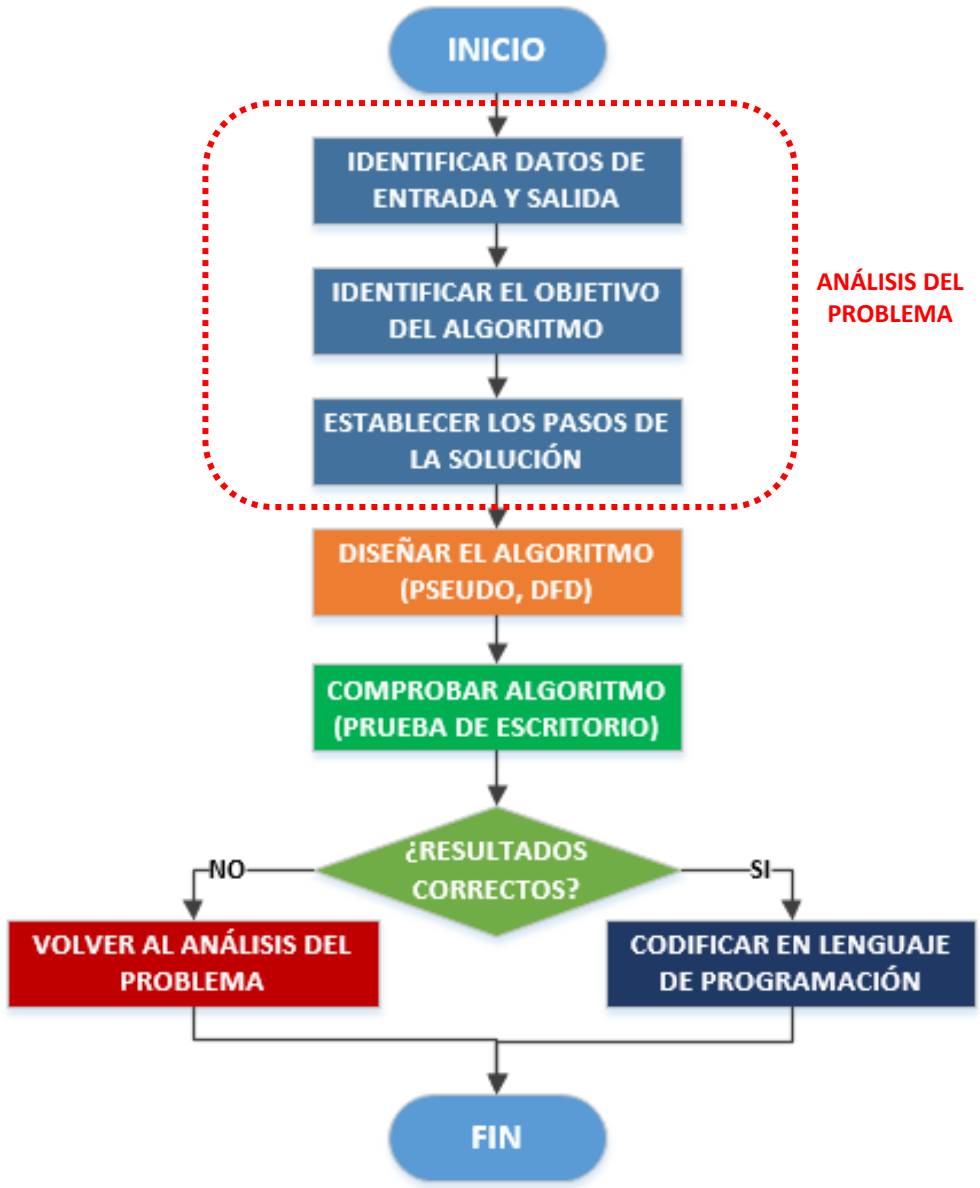
- Escritura

“Mensaje”, variables

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

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

METODOLOGÍA DE TRABAJO (1)



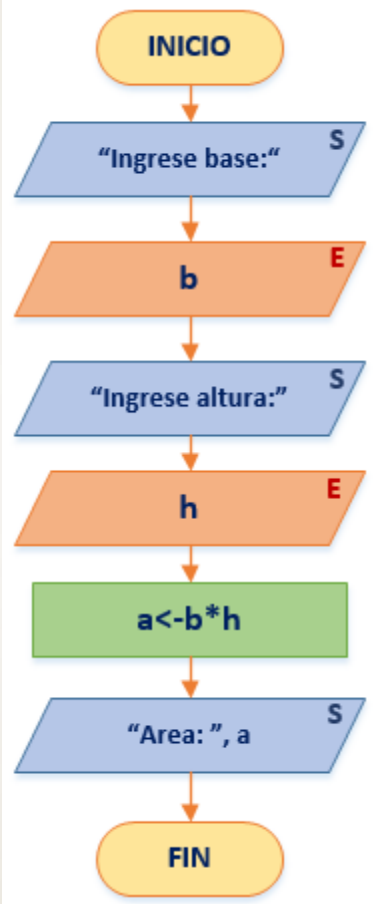
ESTRUCTURAS SECUENCIALES (2)

- Diseñe un algoritmo (diagrama de flujo y pseudocódigo) que calcule la superficie de un rectángulo de base b y altura h.,
- ***Análisis del problema***
- El análisis del problema debe permitirnos responder a las siguientes preguntas:

¿Cuál es el problema que debo resolver?	**Calcular el área de un rectángulo** 👍
¿Qué datos necesito para resolver el problema?	Base y altura del rectángulo 👍
¿Cuál es el resultado que debo obtener?	Área de un rectángulo 👍
¿Cuáles son los pasos que debo realizar para obtener la solución?	1) Obtener los datos de base y altura 👍 2) Aplicar la fórmula de superficie de rectángulos, utilizando los datos de base y altura 👍 3) Mostrar el resultado calculado 👍

ESTRUCTURAS SECUENCIALES (3)

```
PROGRAMA rectángulo
VARIABLES
  b, h, a: REAL
INICIO
ESCRIBIR "ingrese base"
LEER b
ESCRIBIR "ingrese altura"
LEER h
a b*h
ESCRIBIR "Area:" a
FIN
```

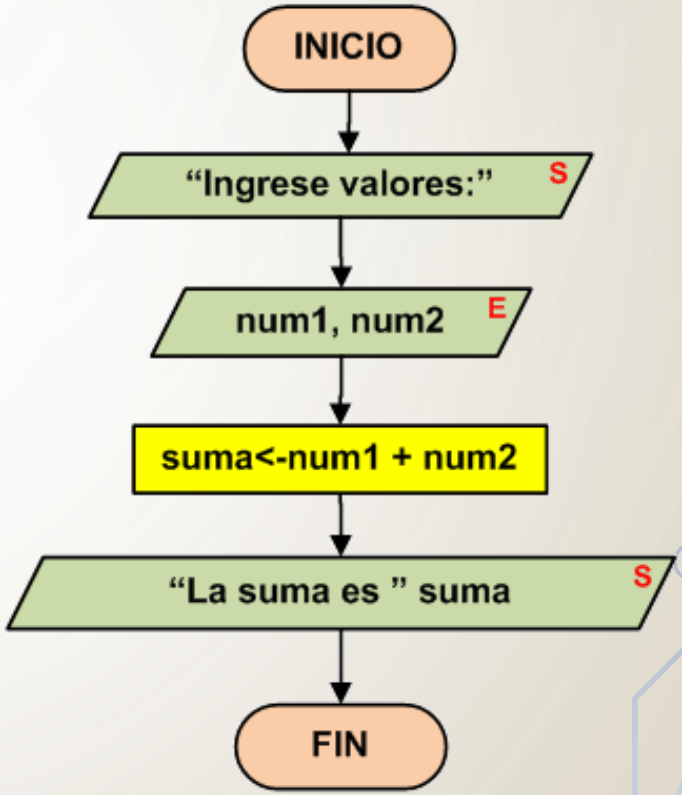


ESCRIBIR: *operación de salida que* envía a un dispositivo de salida un mensaje o valores de variables. Esta operación se conoce como *Escritura*.
LEER: *operación de entrada* que permite capturar valores y asignarlos a variables específicas. Esta operación se conoce como *Lectura*.

ESTRUCTURAS SECUENCIALES (4)

- 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");
}
```

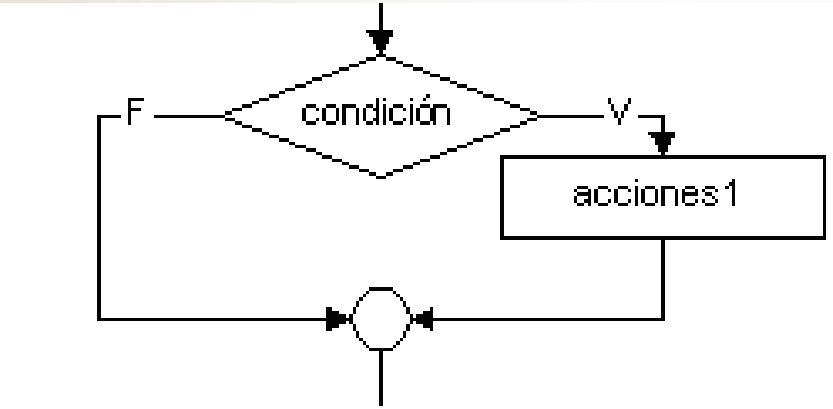
Operación de Lectura

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 (2)

- Diseñe un algoritmo (diagrama de flujo y pseudocódigo) que calcule la longitud de una cadena ingresada por el usuario. Si la cadena estuviese vacía, debe mostrarse notificarse al usuario.

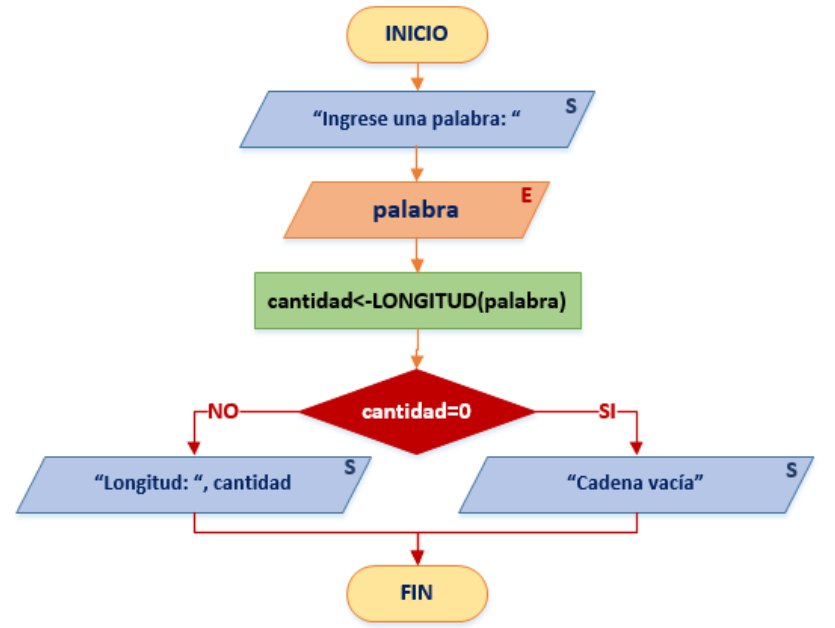
Análisis del problema

- El análisis del problema debe permitirnos responder a las siguientes preguntas:

¿Cuál es el problema que debo resolver?	**Contar los caracteres de una cadena**
¿Qué datos necesito para resolver el problema?	**Una cadena de caracteres**
¿Cuál es el resultado que debo obtener?	**Cantidad de caracteres de la cadena o un mensaje en caso de una cadena vacía.**
¿Cuáles son los pasos que debo realizar para obtener la solución?	**1) Obtener la cadena de caracteres** **2) Aplicar la operación que cuenta los caracteres de una cadena** **3) Mostrar la longitud de la cadena o el mensaje de cadena vacía.**

ESTRUCTURAS SELECTIVAS (3)

```
PROGRAMA cadena
VARIABLES
  palabra: CADENA
  cantidad: ENTERO
INICIO
ESCRIBIR "Ingrese una palabra"
LEER palabra
cantidad<-LONGITUD(palabra)
SI cantidad=0 ENTONCES
 ESCRIBIR "CADENA VACIA"
SINO
 ESCRIBIR "Longitud: ", cantidad
FIN_SI
FIN
```

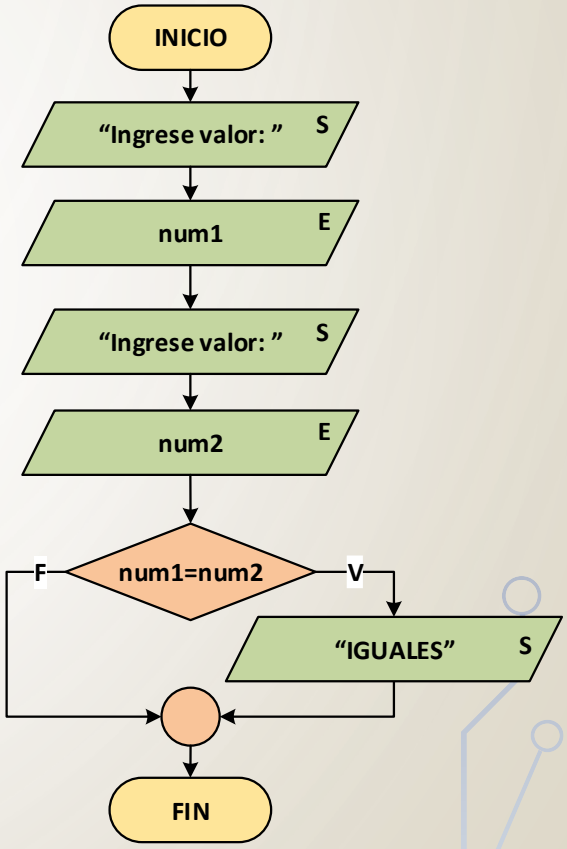


El algoritmo tiene por objetivo **contar los caracteres de una cadena** ingresada por el usuario, usando para ello 2 variables: ***palabra*** de tipo cadena y ***cantidad*** de tipo entero. La variable *palabra* (**dato de entrada**) se cargará mediante una operación de lectura (entrada por teclado), mientras que la variable *cantidad* (**resultado**) almacenará la cantidad de caracteres de la cadena aplicando la operación LONGITUD. Si la cadena estuviese vacía, se deberá presentar un mensaje que notifique al usuario de tal situación.

ESTRUCTURAS SELECTIVAS (4)

- 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 (5)

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

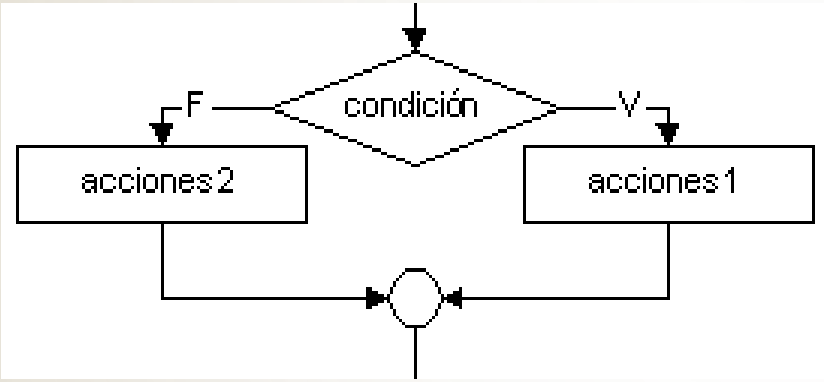
CONDICIONALES O SELECTIVAS SIMPLES	
si condición entonces acciones fin_si	**if** (condición) acciones;

```
#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 (6)

- Selectivas Dobles



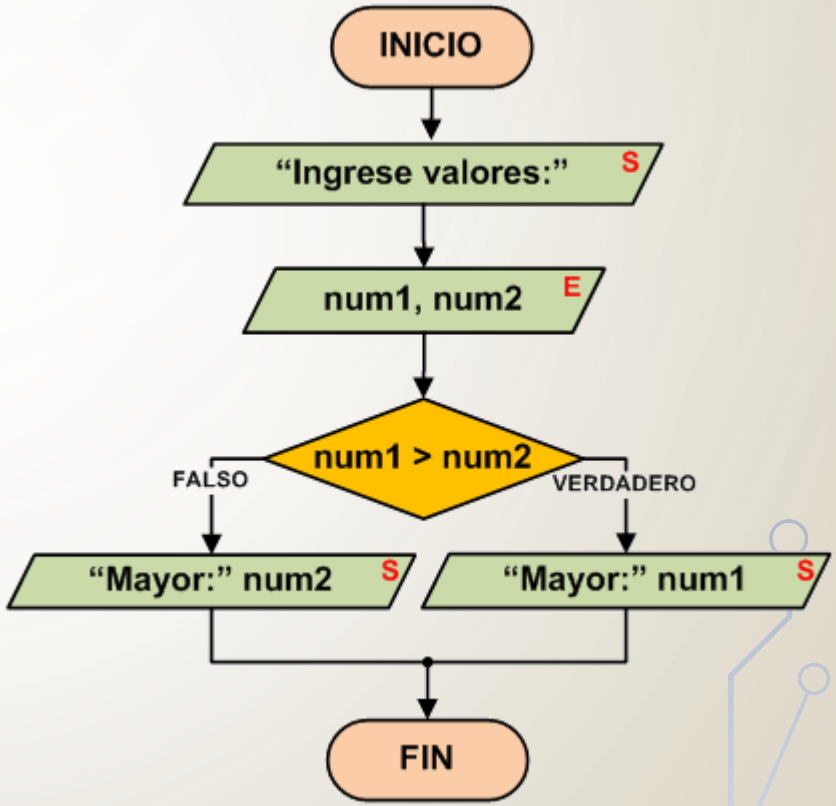
SI condición ENTONCES
acciones_1
SINO
acciones_2
FIN_SI

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


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

ESTRUCTURAS SELECTIVAS (7)

- 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 (8)

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

CONDICIONALES O SELECTIVAS DOBLES

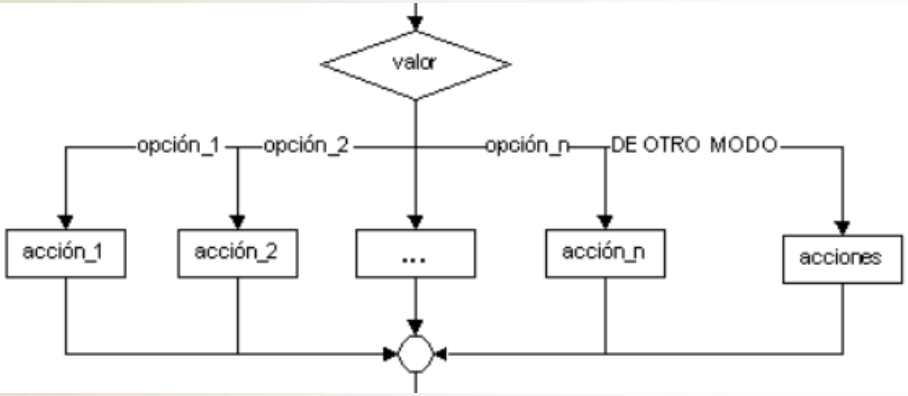
si condición entonces
  acciones1
sino
  acciones2
fin_si
	if (condición)
	  acciones1;
	else
	  acciones2;

```
#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 (9)

- Selectivas Múltiples



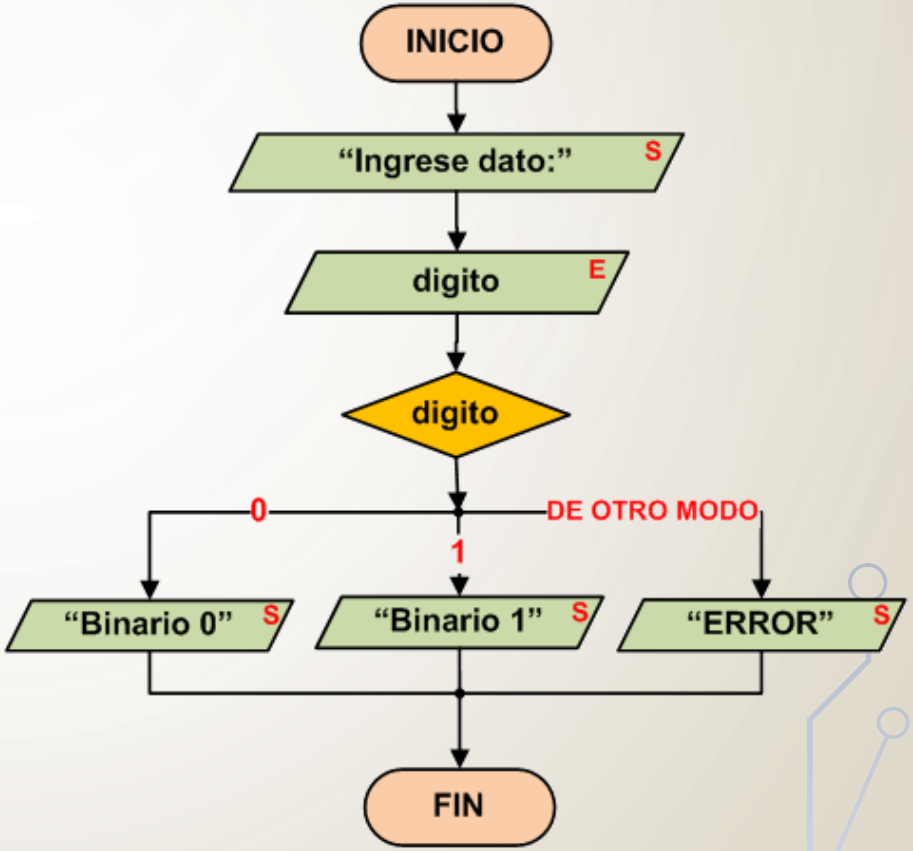

```
switch (valor)
{
 case 1: acción_1;
         break;
 case 2: acción_2;
         break;

 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 (10)

- 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 (11)

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

CONDICIONALES O SELECTIVAS MÚLTIPLES

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

```
#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