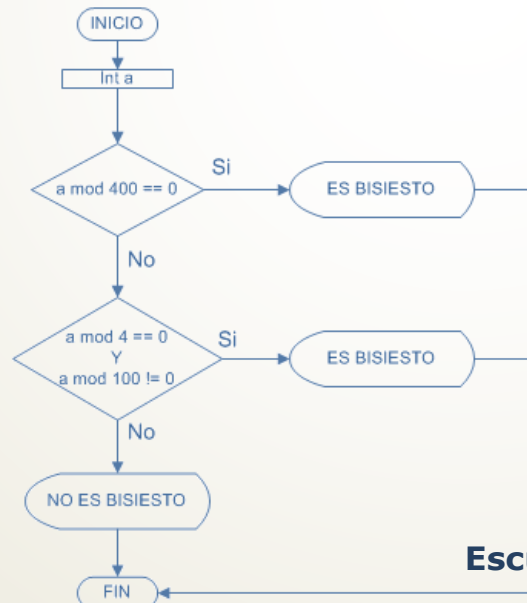


Programación Estructurada

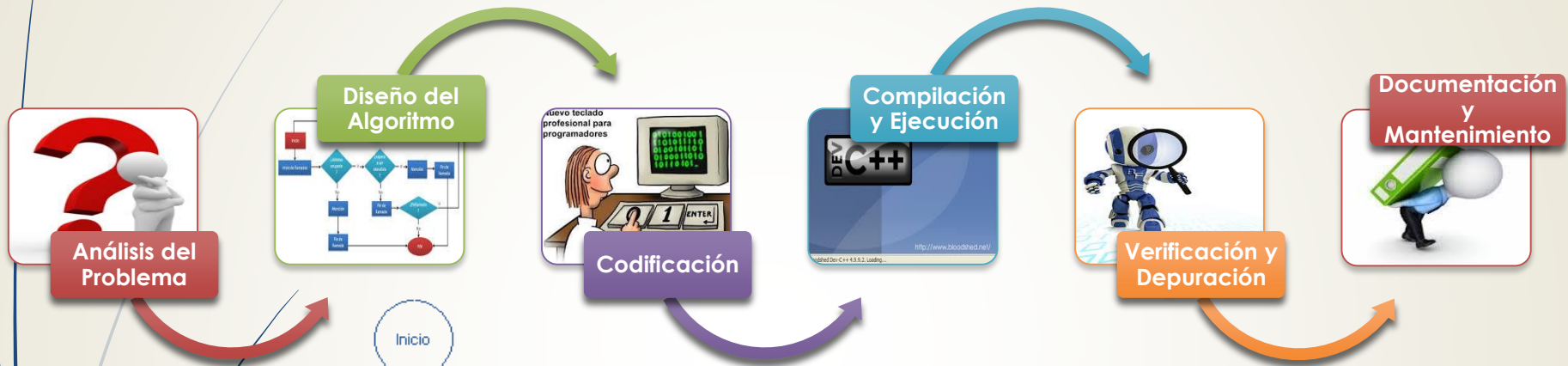
LENGUAJE DE PROGRAMACIÓN C/C++



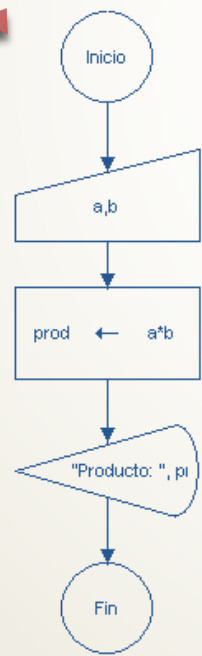
Escuela de Minas "Dr. Horacio Carrillo"
Universidad Nacional de Jujuy



Fases de Desarrollo



Diseño de Algoritmo



```

#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");
}
    
```

Programa Fuente

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 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	$3,4 \cdot 10^{-38}$... $3,4 \cdot 10^{+38}$ (6 decimales)
double	Número real en doble precisión	64 bits	$1,7 \cdot 10^{-308}$... $1,7 \cdot 10^{+308}$ (15 decimales)
long double	Número real largo de doble precisión	80 bits	$3,4 \cdot 10^{-4932}$... $1,1 \cdot 10^{+4932}$
bool	Valor booleano	1 bit	true (VERDADERO) o false (FALSO)

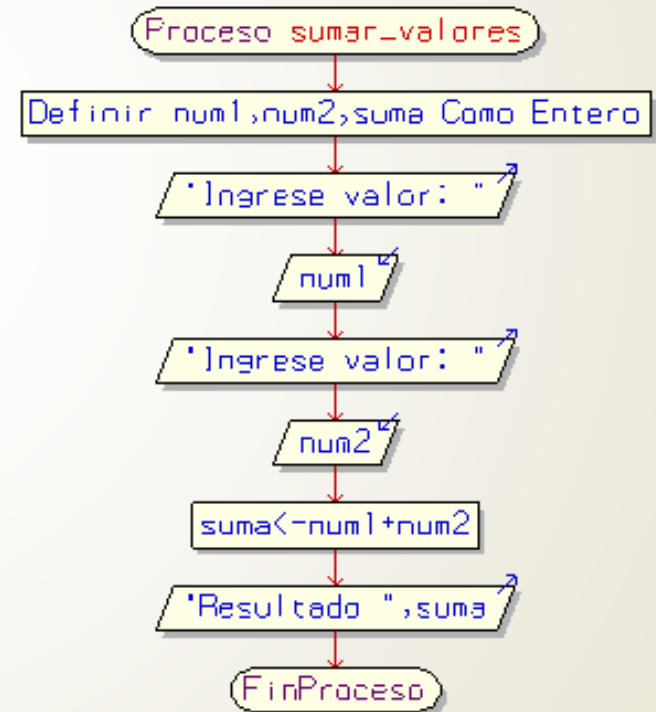
Operadores en C/C++

Tipo	Operadores
Aritmético	Potencia: pow(x,y) (librería <i>math.h</i>); <i>x, y</i> valores numéricos Producto: * Cociente: / Módulo o resto: % Sumar: + Diferencia: -
Alfanuméricos	Operaciones con cadenas (librería <i>string.h</i>) <i>strcat(s,t)</i> ; concatena <i>s</i> al final de <i>t</i> . <i>strcmp(s,t)</i> ; compara <i>s</i> y <i>t</i> , retornando negativo, cero, o positivo para: <i>s < t</i> , <i>s == t</i> , <i>s > t</i> . <i>strcpy(s,t)</i> ; copia <i>t</i> en <i>s</i> . <i>strlen(s)</i> ; retorna la longitud de <i>s</i> . donde <i>s</i> y <i>t</i> 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: <=
Asignación	=

Estructuras Secuenciales (1)

- 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 (2)

Programa que suma 2 valores ingresados por el usuario.

ASIGNACIÓN	
suma \leftarrow 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

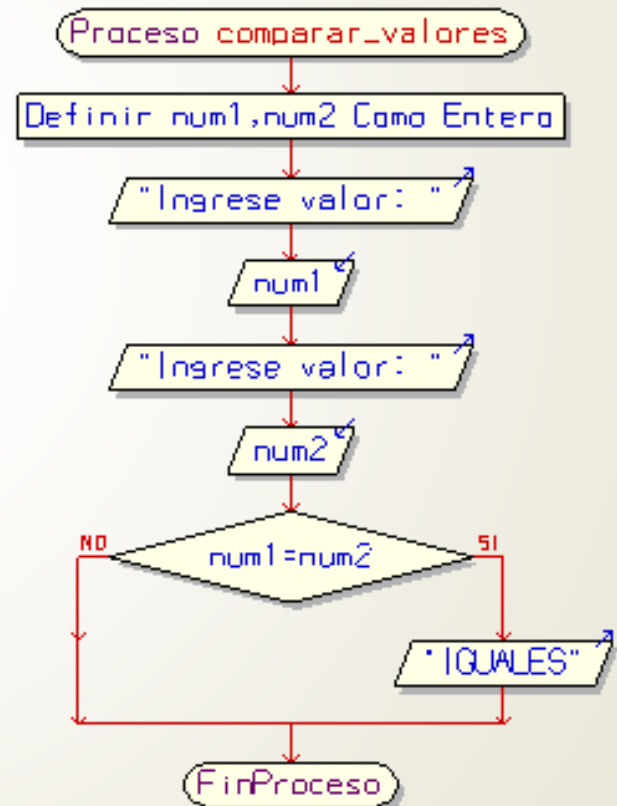
Operación de Escritura

Operación de Asignación

Estructuras Selectivas (3)

- 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 (4)

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");
}
```

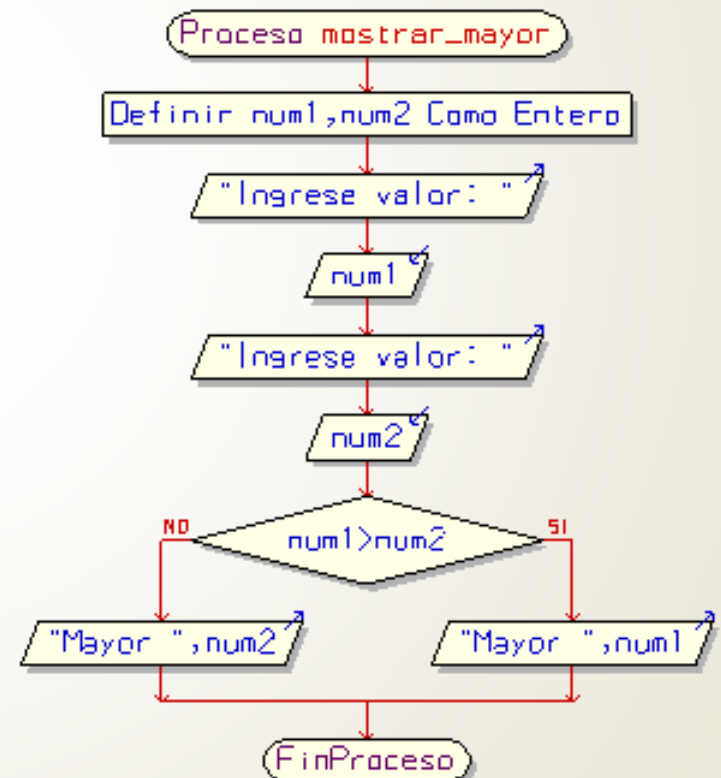
Programa que compara 2 valores ingresados por el usuario y determina si son iguales o no.

Selectiva Simple

Estructuras Selectivas (5)

- 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)

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");
}
```

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

Selectiva Doble

Estructuras Selectivas (7)

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

PROGRAMA binarios

VARIABLES

dato: ENTERO

INICIO

ESCRIBIR "Ingrese valor: "

LEER dato

SEGUN dato HACER

1: ESCRIBIR "Binario UNO"

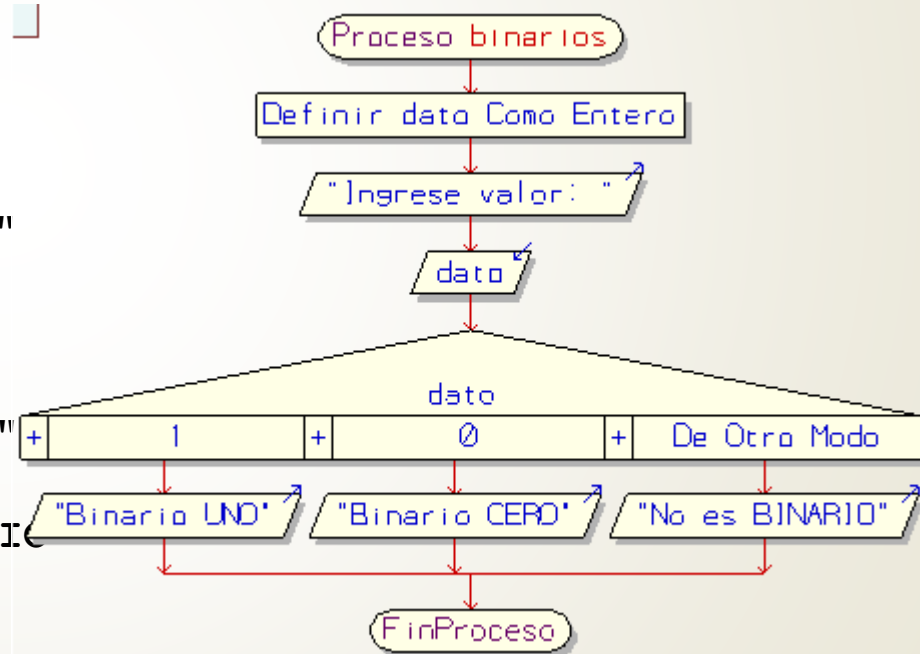
0: ESCRIBIR "Binario CERO"

DE OTRO MODO:

ESCRIBIR "No es BINARIO"

FIN_SEGUN

FIN



Estructuras Selectivas (8)

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

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

using namespace std;

main()
{ int dato;
  cout << "Ingrese valor: ";
  cin >> dato;
  switch (dato)
  { case 1: cout << "Binario UNO" << endl;
    break;
    case 0: cout << "Binario CERO" << endl;
    break;
    default: cout << "No es BINARIO" << endl;
  }
  system("pause");
}
```

CONDICIONALES O SELECTIVAS MÚLTIPLES

según opción hacer
 op1: acciones_1
 op2: acciones_2
 ...
 opn: acciones_n
 de otro modo
 acciones
 fin_según

```
switch (opción)
{
  case op1: acciones_1; break;
  case op2: acciones_2; break;
  ...
  case opn: acciones_n; break;
  default: acciones;
}
```

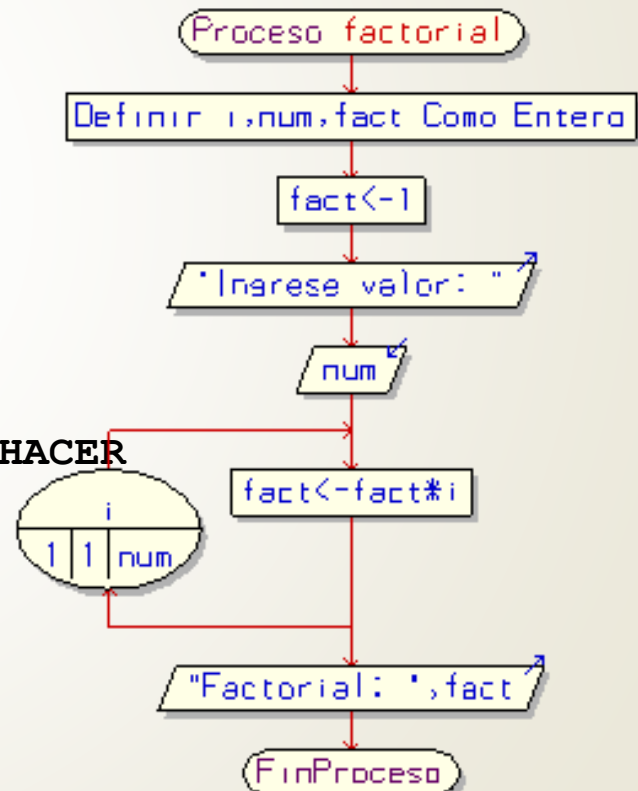
Selectiva
Múltiple

Estructuras Repetitivas (1)

- Diseñe un algoritmo que calcule el factorial de un número ingresado por el usuario.

```

PROGRAMA factorial
VARIABLES
    i, num, fact: Entero
INICIO
    fact<-1
    ESCRIBIR "Ingrese valor: "
    LEER num
    PARA i DESDE 1 HASTA num CON PASO 1 HACER
        fact<-fact*i
    FIN_PARA
    ESCRIBIR "Factorial: ", fact
FIN
  
```



Estructuras Repetitivas (2)

Programa que calcula el factorial de un número ingresado por el usuario utilizando estructuras PARA.

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

```
using namespace std;
```

```
main()
```

```
{ int num,i, fact=1;
  cout << "Ingrese valor: ";
  cin >> num;
```

```
  for(i=1;i<=num;i++)
    fact=fact*i;
```

```
  cout << "Factorial: " << fact << endl;
  system("pause");
```

```
}
```

PARA

```
para  $v$  desde  $vi$  hasta  $vf$  hacer con paso  $n$  hacer
    acciones
fin_para
```

```
for ( $v=vi, v \leq vf, v++$ )
{
    acciones;
}
```

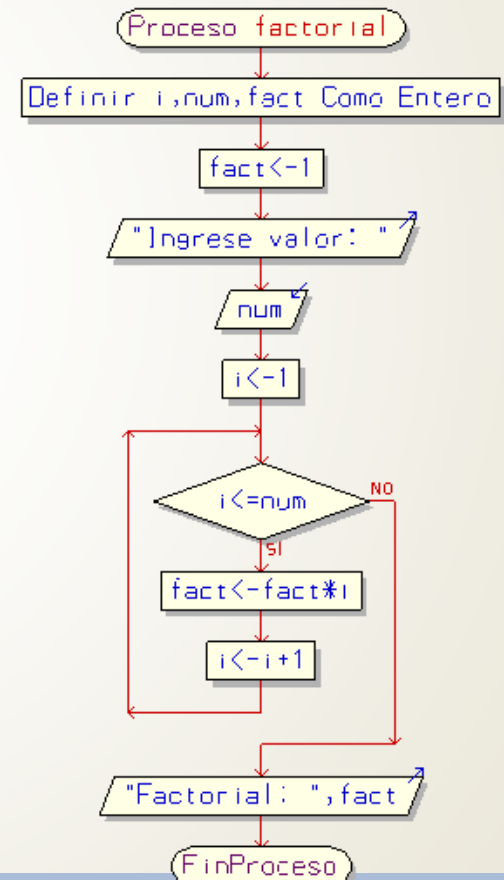
PARA

Estructuras Repetitivas (3)

- Diseñe un algoritmo que calcule el factorial de un número ingresado por el usuario.

```

PROGRAMA factorial
VARIABLES
    i,num, fact: Entero
INICIO
    fact<-1
    ESCRIBIR "Ingrese valor: "
    LEER num
    i<-1
    MIENTRAS i <= num HACER
        fact<-fact*i
        i<-i+1
    FIN_MIENTRAS
    ESCRIBIR "Factorial: ", fact
FIN
  
```



Estructuras Repetitivas (4)

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

```
using namespace std;
```

```
main()
```

```
{ int num,i=1, fact=1;
  cout << "Ingrese valor: ";
  cin >> num;
```

```
  while (i<=num)
  { fact=fact*i;
    i=i+1;
  }
```

```
  cout << "Factorial: " << fact << endl;
  system("pause");
```

```
}
```

MIENTRAS

```
mientras condición hacer
    acciones
fin_mientras
```

```
while (condición)
{
    acciones;
}
```

Programa que calcula el factorial de un número ingresado por el usuario, utilizando estructuras MIENTRAS.

MIENTRAS

Estructuras Repetitivas (5)

- Diseñe un algoritmo que calcula la suma de valores ingresados por el usuario hasta que se introduce un CERO.

PROGRAMA suma_valores

VARIABLES

num, suma: ENTERO

INICIO

suma ← 0

REPETIR

ESCRIBIR "Ingrese un valor: "

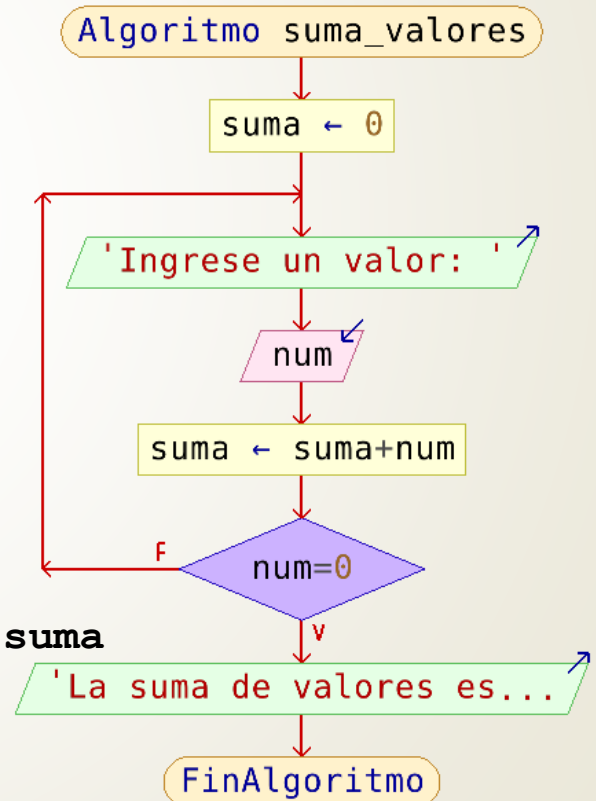
LEER num

suma ← suma + num

HASTA_QUE num = 0

ESCRIBIR "La suma de valores es: ", suma

FIN



Estructuras Repetitivas (6)

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

```
using namespace std;
```

```
main()
```

```
{ int num, suma=0;
```

```
do
```

```
{ cout << "Ingrese un valor: ";
```

```
  cin >> num;
```

```
  suma=suma+num;
```

```
} while (num!=0);
```

```
cout << "La suma de valores es: " << suma << endl;
```

```
system("pause");
```

```
}
```

REPETIR

repetir

acciones

hasta_que condición

Do

{

acciones;

} while (condición);

Programa que calcula la suma de valores ingresados por el usuario hasta que se introduce un CERO.

REPETIR

Estructuras Repetitivas (7)

- Modifique el algoritmo anterior de modo que utilice el concepto de bandera para finalizar el bucle.

```
#include <iostream>
#include <stdlib.h>
using namespace std;
main()
{ float num, suma=0;
  bool seguir;
  do
  { cout << "Ingrese valor: ";
    cin >> num;
    suma=suma+num;
    if (num==0)
      seguir=false;
    else
      seguir=true;
  } while(seguir==true);
  cout << "La suma es: " << suma << endl;
  system("pause");
}
```

- La variable lógica **seguir** permite detectar en qué momento se ingresa un dato cero.
- **seguir** es VERDADERA si el dato ingresado es distinto de cero.
- **seguir** es FALSA cuando el valor ingresado es igual a cero.
- el bucle finaliza cuando **seguir** es FALSA (num=0), ya que la condición de repetición no se cumple.