Sesión 03

Algorítmica I

Sesión 3

- Definición de constantes
- Bibliotecas del C++
- Instrucciones condicionales (IF/THEN/ELSE)
- Instrucciones condicionales anidadas
- Instrucciones Iterativas (While)
  - Casos de Uso
- Instrucciones Iterativas (Do…While)
  - Casos de Uso

Tipos de Datos

Los tipos de datos en C++ se clasifican en primitivos y derivados.

T**ipos de datos primitivos** son los que están definidos dentro del lenguaje.

numéricos enteros, numéricos reales, tipo lógico, caracter

T**ipos de datos derivados** se forman a partir de los tipos primitivos.

Constantes en C++, const y #define

- Las datos constantes, tienen un valor fijo durante toda la ejecución del programa, es decir, este valor no cambia ni puede ser cambiado a lo largo de la ejecución de nuestro programa.
- Para declarar una constante, se hace después de declarar las librerías y antes de las funciones, la sintaxis es la siguiente:

  #define nombre_constante valor.

- En C++ se pueden definir constantes de dos forma, ambas válidas para nosotros.
  - La primera es por medio del comando *#define nombre_constante valor* y
  - la segunda es usando la palabra clave *const*,

Uso de #define para declarar constantes en C++

- La instrucción *#define* nos permite declarar constantes (y algunas cosas más) de una manera rápida y sencilla.
- Hay que tener en cuenta que al declarar constantes con *#define* debemos hacerlo despues de los *#include* para importar librerías pero antes de declarar nuestras funciones y demás.

```
#include <iostream>
using namespace std;
#define PI 3.1416; //Definimos una constante llamada PI
int main()
{
cout << "Mostrando el valor de PI: " << PI;
return 0;
}
```

Uso de #define para declarar constantes en C++

```cpp
#include <iostream>
using namespace std;
#define PI 3.1416; //Definimos una constante llamada PI
int main()
{
cout << "Mostrando el valor de PI: " << PI << endl;
return 0;
}
```

- Si intentamos ejecutar el código anterior obtendremos un error al haber usado el operador << justo despues de PI, esto sucede porque PI no es tratado exactamente como una variable cualquiera sino como una expresión, así que realmente aunque podemos usar *#define* para declarar constantes no es la mejor opción..

Uso de const para declarar constantes en C++

```cpp
#include <iostream>
using namespace std;
int main()
{
const float PI = 3.1416; //Definimos una constante llamada PI
cout << "Mostrando el valor de PI: " << PI << endl;
return 0;
}
```

```cpp
#include <iostream>
using namespace std;
int main()
{
const float PI = 3.1416; //Definimos una constante llamada PI
cout << "Mostrando el valor de PI: " << PI << endl;
PI = 2; //Esto generará un error pues PI es de solo lectura (constante)
return 0;
}
```

Dado el siguiente código, define los valores de x, y, z.

```
int x, y, z;
cin>>x;
cin>> y;
cin>>z;

if(( y < x) && (z > y))
        if(x%y>0)
                z++;
        else {
                y++;
                x–;
        }
```

Operadores

OPERADORES DE ASIGNACIÓN

Operador	Acción	Ejemplo	Resultado
=	Asignación Básica	X = 6	X vale 6
*=	Asigna Producto	X *= 5	X vale 30
/=	Asigna División	X /= 2	X vale 3
+=	Asigna Suma	X += 4	X vale 10
-=	Asigna Resta	X -= 1	X vale 5
%=	Asigna Modulo	X %= 5	X vale 1

OPERADORES ARITMÉTICOS

Operador	Acción	Ejemplo	Resultado
-	Resta	X = 5 - 3	X vale 2
+	Suma	X = 5 + 3	X vale 8
*	Multiplicación	X = 2 * 3	X vale 6
/	División	X = 6 / 3	X vale 2
%	Módulo	X = 5 % 2	X vale 1
--	Decremento	X = 1; X--	X vale 0
++	Incremento	X = 1; X++	X vale 2

```
int x,y;
x = 2004;
y = ++x;
/* x e y valen 2005. */
```

```
int x,y
x = 2004;
y = x++;
/* y vale 2004 y x vale 2005 */
```

OPERADORES RELACIONALES

Los operadores relacionales, también denominados operadores binarios lógicos y de comparación, se utilizan para comprobar la veracidad o falsedad de determinadas propuestas de relación (en realidad se trata respuestas a preguntas).
Las expresiones que los contienen se denominan expresiones relacionales. Aceptan diversos tipos de argumentos, y el resultado, que es la respuesta a la pregunta, es siempre del tipo cierto/falso, es decir, producen un resultado booleano.
Si la propuesta es cierta, el resultado es true (un valor distinto de cero), si es falsa será false (cero)

Operador	Relación	Ejemplo	Resultado
<	Menor	X = 5; Y = 3; if(x < y) x+1;	X vale 5 Y vale 3
>	Mayor	X = 5; Y = 3; if(x > y) x+1;	X vale 6 Y vale 3
<=	Menor o igual	X = 2; Y = 3; if(x <= y) x+1;	X vale 3 Y vale 3
>=	Mayor o igual	X = 5; Y = 3; if(x >= y) x+1;	X vale 6 Y vale 3
==	Igual	X = 5; Y = 5; if(x == y) x+1;	X vale 6 Y vale 5
¡=	Diferente	X = 5; Y = 3; if(x ¡= y) y+1;	X vale 5 Y vale 4

OPERADORES LÓGICOS

Los operadores lógicos producen un resultado booleano, y sus operandos son también valores lógicos o asimilables a ellos (los valores numéricos son asimilados a cierto o falso según su valor sea cero o distinto de cero)

Operador	Acción	Ejemplo	Resultado
&&	AND Lógico	A && B	Si ambos son verdaderos se obtiene verdadero(true)
||	OR Lógico	A || B	Verdadero si alguno es verdader
!	Negación Lógica	!A	Negación de a

BIBLIOTECAS

- Las bibliotecas contienen el código objeto de muchos programas que permiten hacer cosas comunes, como leer el teclado, escribir en la pantalla, manejar números, realizar funciones matemáticas, etc.
- Las bibliotecas están clasificadas por el tipo de trabajos que hacen, hay bibliotecas de entrada y salida, matemáticas, de manejo de memoria, de manejo de textos y como imaginarás existen muchísimas librerías disponibles y todas con una función especifica.
- Hay un conjunto de bibliotecas (o librerías) muy especiales, que se incluyen con todos los compiladores de C y de C++. Son las librerías (o bibliotecas) ANSI o estándar

Sintaxis para declarar Librerías en C++

- La declaración de librerías, tanto en C como en C++, se debe hacer al principio de todo nuestro código, antes de la declaración de cualquier función o línea de código,
- Debemos indicarle al compilador que librerías usar, para el saber que términos estarán correctos en la escritura de nuestro código y cuáles no.
- La sintaxis es la siguiente**: #include <nombre de la librería>** o alternativamente **#include "nombre de la librería".**

Librerías Estandar de C++ (Standar Template Library o *STL*)

- **fstream:** Flujos hacia/desde ficheros. Permite la manipulación de archivos desde el programar, tanto leer como escribir en ellos.
- **iosfwd:** Contiene declaraciones adelantadas de todas las plantillas de flujos y sus typedefs estándar. Por ejemplo ostream.
- **iostream:** Parte del a *STL* que contiene los algoritmos estándar, es quizá la más usada e importante (aunque no indispensable).
- **La biblioteca list:** Parte de la *STL* relativa a contenedores tipo list; listas doblemente enlazadas
- **math:** Contiene los prototipos de las funciones y otras definiciones para el uso y manipulación de funciones matemáticas.
- **memory:** Utilidades relativas a la gestión de memoria, incluyendo asignadores y punteros inteligentes (*auto_ptr*).
- **typeinfo:** Mecanismo de identificación de tipos en tiempo de ejecución
- **vector:** Parte de la *STL* relativa a los contenedores tipo vector; una generalización de las matrices unidimensionales C/C++

Librerías Estandar de C++ (Standar Template Library o *STL*)

- **Biblioteca new:** Manejo de memoria dinámica
- **numeric:** Parte de la librería numérica de la *STL* relativa a operaciones numéricas.
- **ostream:** Algoritmos estándar para los flujos de salida.
- **queue:** Parte de la *STL* relativa a contenedores tipo queue (colas de objetos).
- **Librería stdio:** Contiene los prototipos de las funciones, macros, y tipos para manipular datos de entrada y salida.
- **Librería stdlib:** Contiene los prototipos de las funciones, macros, y tipos para utilidades de uso general.
- **string:** Parte de la *STL* relativa a contenedores tipo string; una generalización de las cadenas alfanuméricas para albergar cadenas de objetos. Muy útil para el fácil uso de las cadenas de caracteres, pues elimina muchas d elas dificultades que generan los char
- **typeinfo:** Mecanismo de identificación de tipos en tiempo de ejecución
- **vector:** Parte de la *STL* relativa a los contenedores tipo vector; una generalización de las matrices unidimensionales C/C++

¿Cómo declarar una librería en C++?

```
#include "iostream"
#include "string"
#include <math.h>
#include <conio.h>
using namespace std;
```

```
#include <stdlib.h>
#include <iostream>
using namespace std;
int main ()
{
cout << ("Se va a generar un numero aleatorio ....\n");
cout << ("El numero generado es : ");
cout << rand(); //Se genera el número con rand y se muestra en pantalla
return 0;
}
```

hemos hecho uso de dos librerías: iostream y stdlib. La librería o biblioteca iostream, nos permitirá hacer uso del cin y el cout para obtener o imprimir valores por pantalla, respectivamente mientras stdlib nos dará acceso a la función rand que generará por nosotros un número cualquiera.

```cpp
#include <string.h>
#include <iostream>
using namespace std;
int main ()
{
cout << ("Hola! Por favor ingrese su nombre ....\n");
string cadena = "Hola "; //Se le da un valor inicial al string
string nombre; //Esta cadena contendrá el nombre
cin >> nombre; //Se lee el nombre
cadena = cadena + nombre; //Se juntan el saludo con el nombre usando "+"
cout << (cadena); //Se muestra el resultado final.
return 0;
}
```

Aquí hemos mostrado un mensaje solicitando el nombre al usuario y luego usando string, hemos creado un saludo que incluya el nombre del usuario. "Hola Juan".

Instrucciones Condicionales

Existen diferentes tipos de condicionales, cada uno tiene una utilidad y funcionalidad diferente, que consideran diferentes situaciones que se pueden llegar a presentar durante la ejecución de un algoritmo.

- Condicional If
- Condicional if-else
- Condicional Switch

Condicional if

- Los condicionales if, son una estructura de control condicional, que nos permiten tomar cierta decisión al interior de nuestro algoritmo, es decir, nos permiten determinar que acciones tomar dada condición, por ejemplo determinar si un numero cualquiera es mayor que 10 y de acuerdo a esto mostrar un mensaje.

- En resumen, un condicional if es una estructura que nos posibilita definir las acciones a ejecutar si se cumple cierta condición.

¿Cómo funciona un Condicional If?

Supongamos que queremos verificar si el resultado de una suma ingresada por el usuario es correcto o no. Para este ejemplo, el condicional if, es el encargado de verificar si el resultado ingresado corresponde o no a la respuesta correcta de la suma. El condicional if, funciona verificando la condición ingresada y de acuerdo a su valor de verdad (falso o verdadero) lleva a cabo o no una serie de instrucciones.

```
if(condición a evaluar) //Por ejemplo X <= 10
{
....
....
Bloque de Instrucciones si se cumple la condición....
....
....
}
....
Bloque de Instrucciones restante DEL ALGORITMO....
....
```

Condicional if else

- Los condicionales if-else, son una estructura de control, que nos permiten tomar cierta decisión al interior de nuestro algoritmo, es decir, nos permiten determinar que acciones tomar dada o no cierta condición, por ejemplo determinar si la contraseña ingresada por el usuario es válida o no y de acuerdo a esto darle acceso al sistema o mostrar un mensaje de error.
- En resumen, un condicional if-else es una estructura que nos posibilita definir las acciones que se deben llevar a cabo si se cumple cierta condición y también determinar las acciones que se deben ejecutar en caso de que no se cumpla.

```
if(condición a evaluar) //Por ejemplo 50 <= 10
{
....
....
Bloque de Instrucciones si se cumple la condición....
....
....
}
else
{
....
....
Bloque de Instrucciones si NO se cumple la condición....
....
....
}
```

Introducción a la Programación

Estructura Selectiva Simple

Si Entonces

Pseudocodigo

```
Si condición entonces
    Operaciones
Fin Si
```

Diagrama de Flujo

Introducción a la Programación

Estructura Selectiva Simple

Si Entonces

```
Si condición entonces
    Operaciones
Fin Si
```

Donde:

CONDICIÓN expresa la condición o conjunto de condiciones a evaluar.

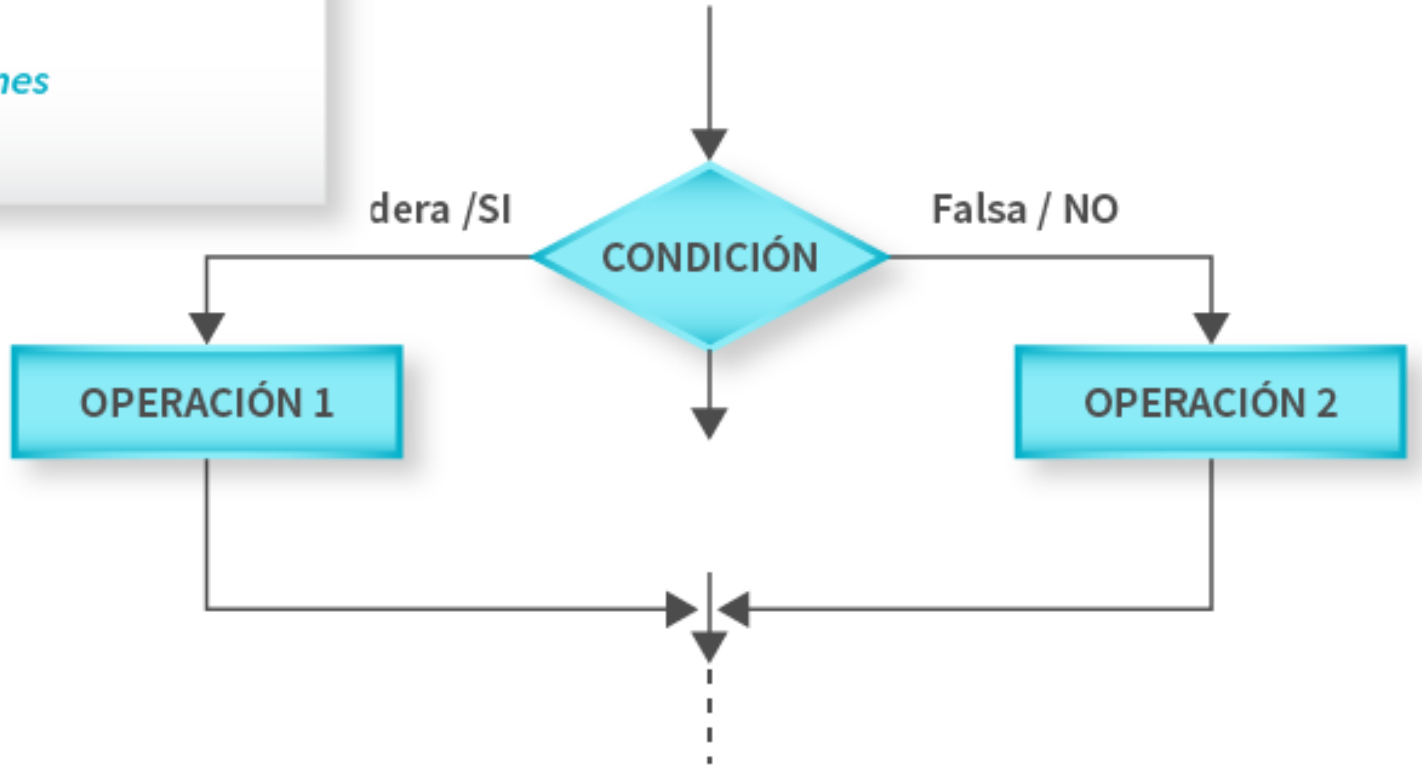
OPERACIÓN expresa la operación o conjunto de operaciones que se van a realizar si la condición resulta verdadera.

Introducción a la Programación

Estructura Selectiva Doble

Si Entonces / Sino

```
Si condición entonces
    Operaciones
Sino
    Operaciones
Fin Si
```

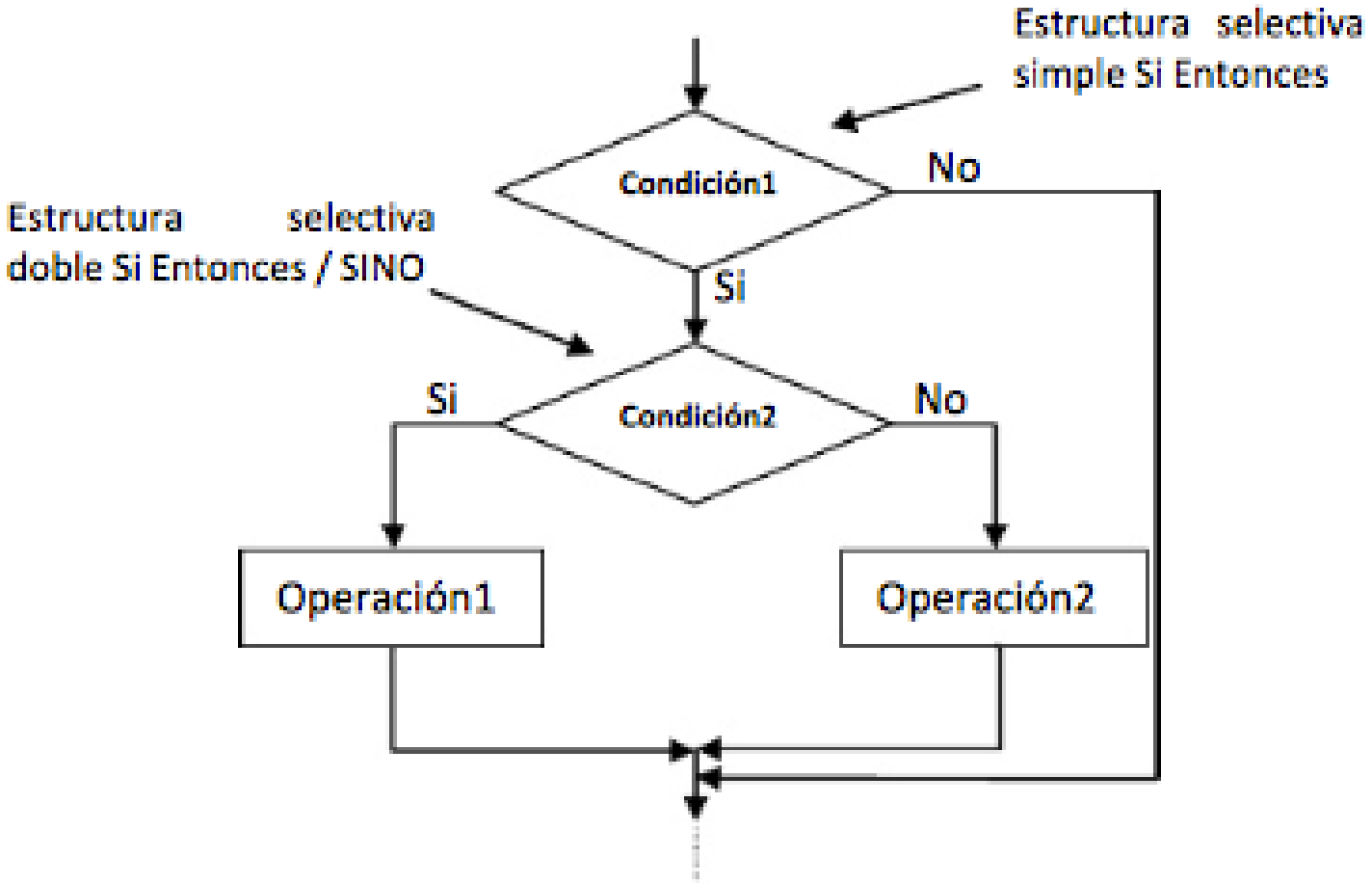


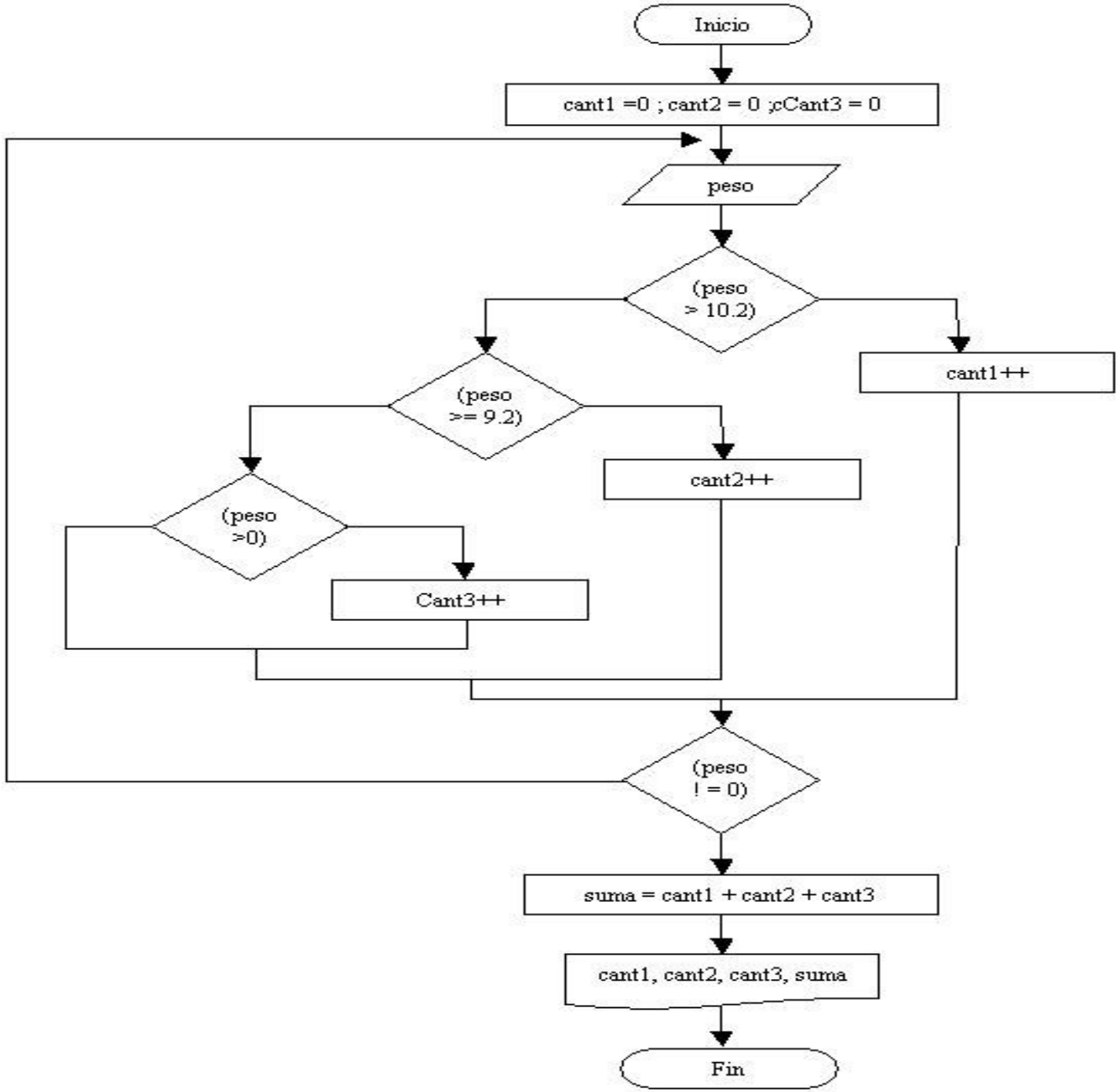
Introducción a la Programación

Estructura Selectiva En Cascada

ANIDADAS

Diagrama de Flujo




Inicio
cant1 =0 ; cant2 = 0 ;cCant3 = 0
peso
(peso
> 10.2)
cant1++
(peso
>= 9.2)
cant2++
(peso
>0)
Cant3++
(peso
! = 0)
suma = cant1 + cant2 + cant3
cant1, cant2, cant3, suma
Fin

Introducción a la Programación

Ejemplo: Estructura Selectiva En Cascada

ANIDADAS

Dado como dato tres números, que representan números enteros diferentes, construya el pseudocodigo y diagrama de flujo para escribir estos números en forma descendente.

Solución

Entrada	Proceso	Salida
N1, N2, N3	*Ordenar los números en forma descendente*	*Mostrar los números en forma Descendente*

Condicional switch

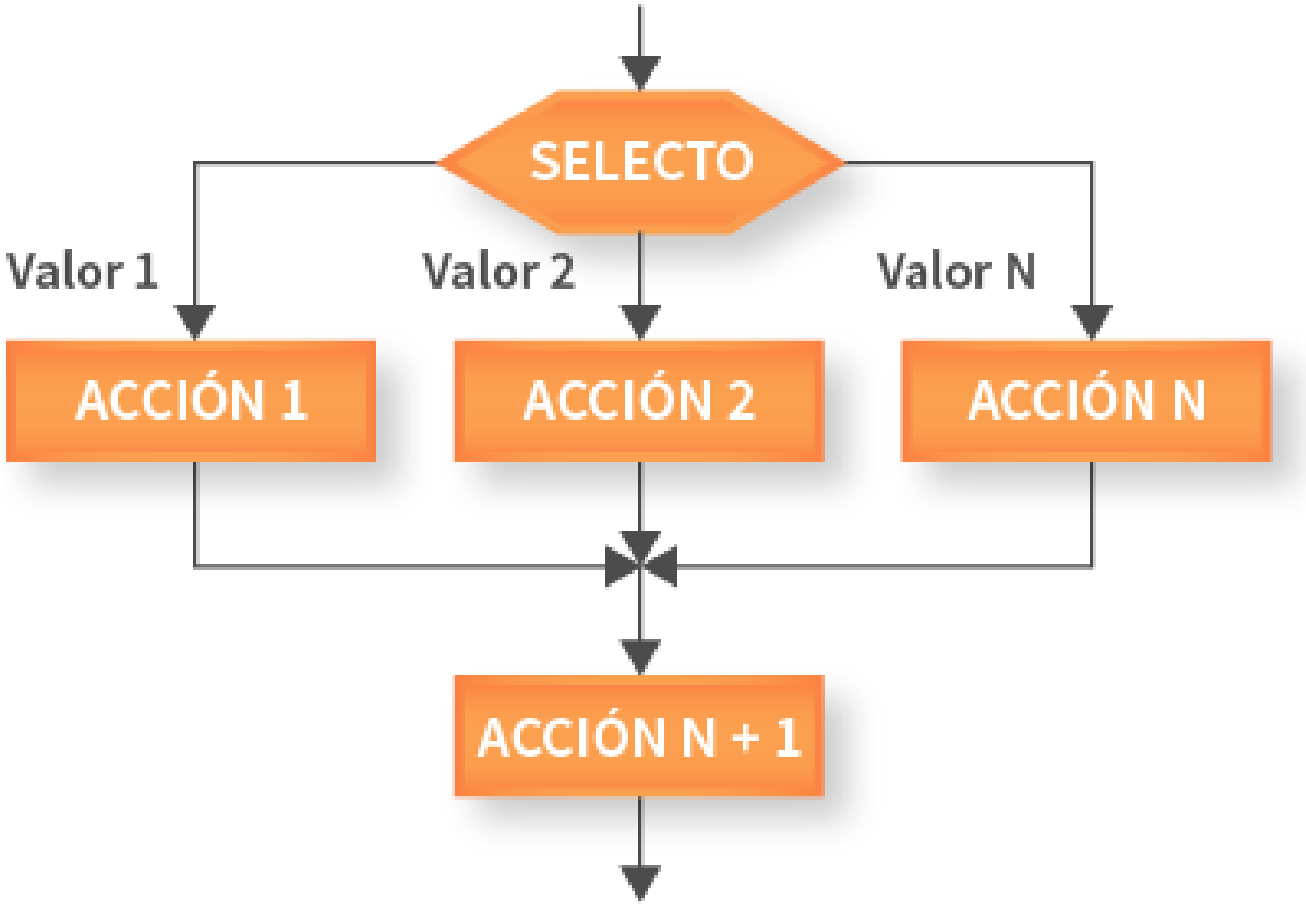
- Los condicionales Switch, son una estructura de control condicional, que permite definir múltiples casos que puede llegar a cumplir una variable cualquiera, y qué acción tomar en cualquiera de estas situaciones, incluso es posible determinar qué acción llevar a cabo en caso de no cumplir ninguna de las condiciones dadas.

```
switch(opción) //donde opción es la variable a comparar
{
case valor1: //Bloque de instrucciones 1;
break;
case valor2: //Bloque de instrucciones 2;
break;
case valor3: //Bloque de instrucciones 3;
break;
//Nótese que valor 1 2 y 3 son los valores que puede tomar la opción
//la instrucción break es necesaria, para no ejecutar todos los casos.
default: //Bloque de instrucciones por defecto;
//default, es el bloque que se ejecuta en caso de que no se de ningún caso
}
```

Introducción a la Programación

Estructura Selectiva Múltiple

Si Múltiple



```
cout << "Ingrese la Opción a ejecutar: ";
int opcion = 0;
cin >> opcion;
switch(opcion)
{
case 1: cout << "Usted ha seleccionado la opción 1";
break;
case 2: cout << "Usted ha seleccionado la opción 2";
break;
case 3: cout << "Usted ha seleccionado la opción 3";
break;
default: cout << "Usted ha ingresado una opción incorrecta";
}
```

```
# include "iostream"
using namespace std;
void main()
{
cout << "Ingrese la Opción a ejecutar: ";
int opcion = 0;
cin >> opcion;
switch(opcion)
{
case 1: cout << "Usted ha seleccionado la opción 1";
break;
case 2: cout << "Usted ha seleccionado la opción 2";
break;
case 3: cout << "Usted ha seleccionado la opción 3";
break;
default: cout << "Usted ha ingresado una opción incorrecta";
}
system("PAUSE");
}
```

Ejercicios

1. **Que pida un número y diga si es par o impar.**
2. **Que pida un número del 1 al 7 y diga el día de la semana correspondiente.**
3. **Que pida un número del 1 al 12 y diga el nombre del mes correspondiente.**

Ejercicio

Que calcule el sueldo que le corresponde al trabajador de una empresa que cobra 40.000 soles anuales, el programa debe realizar los cálculos en función de los siguientes criterios:

a. Si lleva más de 10 años en la empresa se le aplica un aumento del 10%.
b. si lleva menos de 10 años pero más que 5 se le aplica un aumento del 7%.
c. Si lleva menos de 5 años pero más que 3 se le aplca un aumento del 5%.
d. Si lleva menos de 3 años se le aplica un aumento del 3%.

Estructuras Algorítmicas Iterativas

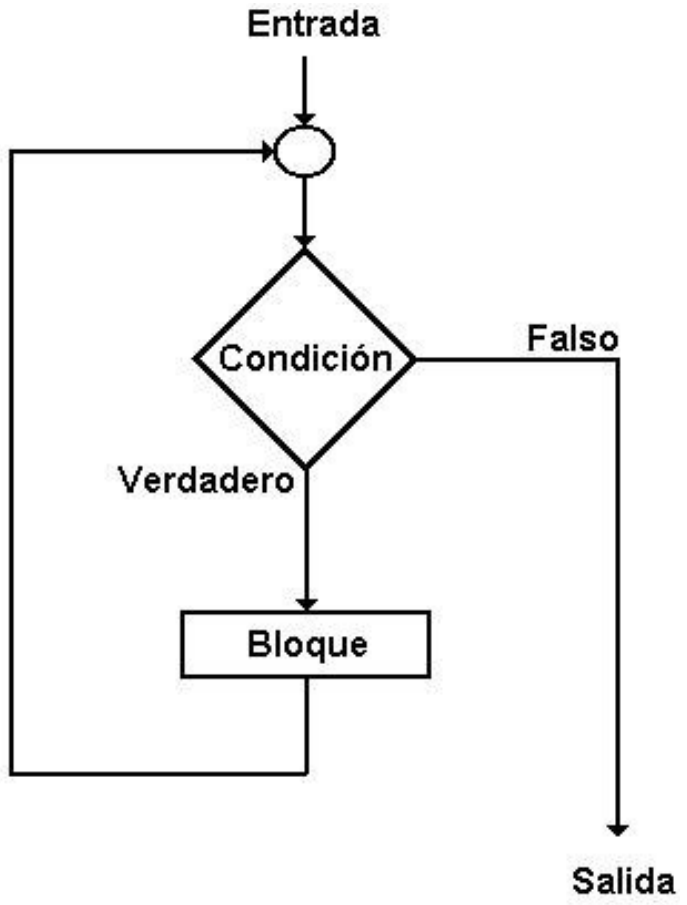
- while
- do..while
- for

Introducción a la Programación

Estructura Repetitiva Mientras

While

Diagrama de Flujo



Introducción a la Programación

Ejemplo: Estructura Repetitiva Mientras

Desarrolle un programa en el cual se calcule el aumento de sueldo para un grupo de empleado de una empresa, teniendo en cuenta los siguientes criterios.

- ✓ Si el sueldo es inferior a $ 1,000.00 = Aumento 15%
- ✓ Si el sueldo es mayor o igual a $ 1,000.00 = Aumento 12%

Imprimir el sueldo nuevo del Empleado y el total de nomina de la empresa, considerando este nuevo aumento.

Solución

Entrada	Proceso	Salida
Sueldo	*Verificar si el sueldo es <, >= 1000* *NuevoSueldo = Sueldo * %* *Nomina = Nomina + NuevoSueldo*	*NuevoSueldo* *Nomina*

Estructura Repetitiva Mientras

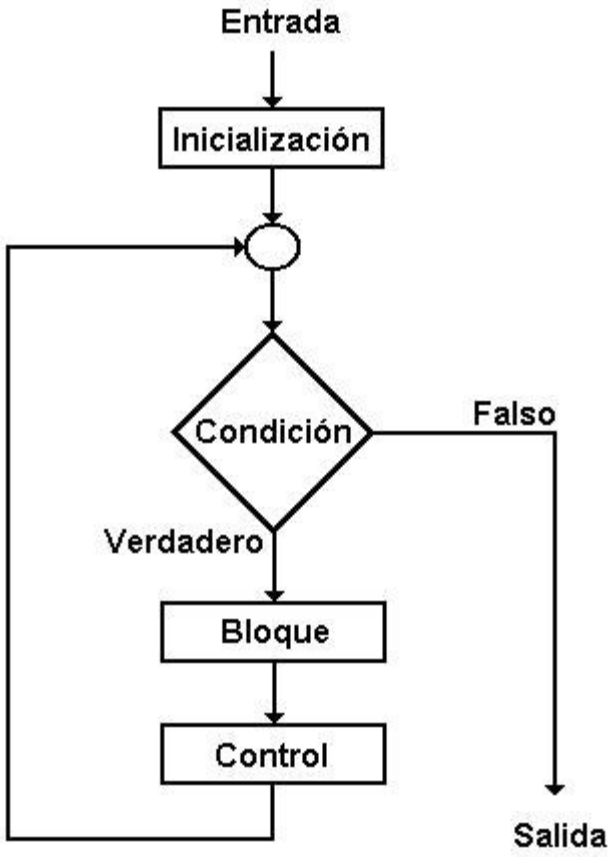


Figura 3.6

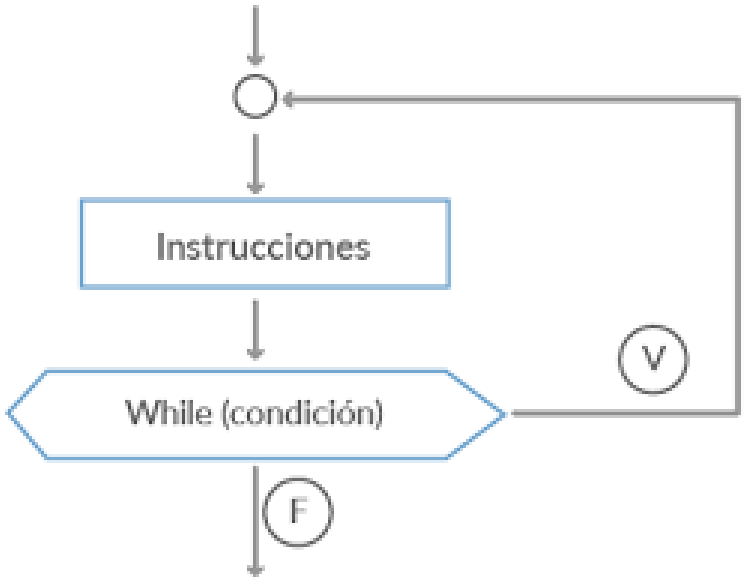
```
#include <iostream>
using namespace std;
int main(){
   int cont = 1,num;
   cin>>num;
   while(cont<=10){
      cout<<num*cont<<endl;
      cont++;//cont=cont+1;
   }

   return 0;
}
```

```
C:\Users\Gianm
10
10
20
30
40
50
60
70
80
90
100

Process retur
Press any key
```

Estructura Repetitiva Hacer… Mientras




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

   do{
      cout<<num*cont<<endl;
      cont++;
   }while(cont<=10);

   return 0;
}
```

```
C:\User
10
10
20
30
40
50
60
70
80
90
100

Process
Press ar
```

Cómo Construir un Menú de Opciones

- Usando switch..case
- Usando do..while