



[ucontinental.edu.pe](http://ucontinental.edu.pe)



# Fundamentos de Programación

Carol R. Rojas Moreno  
Manual – Unidad 1

## Índice

Introducción .....	4
Organización de la Asignatura .....	5
Unidades didácticas.....	5
Tiempo mínimo de estudio .....	5
UNIDAD 1:.....	6
Estructuras de control para la programación: Secuencial y Selectiva .....	6
Diagrama de organización .....	6
Tema n.º 1: Algoritmo: Definición, Características y Representación .....	6
Sub tema 1.1 Definición de Algoritmo .....	7
Sub tema 1.2 Características de un algoritmo (Joyanes Luis, 2008) .....	8
Sub tema 1.3 Instrucciones Algorítmicas Básicas .....	8
Sub tema 1.4 Representación del Algoritmo .....	9
Tema n.º 2: Variables y Tipos de Datos.....	14
Sub tema 2.1 Definición de Variable .....	14
Sub tema 2.2 Tipos de Datos.....	16
De la teoría a la práctica .....	18
Tema n.º 3: Introducción a la Programación:.....	20
Programas Traductores, Programación Estructurada.....	20
Sub tema 3.1 Definición de Programa .....	20
Sub tema 3.2 Lenguaje de Programación .....	21
Sub tema 3.3 Programa Traductor .....	22
Sub tema 3.4 Programación Estructurada .....	23
Tema n.º 4: Estructuras de control para la programación: .....	26
Estructuras de control Secuencial.....	26
Sub tema 4.1 Definición de Estructuras de Control Secuencial .....	26
De la teoría a la práctica .....	27
Tema n.º 5: Estructuras de control para la programación: .....	32
Estructuras de control Selectiva Simple. ....	32
Sub tema 5.1 Definición de Estructuras de Control Selectiva .....	32
Sub tema 5.2 Definición de Estructuras de Control Selectiva Simple .....	33
De la teoría a la práctica .....	33
Tema n.º 6: Estructuras de control para la programación: .....	36
Estructuras de control Selectiva Compuesta.....	36
Sub tema 6.1 Definición de Estructuras de Control Selectiva Compuesta .....	36
De la teoría a la práctica .....	37

Tema n.º 7: Estructuras de control para la programación: .....	40
Estructuras de control Selectiva Múltiple. ....	40
Sub tema 7.1 Definición de Estructuras de Control Selectiva Múltiple .....	40
De la teoría a la práctica .....	40
Glosario de la Unidad 1 .....	47
Bibliografía de la Unidad 1 .....	48

## Introducción

Fundamentos de Programación es una asignatura que tiene como finalidad proporcionar al estudiante, los conocimientos necesarios en las técnicas para la programación basada en el enfoque estructurado, requeridos en su formación básica para poder desarrollar programas en otros niveles más avanzados. El resultado de aprendizaje: Al finalizar la asignatura el estudiante será capaz de implementar programas para computadora con la finalidad de resolver problemas de ingeniería (Rojas Moreno, 2013).

El presente material consta de cuatro unidades: Unidad I: Estructuras de control para la programación: Secuencial y Selectiva. Unidad II: Estructuras de control para la programación: Repetitiva. Unidad III: Módulos para la programación: Función y Procedimiento. Unidad IV: Módulos para la programación: Funciones propias del Lenguaje de Programación – Recursividad, desarrollados a partir del texto Fundamentos de Programación (Joyanes Aguilar, 2008).

Es recomendable que el estudiante desarrolle una permanente lectura de estudio junto a la elaboración de programas en un lenguaje de programación, así como la investigación en otros textos y vía internet. El contenido del material se complementará con las lecciones presenciales y a distancia que se desarrollan en la asignatura.

Agradecemos a quienes con sus aportes y sugerencias han contribuido a mejorar la presente edición, que sólo tiene el valor de una introducción al conocimiento de las estructuras de datos para la programación en un computador.

La autora.

## Organización de la Asignatura

<b>Resultado de aprendizaje de la asignatura</b>
Al finalizar la asignatura, el estudiante será capaz de implementar programas para computadora con la finalidad de resolver problemas de ingeniería.

### Unidades didácticas

<b>UNIDAD 1</b>	<b>UNIDAD 2</b>	<b>UNIDAD 3</b>	<b>UNIDAD 4</b>
Estructuras de control para la programación: Secuencial y Selectiva.	Estructuras de control para la programación: Repetitiva.	Módulos para la programación: Función y Procedimiento.	Módulos para la programación: Funciones propias del Lenguaje de Programación – Recursividad
<b>Resultado de aprendizaje</b>  Al finalizar la unidad, el estudiante será capaz de aplicar estructuras de control secuencial y selectivas, en la construcción de programas computacionales usando un lenguaje de programación .	<b>Resultado de aprendizaje</b>  Al finalizar la unidad, el estudiante será capaz de aplicar estructuras de control repetitivas, en la construcción de programas computacionales usando un lenguaje de programación.	<b>Resultado de aprendizaje</b>  Al finalizar la unidad, el estudiante será capaz de aplicar módulos de programación y librerías de programación, en la construcción de programas computacionales.	<b>Resultado de aprendizaje</b>  Al finalizar la unidad, el estudiante será capaz de aplicar las funciones propias del lenguaje de programación, en la construcción de programas computacionales.

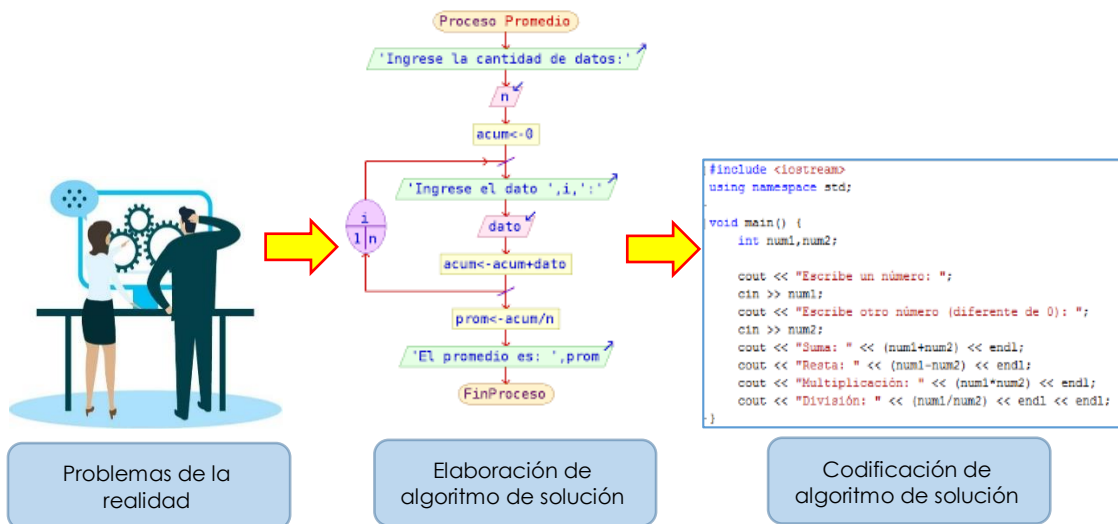
### Tiempo mínimo de estudio

<b>UNIDAD 1</b>	<b>UNIDAD 2</b>	<b>UNIDAD 3</b>	<b>UNIDAD 4</b>
24 horas	24 horas	24 horas	24 horas

# UNIDAD 1:

## Estructuras de control para la programación: Secuencial y Selectiva

### Diagrama de organización



### Tema n.º 1: Algoritmo: Definición, Características y Representación

En la realidad se tiene diferentes situaciones que pueden ser descritas como un conjunto de pasos en una determinada secuencia, ya sea en la vida cotidiana y otras pertenecientes a las actividades de las organizaciones.

Para iniciar el presente tema, se tiene presente que estas actividades pueden resultar conocidas por muchos debido a su frecuencia de utilización, y como otras actividades que pertenecen a un proceso y a un conjunto de actores propietarios, por lo que debe considerar revisar y observar todas las actividades próximas en su quehacer personal, a fin de facilitar la creación de algoritmos y su posterior codificación en un lenguaje de programación.

## Sub tema 1.1 Definición de Algoritmo

Se define a un algoritmo como un conjunto ordenado y finito de pasos (actividades, instrucciones) que conducen a la solución de un problema.

Por ejemplo:

- Al instalar un equipo de sonido ejecutamos las instrucciones contenidas en su manual del equipo.
- Algoritmo para sumar dos números.

Si un algoritmo puede ser elaborado y ejecutado por una computadora, se dice que es un algoritmo computacional.

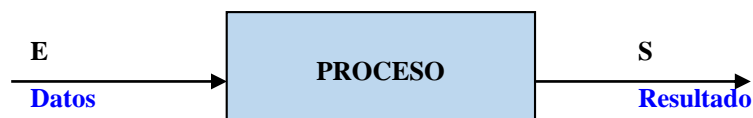
Para que un algoritmo sea computacional se debe expresar en instrucciones a través de un lenguaje de programación, teniendo como resultado un programa.

Todo algoritmo puede ser descompuesto en tres partes:

Entrada de datos.

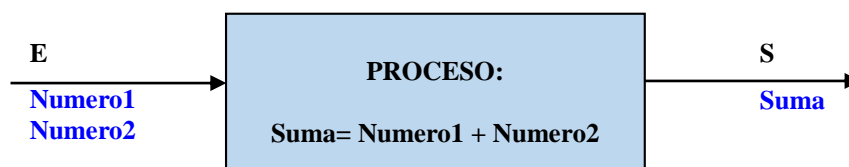
Proceso.

Salida de resultados.



En cada uno de ellos se necesita el uso de variables, por ejemplo, en la entrada de datos, que a través del proceso (algoritmo), ordenaran, buscaran, calcularan, para luego reportar la salida como información, útil y significativa para el usuario.

Como ejemplo, la suma de dos números:



**Sub tema 1.2 Características de un algoritmo** (Joyanes Luis, 2008)**Ser preciso**

los pasos del algoritmo deben desarrollarse en un orden estricto (orden lógico).

**Ser finito**

implica que el número de pasos de un algoritmo debe ser limitado

**Ser definido**

el algoritmo desarrollará las tareas programadas no puede improvisar.

**Presentación formal**

para que el algoritmo sea entendido es necesario que se exprese en alguna de las formas comúnmente aceptadas; por ejemplo: el pseudocódigo, diagrama de flujo y diagramas de Nassi/Schneiderman.

**Eficiencia:**

es evaluar los recursos que requiere para almacenar datos y para ejecutar operaciones frente al beneficio que ofrece.

**Corrección**

para garantizar que el algoritmo logre el objetivo, es necesario ponerlo a prueba: verificación o prueba de escritorio.

**Sub tema 1.3 Instrucciones Algorítmicas Básicas**

**a. Entrada:** Consiste en obtener un dato de un dispositivo de entrada, como el teclado, y almacenarlo en una variable. En general, la acción de ingresar un dato a una **variable** se expresa en el pseudocódigo mediante la palabra **LEER** (ingresar, obtener), de la siguiente forma:

**LEER** variable

Ejemplo: **LEER** edad

Si se emplea un lenguaje de programación, la instrucción LEER es de acuerdo a la sintaxis, ejemplo en el lenguaje C/C++:

Ejemplo: **cin>>** edad;

**b. Salida:** Consiste en mostrar el valor de una variable en un dispositivo de salida, como la pantalla. En general, la acción de mostrar el valor de una **variable** se expresa en el pseudocódigo mediante la palabra **IMPRIMIR** (Escribir, mostrar, reportar) de la siguiente forma:

**IMPRIMIR** variable

Ejemplo: **IMPRIMIR** TotalCompra

Si se emplea un lenguaje de programación, la instrucción IMPRIMIR es de acuerdo a la sintaxis, ejemplo en el lenguaje C/C++:

Ejemplo: **cout<<** edad;



**c. Asignación:** Consiste en dar a una **variable** el valor de una **expresión**. La asignación se expresa en el pseudocódigo de la siguiente forma:

variable = expresión      o      variable ← expresión

Donde **variable** y el valor de **expresión** deben tener el mismo tipo de dato.

Ejemplo:

resultado ← numero1 + numero 2

Si se emplea un lenguaje de programación, la instrucción ASIGNACIÓN es de acuerdo a la sintaxis, ejemplo en el lenguaje C/C++:

Ejemplo:      resultado = numero1 + numero 2;

## Sub tema 1.4 Representación del Algoritmo

Existen muchos diagramas de representación del algoritmo, dependiendo de la técnica y enfoque de desarrollo de software (análisis, diseño, implementación) que se esté usando, por ejemplo: diagrama de actividades, diagrama de secuencia, diagramas de colaboración (en el caso del enfoque orientado a objetos), y otros diagramas pertenecientes al enfoque estructurado como:

### 2.1 Pseudocódigo:

Es la representación del algoritmo en lenguaje natural. Ejemplo: sumar dos números enteros.

INICIO  
Tipo de Datos ← Declaración de Variables  
Ingreso de Datos  
Proceso  
Salida de Datos  
FIN

Por ejemplo:

INICIO  
entero ← numero1, numero2, resultado  
Leer numero1  
Leer numero2  
resultado ← numero1 + numero 2  
Escribir "El resultado de la suma es: "  
Escribir resultado  
FIN

Se puede considerar más actividades, ya sea que se puedan realizar de forma manual, como las que se puedan realizar con ayuda de una computadora:

```

1  Proceso ORDENAR_NUMEROS
2      Escribir "INGRESE LOS NUMEROS";
3      Leer N1, N2, N3;
4      Si N1 > N2 Entonces
5          Si N2 > N3 Entonces
6              Escribir 'LOS NUMEROS ORDENADOS ES=',N3,', ',N2,', ',N1;
7          Sino
8              Si N1 > N3 Entonces
9                  Escribir 'LOS NUMEROS ORDENADOS ES=',N2,', ',N3,', ',N1;
10             Sino
11                 Escribir 'LOS NUMEROS ORDENADOS ES=',N2,', ',N1,', ',N3;
12             FinSi
13         FinSi
14     Sino
15         Si N1 > N3 Entonces
16             Escribir 'LOS NUMEROS ORDENADOS ES=',N3,', ',N1,', ',N2;
17         Sino
18             Si N2 > N3 Entonces
19                 Escribir 'LOS NUMEROS ORDENADOS ES=',N1,', ',N3,', ',N2;
20             Sino
21                 Escribir 'LOS NUMEROS ORDENADOS ES=',N1,', ',N2,', ',N3;
22             FinSi
23         FinSi
24     FinSi
25 FinProceso

```

Fuente: <http://victorhugocc.choccac.com/page/4/>

#### Proceso Promedio

```

Escribir "Ingrese la cantidad de datos:"
Leer n

acum<-0

Para i<-1 Hasta n Hacer
    Escribir "Ingrese el dato ",i,":"
    Leer dato
    acum<-acum+dato
FinPara

prom<-acum/n

Escribir "El promedio es: ",prom

```

#### FinProceso

Fuente: <http://pseint.sourceforge.net/index.php?page=ejemplos.php&cual=Promedio&mode=flexible>

```

Proceso notas
definir matricula,prom,total,n1,n2,n3,n4,n5,n6 como real;
escribir "Digite el costo de su matricula";
leer matricula;
Escribir "Digite las ultimas 6 notas de las materias";
Leer n1,n2,n3,n4,n5,n6;
prom:=(n1+n2+n3+n4+n5+n6)/6;

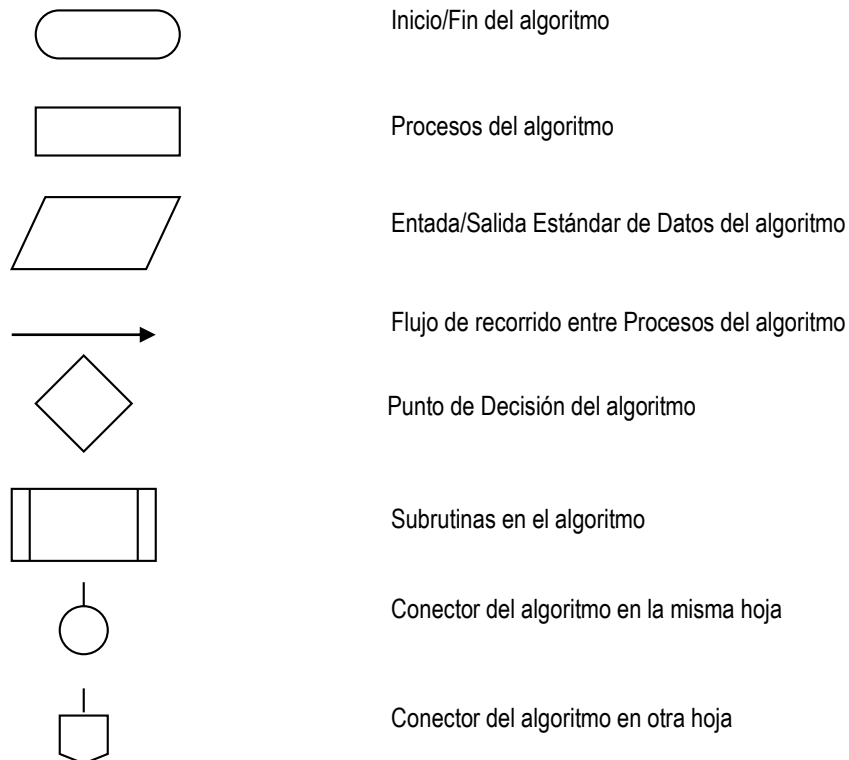
Si prom>=4 Entonces
    total=matricula-matricula*0.3;
Sino
    total=matricula+(matricula*0.1);
finsi
escribir "el valor a pagar es de ",total;
FinProceso

```

Fuente: <https://augustosalazarblog.wordpress.com/category/ejercicios-pseint/>

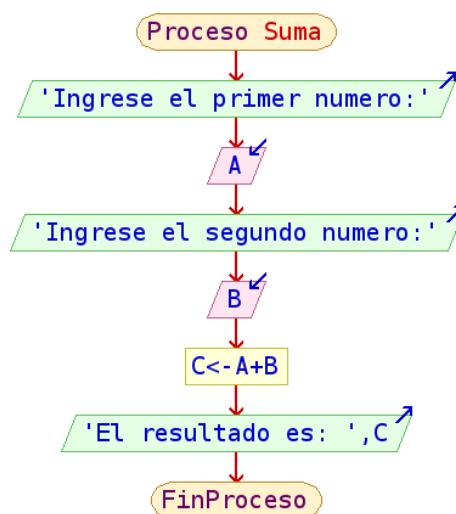
## Diagrama de Flujo Estructurado:

Representación con flujos entre los procesos a realizar. Los principales símbolos de representación para este diagrama son.

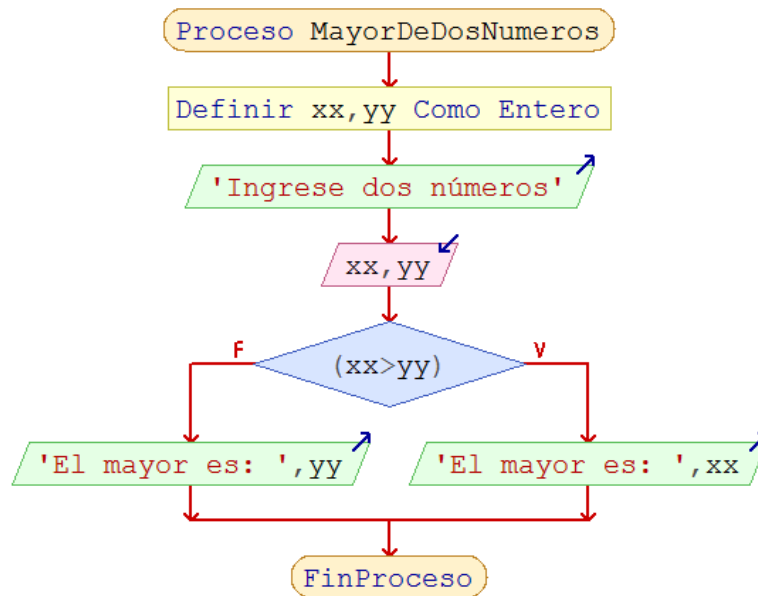


Fuente: Joyanes Aguilar, Fundamentos de Programación (2008)

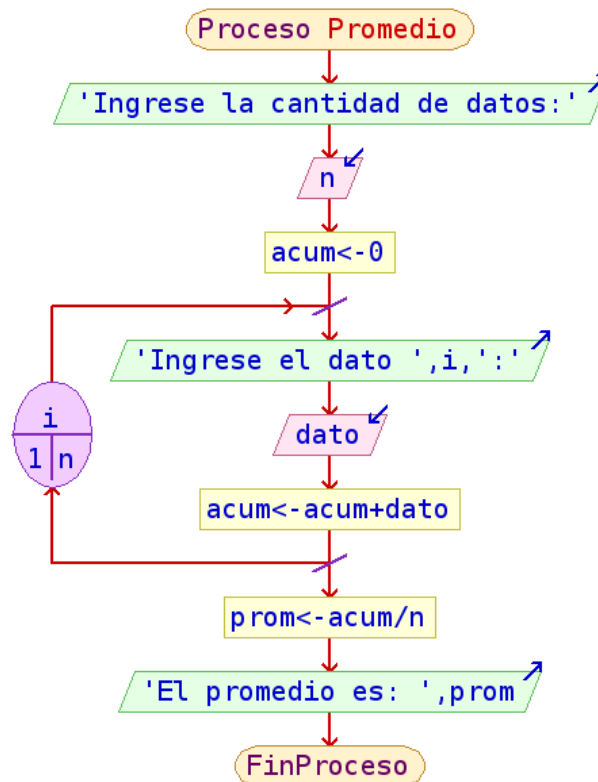
Algunos ejemplos de diagrama de flujo:



Fuente: <http://pseint.sourceforge.net/index.php?page=ejemplos.php&cual=Suma&mode=flow>



Fuente: [https://www.aprenderaprogramar.pro/2017/06/algoritmos-pseint\\_25.html](https://www.aprenderaprogramar.pro/2017/06/algoritmos-pseint_25.html)



Fuente: <http://pseint.sourceforge.net/index.php?page=ejemplos.php&cual=Promedio&mode=flow>

## 2.2 Diagrama Nassi/Schneiderman (N-S):

Representación en bloques, es decir cada uno de los procesos como ingreso/salida de datos, decisiones, acciones, repeticiones; están representados en bloques de tareas consecutivos.

INICIO
Declaración de variables
Lectura de Datos
Acción 1
Acción 2
Acción n
Reporte de Datos
FIN

Algunos ejemplos de diagrama N-S:

<b>Proceso Suma</b>
Escribir 'Ingrese el primer numero: '
Leer A
Escribir 'Ingrese el segundo numero: '
Leer B
$C \leftarrow A+B$
Escribir 'El resultado es: ',C
FinProceso

Fuente: <http://pseint.sourceforge.net/index.php?page=ejemplos.php&cual=Suma&mode=ns>

<b>Proceso Promedio</b>
Escribir 'Ingrese la cantidad de datos: '
Leer n
$acum \leftarrow 0$
Para i Desde 1 Hasta n Con Paso
Escribir 'Ingrese el dato ',i,': '
Leer dato
$acum \leftarrow acum + dato$
$prom \leftarrow acum/n$
Escribir 'El promedio es: ',prom
FinProceso

Fuente: [http://pseint.sourceforge.net/ejemplos/Promedio\\_ns.png](http://pseint.sourceforge.net/ejemplos/Promedio_ns.png)

<b>SubProceso resultado&lt;-Potencia (base,exponente)</b>
Definir resultado Como Entero
$exponente = 0$
Si
$resultado \leftarrow -1$
No
$resultado \leftarrow base * Potencia(base,exponente-1)$
FinSubProceso

<b>Proceso DosALaDiezRecursoivo</b>
Escribir 'Ingrese Base'
Leer base
Escribir 'Ingrese Exponente'
Leer exponente
Escribir 'El resultado es ',Potencia(base,exponente)
FinProceso

Fuente: <http://pseint.sourceforge.net/index.php?page=ejemplos.php&cual=Potencia&mode=ns>

## **Tema n.º 2: Variables y Tipos de Datos**

*Los algoritmos descritos anteriormente, ya sean computacionales o no, deben considerar valores, que pueden ser cambiantes según los datos que se obtengan, por lo que, dichos valores requieren de un espacio contenedor o de almacenamiento, denominado Variable.*

### **Sub tema 2.1 Definición de Variable**

“En los programas de computadoras existen valores que cambiarán durante la ejecución del programa”(Joyanes Luis, 2008)

En decir, una variable es un espacio de memoria en el computador, creado y usado durante la ejecución del programa.

#### **Representación de una variable**



Las variables suelen representar los datos que se usan en el algoritmo, y cuyos valores cambiarán según el proceso.

Se recomienda que el nombre de la variable se aproxime al nombre real del dato según el algoritmo, aún si el nombre es muy largo.

Por ejemplo:

- Dato en la realidad : **Cantidad de Horas Trabajadas**
- Creación de la variable : **Cant\_Hor\_Trabj**
- O También: : **CantHorTrabj**

Además, existen datos que no cambian durante la ejecución del programa, denominado Constante, y al igual que la variable, se recomienda asignarle un nombre aproximado a su nombre real.

Por ejemplo:

- Dato en la realidad : **Máximo de alumnos por asignatura**
- Creación de la variable : **Max\_Alumn**
- O También: : **MaxAlumn**

También, se debe considerar que existen variables globales y variables locales.

Las Variables Locales se crean y usan en un mismo módulo de programa, mientras que las Variables Globales se crean externamente y se usan en varios módulos de programa.

Los valores que pueden tomar una variable y constante dependen del tipo de información que almacena.

Por ejemplo:

- La Variable Cantidad de Horas Trabajadas: **CantHorTrabj = 5**
- O También : **CantHorTrabj = 3.5**

Existen otros valores según el tipo de variable:

Por ejemplo:

- La Variable Genero del Trabajador : **GenTrabaj = 'M'**
- O También : **GenTrabaj = "Masculino"**
- La Variable Respuesta : **Respuesta = True**
- O También : **Respuesta = False**

Para ello, se usan los denominados Tipos de Datos, para crear el tipo de variable.

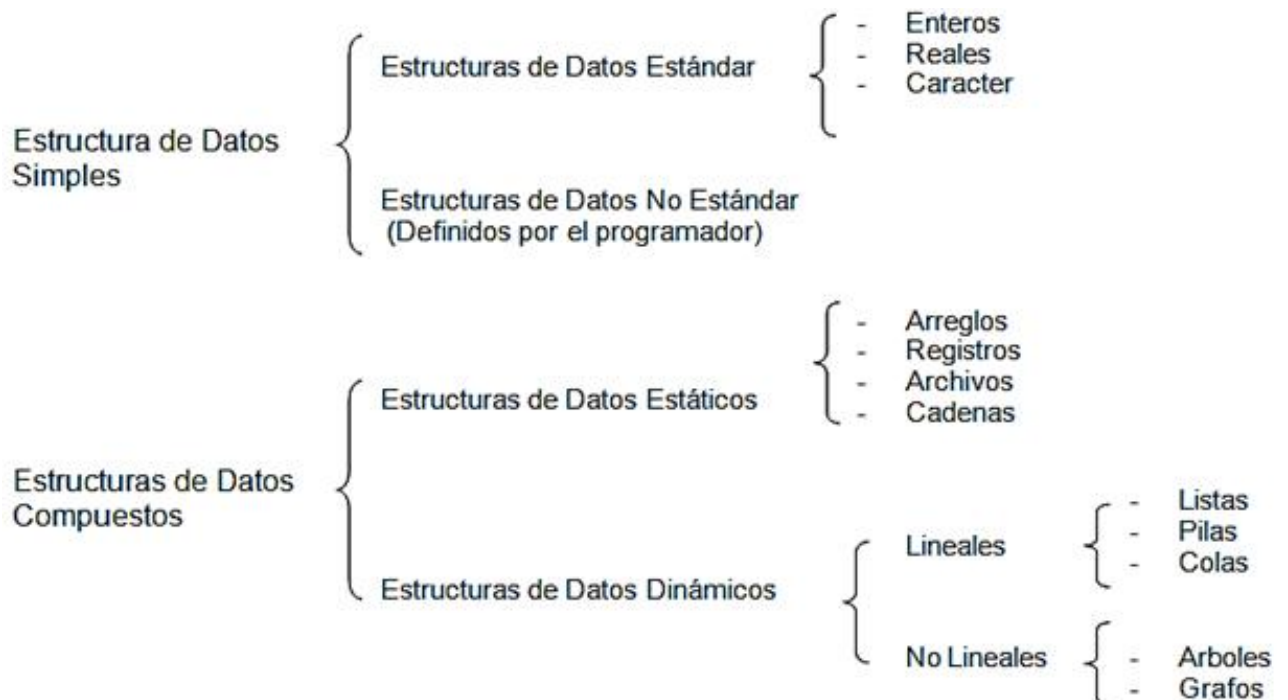
## Sub tema 2.2 Tipos de Datos

“Conjunto específico de valores de los datos y un conjunto de operaciones que actúan sobre estos” (Joyanes, Luis, 2008)

Es decir, los tipos de datos dan el formato de información que se quiere usar en la variable, como pueden ser numéricos (enteros o reales) o cadenas de caracteres (letras, números, símbolos)

Existen tipos de datos básicos (estándar), que se incluyen en el lenguaje de programación a usar, y otros que el programador puede crear.

Además existen tipos de datos simples, que permiten guardar un solo valor, y tipos de datos compuestos, que por su estructura permite guardar varios valores en un espacio de memoria.



Fuente: Joyanes Aguilar. Fundamentos de Programación (2008)



Para el lenguaje de programación C/C++ por ejemplo, se tienen los siguientes tipos de datos y sus características:

Tipo	Tam. Bits	Dígitos de precisión	Rango	
			Min	Max
Bool	8	0	0	1
Char	8	2	-128	127
Signed char	8	2	-128	127
unsigned char	8	2	0	255
short int	16	4	-32,768	32,767
unsigned short int	16	4	0	65,535
Int	32	9	-2,147,483,648	2,147,483,647
unsigned int	32	9	0	4,294,967,295
long int	32	9	-2,147,483,648	2,147,483,647
unsigned long int	32	9	0	4,294,967,295
long long int	64	18	-9,223,372,036,854,775,808	9,223,372,036,854,775,807
unsigned long long int	64	18	0	18,446,744,073,709,551,615
Float	32	6	1.17549e-38	3.40282e+38
Double	64	15	2.22507e-308	1.79769e+308

Fuente: <http://aprende-a-programar-ya.blogspot.com/2011/08/tipos-de-datos-en-c.html>

Como se muestra en la figura anterior, existen diversos tipos de datos que pueden ser asignados a las variables y constantes, según su uso y su sintaxis en el lenguaje de programación, por ejemplo en este caso el lenguaje C/C++:

Sintaxis para una Variable:

Tipo de Dato      Nombre de Variable;  
 Ejemplo:              **int**              **edad;**

Sintaxis para una Constante:

const Tipo de Dato    Nombre de Constante = Valor;  
 Ejemplo:              **const**      **int**              **MaximoDatos = 100;**

O también para la constante:

#define Nombre de Constante    Valor  
 Ejemplo:              **#define**      **MaximoDatos**      **100**

Por lo que, se puede concluir como ejemplos para diferentes tipos de datos:

Tipo de Dato	Nombre de Variable	Sintaxis en C/C++	Valores
entero	Cantidad_Productos	int Cantidad_Productos;	Cantidad_Productos = 42;
real (double)	Precio_Producto	double Precio_Producto;	Precio_Producto = 10.5;
caracter	Genero_Trabajador	char Genero_Trabajador;	Genero_Trabajador = ' M ';
cadena	Nombre_Trabajador	string Nombre_Trabajador;	Nombre_Trabajador = " Juan ";

Fuente: Propia

Además, según las instrucciones algorítmicas básicas presentadas en el tema anterior, se puede usar la sintaxis del lenguaje de programación del lenguaje, por ejemplo del lenguaje C/C++:

#### a. Entrada

**INGRESAR** variable

```
// Ingresar Valores
cin>>Variable1;
cin>>Variable2;
```

#### b. Asignación

variable ← expresión

```
// Procesar
Total = Variable 1 + Variable2;
```

#### c. Salida

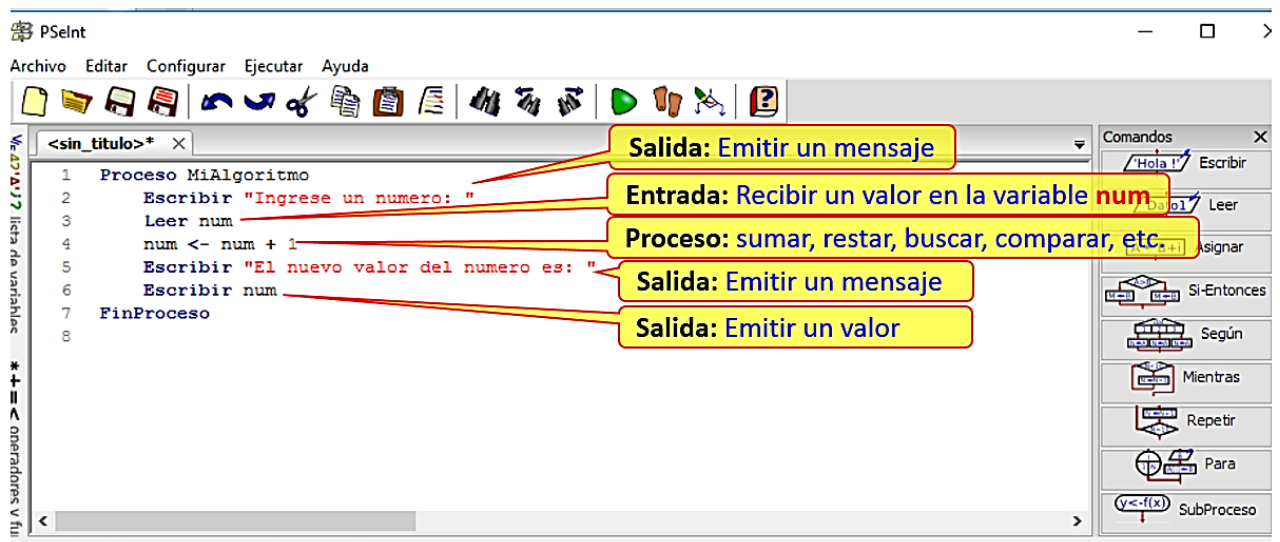
**MOSTRAR** variable

```
// Mostrar Mensajes y Valores
cout<<"El resultado es: ";
cout<<Total;
```

Fuente: Propia

## De la teoría a la práctica

Usando la herramienta del PSeInt:



Fuente: Propia

Se puede visualizar los diagramas: de flujo y N-S:

Diagrama de Flujo

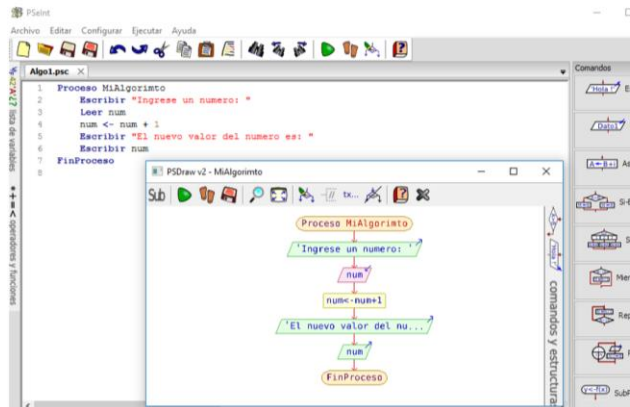
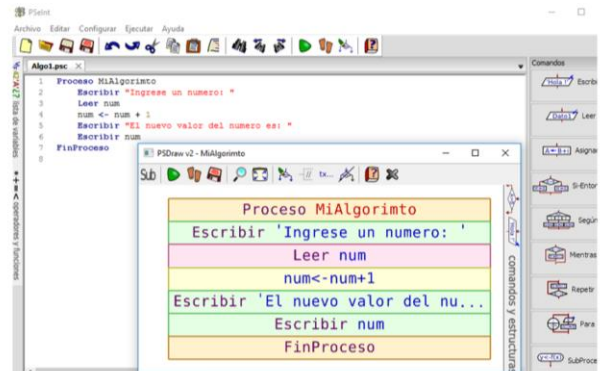
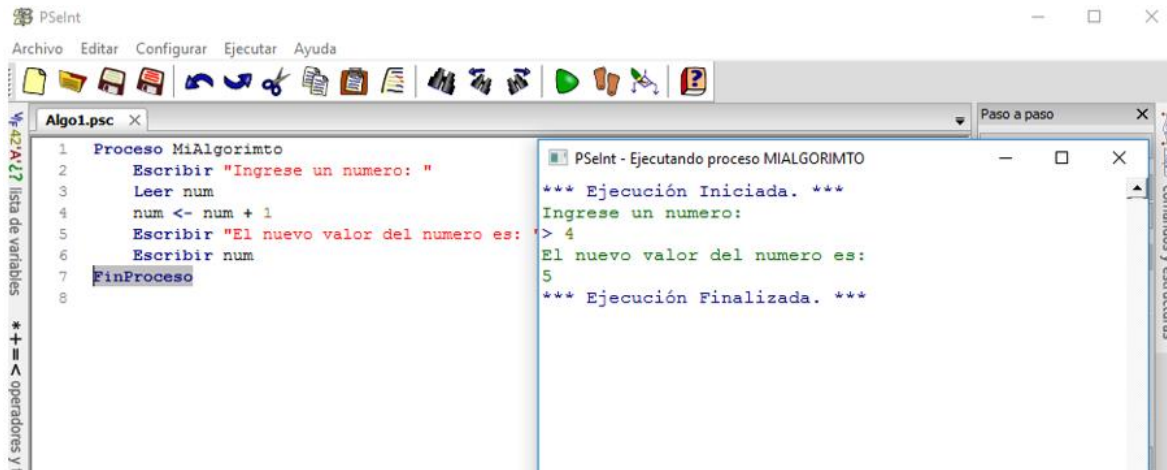


Diagrama de N-S



Fuente: Propia

Y la ejecución del respectivo algoritmo:



Fuente: Propia

Como se puede observar, esta herramienta permite usar las variables, tanto para el ingreso, el proceso y al salida de datos.

Además, permite elaborar el pseudocódigo y como consecuencia, obtener el diagrama de flujo y el diagrama N-S, con su respectiva ejecución.

### **Tema n.º 3: Introducción a la Programación: Programas Traductores, Programación Estructurada.**

*Los algoritmos computacionales o no, pueden ser escritos en un lenguaje para que pueda ser ejecutado en una computadora. Este lenguaje permitirá el reconocimiento de lo escrito por parte del procesador hacia lenguaje máquina.*

#### **Sub tema 3.1 Definición de Programa**

Se entiende por programa, como una secuencia de instrucciones que especifican las operaciones que debe realizar la computadora, en la cual cada paso del algoritmo está expresado por medio de una instrucción.

Es decir, el algoritmo en sí, ya puede considerarse un programa, si está escrito en un lenguaje de programación.

También, un programa puede contener más de un algoritmo.

Ejemplo de Programa Lenguaje en C/C++:

```
#include <iostream>

using namespace std;

int main() {
    cout << "Hola mundo!" << endl;
    return 0;
}
```

Fuente: <https://www.monografias.com/trabajos104/estructura-basica-programa-c-c/estructura-basica-programa-c-c.shtml>

Ejemplo de Programa Lenguaje en Java++:

```
1
2 public class HolaMundo {
3
4     public static void main(String[] args) {
5         // TODO Auto-generated method stub
6         System.out.print("Hola Mundo");
7     }
8
9
10 }
```

Fuente: <https://programacionparatodosite.wordpress.com/1-2-hola-mundo-en-java/>

### Sub tema 3.2 Lenguaje de Programación

Los programas son escritos en lenguajes de programación, es decir en un conjunto de símbolos que permiten identificar la sintaxis y semántica del algoritmo.

Si tomamos como ejemplo el lenguaje natural con el que nos expresamos:

- Lenguaje Español : **HOLA**
- Lenguaje Inglés : **HELLO**
- Lenguaje Alemán : **HALLO**

Observamos que la palabra HOLA se puede escribir en diferentes lenguajes con sus propios símbolos (sintaxis) pero tiene el mismo significado (semántica)

Por lo que, se puede decir que un lenguaje de programación es un conjunto de sentencias utilizadas para escribir secuencias de instrucciones que para que ejecute un programa en una computadora.

Los lenguajes de Programación pueden ser:

- Lenguaje Máquina:** Lenguaje de programación que la computadora interpreta y ejecuta directamente, y está compuesto de instrucciones codificadas en binario (0, 1).
- Lenguaje de Bajo Nivel:** también llamados lenguajes ensambladores, permiten al programador escribir instrucciones de un programa usando abreviaturas del lenguaje natural (inglés), también llamadas palabras nemotécnicas (ADD, DIV, SUB, etc).
- Lenguaje de Alto Nivel:** permite al programador escribir las instrucciones de un programa utilizando palabras o expresiones sintácticas muy similares al lenguaje natural (inglés).

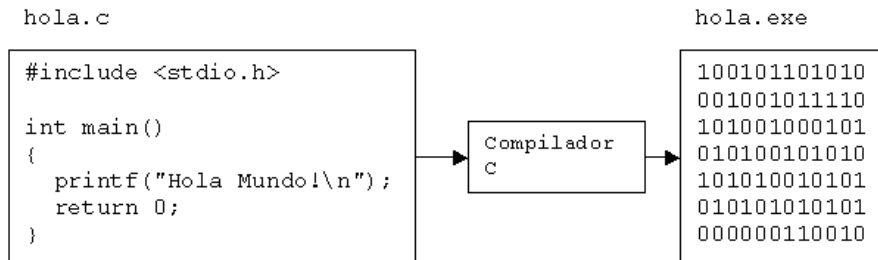


Fuente: <https://conociendoalgoritmo.weebly.com/lenguaje-de-alto-nivel.html>

### Sub tema 3.3 Programa Traductor

Este tipo de programa traduce los programas fuente de a código máquina, es decir para que el computador pueda entender y ejecutar las instrucciones dadas en el programa.

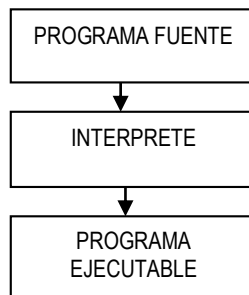
#### Ejemplo de Lenguaje máquina



Fuente: [http://go.yuri.at/juegos/clase1/clase1\\_introduccion.html](http://go.yuri.at/juegos/clase1/clase1_introduccion.html)

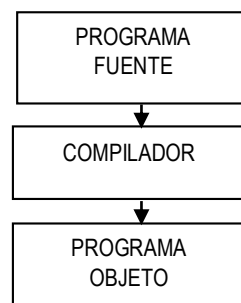
Pueden ser:

- a. Intérprete:** es un traductor que toma un programa fuente, lo traduce y a continuación lo ejecuta.



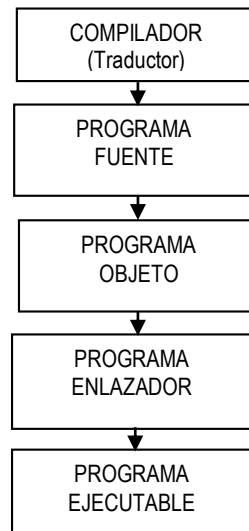
Fuente: Joyanes Aguilar. Fundamentos de Programación (2008)

- b. Compilador:** es un traductor que toma un programa fuente, y lo traduce sentencia por sentencia.



Fuente: Joyanes Aguilar. Fundamentos de Programación (2008)

La compilación traduce el programa fuente a programa objeto (código máquina), y con un programa enlazador, conduce al programa ejecutable.



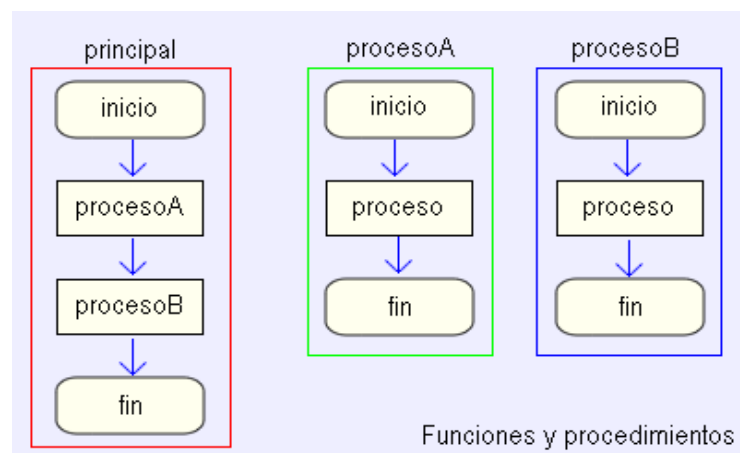
Fuente: Joyanes Aguilar. Fundamentos de Programación (2008)

### Sub tema 3.4 Programación Estructurada

Para construir (analizar, diseñar, implementar) un software, se puede realizar considerando los conceptos y técnicas de uno de dos enfoques: estructurado y orientado a objetos.

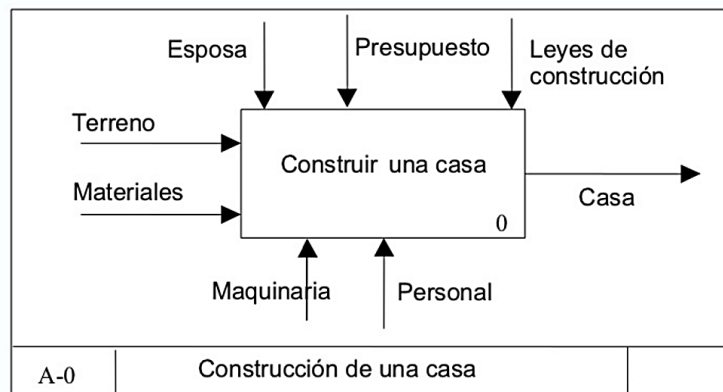
Para este caso, vamos a considerar la programación bajo el enfoque estructurado, y en asignaturas posteriores la programación orientada a objetos. Por lo que, la programación estructurada es un conjunto de técnicas para escribir, verificar, depurar, y mantener los programas; el cual plantea una operación como un todo y se divide en segmentos más sencillos o de menor complejidad. Una vez terminado todos los segmentos del programa, se procede a unificar las aplicaciones, esta integración debe ser sencilla.

#### Ejemplo de Enfoque estructurado



Fuente: <http://programacioncittlajessiyaz.blogspot.pe/p/unidad-5-funciones.html>

### Ejemplo de Enfoque estructurado con IDEF



Fuente: <http://es.slideshare.net/JuanPablo157/diagramas-idef-0-y-3>

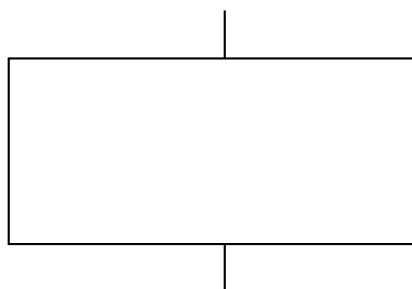
Este conjunto de técnicas son:

#### a. Recursos Abstractos:

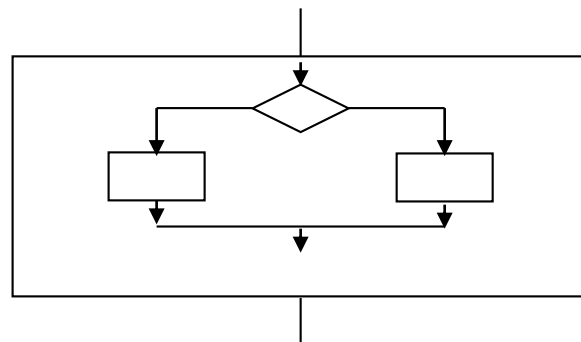
Identificar de la realidad una acción compleja (problema) y especificarla en acciones más simples, para ser ejecutadas.

#### b. Diseño Descendente:

El problema se descompone en niveles o pasos sucesivos: que hace? como lo hace?



Que hace?



Como hace?

#### c. Estructuras de Control (Sentencias de Programación):

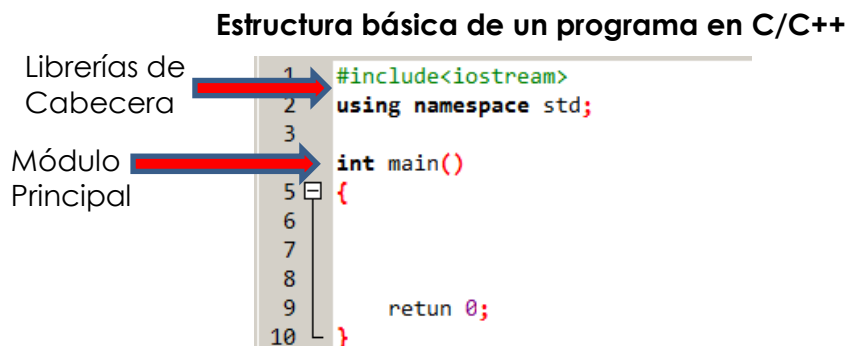
Son instrucciones que permiten escribir la solución del problema (algoritmo), y se clasifican en:

- ✓ Secuenciales
- ✓ Selectivas
- ✓ Repetitivas



## Estructura de un programa en el Lenguaje C/C++

Para iniciar la elaboración de programas en esta asignatura, se muestra la Estructura de un programa simple en C++, en el tema de Módulos de programa, la estructura del programa tendrá algunas modificaciones, es decir se crearán más módulos que serán invocados por el módulo principal.



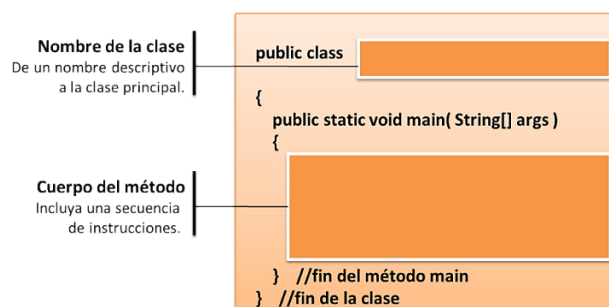
Fuente: elaboración propia

Descripción:

- El símbolo # es una directiva que permite acceder la carpeta include.
- La librería include es propia del compilador, donde se encuentran las librerías de cabecera (extensión .h) del lenguaje C/C++
- Una de las librerías de cabecera es iostream, (i: input, o: output, stream: flujo de cadenas) que permite reconocer las instrucciones de entrada, salida del programa.
- using namespace std; uso del espacio de conjunto de símbolos estándar.
- El módulo principal main( ), en el cual se escribirán las instrucciones de programa y las invocaciones a otros módulos de programa.

En otros lenguajes de programación como, por ejemplo, en java, se presenta la siguiente estructura.

## Estructura básica de un programa en Java



Fuente: <http://profejavoramas.blogspot.pe/2010/04/estructura-de-un-programa-en-java.html>

## Tema n.º 4: Estructuras de control para la programación: Estructuras de control Secuencial.

*Iniciaremos la construcción de programas en un lenguaje de programación, considerando que se ha identificado problemas de la realidad (enunciados de ejercicios y problemas) y se diseñaran propuestas de solución (algoritmo) para luego ser escritos y ejecutados en lenguaje de programación (programa).*

### Sub tema 4.1 Definición de Estructuras de Control Secuencial

Una Estructura de Control o Sentencia de Programación Secuencial, a la acción (instrucción) que se ejecuta paso a paso, sin bifurcaciones ni repeticiones; donde la salida de un proceso es entrada para otro.

Se puede representar con el Diagrama de Flujo o el Diagrama N-S.

Diagrama de Flujo:

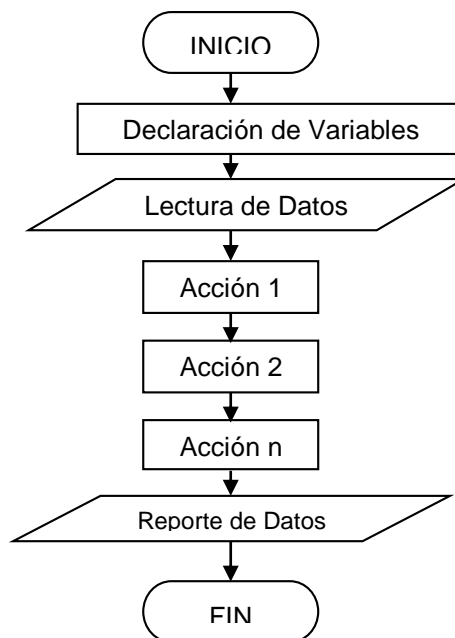


Diagrama N-S:

INICIO
Declaración de variables
Lectura de Datos
Acción 1
Acción 2
Acción n
Reporte de Datos
FIN

## De la teoría a la práctica

**Ejemplo 1:** para el caso de calcular el área de un triángulo.

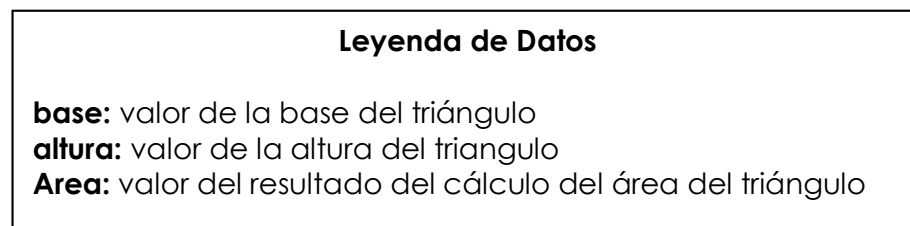
$$\text{Area} = (\text{base} * \text{altura}) / 2$$

### Solución:

Recuerde crear las variables para el ingreso, proceso y salida de los datos.

Además, se sugiere que el nombre de las variables a crear, sean similares a sus nombres en el contexto real.

En caso de que el problema pueda ser muy complejo, o que los datos no sean muy conocidos, se puede establecer una leyenda de datos con una breve descripción de las variables.



Se puede realizar uno de los diagramas para representar al algoritmo propuesto.

Diagrama de Flujo:

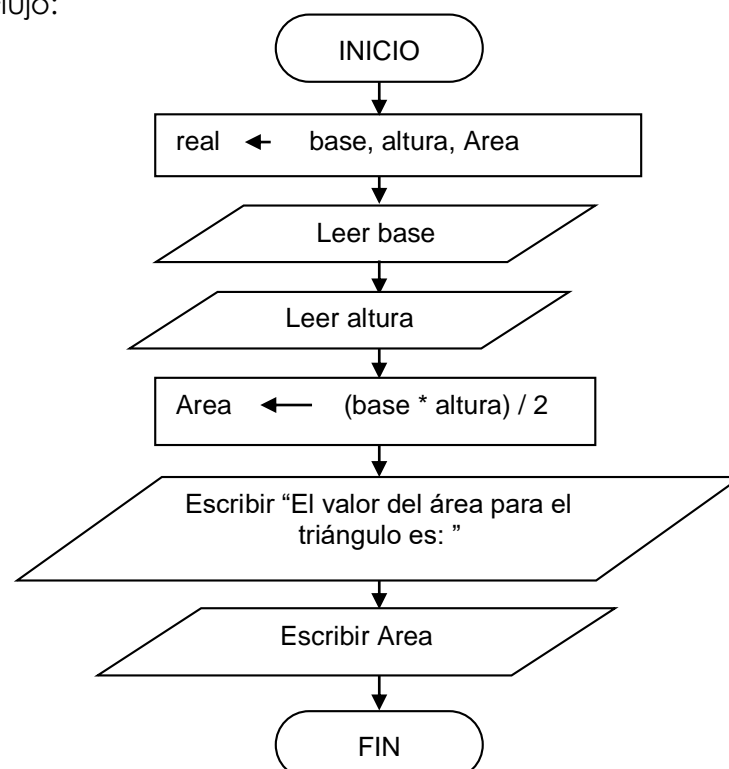


Diagrama N-S:

INICIO
real $\leftarrow$ base, altura, Area
Leer base
Leer altura
Area $\leftarrow$ (base * altura) / 2;
Escribir "El valor del área para el triángulo es: "
Escribir Area
FIN

Después de haber planteado la solución (algoritmo) se puede elaborar el programa en un lenguaje de programación.

Código en C/C++:

```

1  #include<iostream>
2  using namespace std;
3
4  void main( )
5  {
6      float base, altura, Area;
7
8      cout<<"Ingrese valor de la base: ";
9      cin>>base;
10     cout<<"\n";
11
12     cout<<"Ingrese valor de la altura: ";
13     cin>>altura;
14     cout<<"\n";
15
16     Area = (base * altura) / 2;
17
18     cout<<"El valor del área para el triángulo es: ";
19     cout<<Area;
20     cout<<"\n";
21 }

```

Fuente: elaboración propia

**Ejemplo 2:** Tenemos los datos de presión, el volumen y la temperatura de una masa de aire se relacionan por la fórmula:

$$\text{Masa} = (\text{presión} * \text{volumen}) / (0.37 * (\text{temperatura} + 460))$$

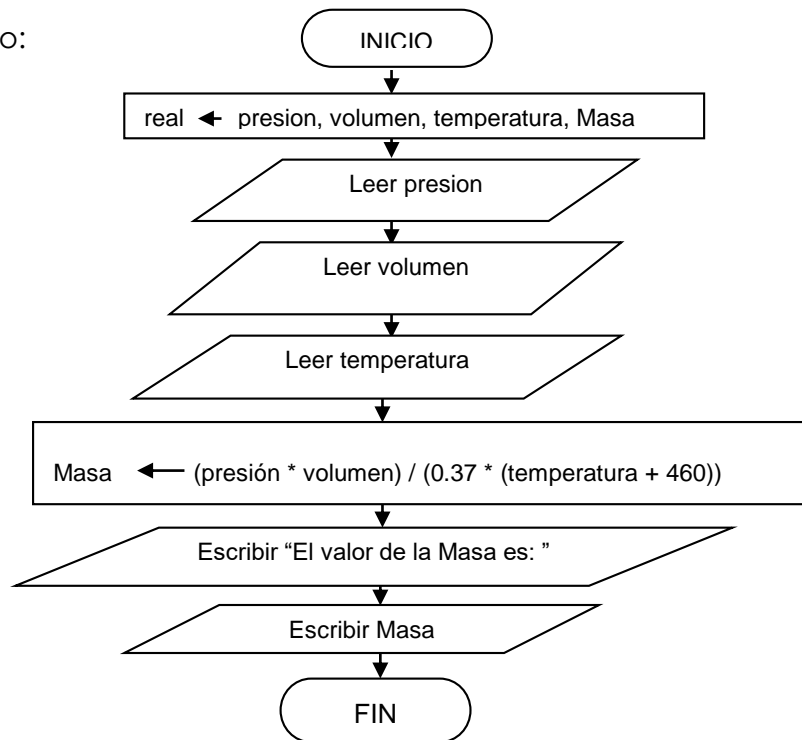
### Solución:

De igual forma que el ejemplo anterior, se sugiere definir las variables asignándoles un nombre que se aproxime a su significado según el contexto que se trabaje:

Definición de variables:

<b>presion</b>	: valor de la presión a ingresar
<b>volumen</b>	: valor del volumen a ingresar
<b>temperatura</b>	: valor de la temperatura ingresar
<b>Masa</b>	: valor del resultado del cálculo

Diagrama de Flujo:



Código C/C++:

```

1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      double presion, volumen, temperatura, Masa;
7
8      //ENTRADA DE DATOS
9      cout<<"Ingrese presion: ";
10     cin>>presion;
11
12     cout<<"Ingrese volumen: ";
13     cin>>volumen;
14
15     cout<<"Ingrese temperatura: ";
16     cin>>temperatura;
17
18     //PROCESO: cálculo
19     Masa = (presion * volumen) / (0.37 * (temperatura + 460));
20
21     //SALIDA DE DATOS
22     cout<<"El valor de Masa es: ";
23     cout<<Masa;
24
25     return 0;
26 }

```

Fuente: elaboración propia

**Ejemplo 3:** Ingresar el sueldo de tres empleados y aplicarles un incremento de 10%, 20%, 30% respectivamente. Reportar los nuevos valores de sueldo:

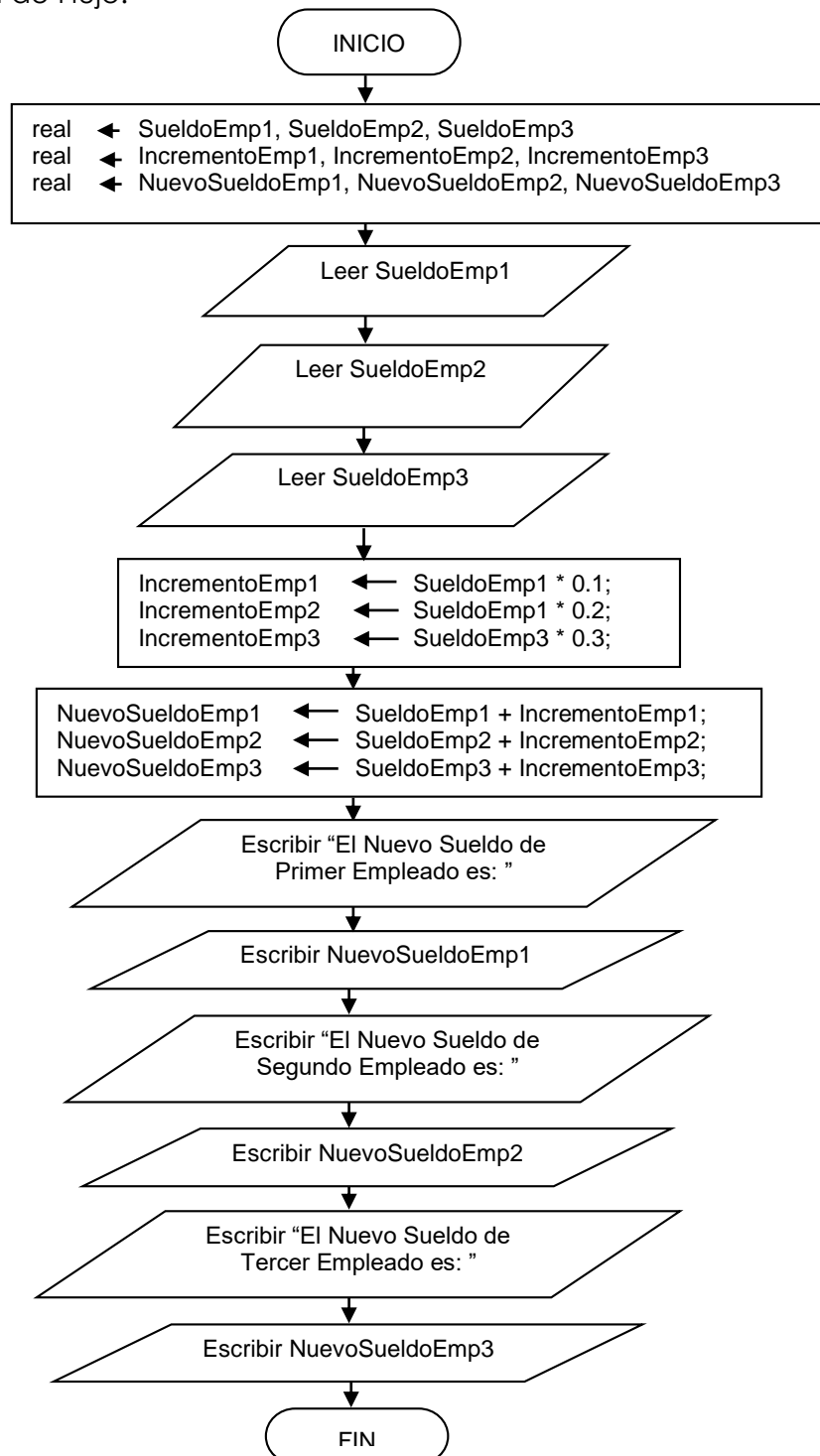
**Solución:**

También se debe definir variables, pero en esta oportunidad, se propondrá un cuadro que le permita definir las tres partes del algoritmo de solución:

Identificar Variables a usar	Variable(s) a ingresar	Proceso (cálculo)	Variable(s) para reportar
SueldoEmp1 SueldoEmp2 SueldoEmp3 IncrementoEmp1 IncrementoEmp2 IncrementoEmp3 NuevoSueldoEmp1 NuevoSueldoEmp2 NuevoSueldoEmp3	SueldoEmp1  SueldoEmp2  SueldoEmp3	$\text{IncrementoEmp1} = \text{SueldoEmp1} * 0.1;$ $\text{IncrementoEmp2} = \text{SueldoEmp1} * 0.2;$ $\text{IncrementoEmp3} = \text{SueldoEmp3} * 0.3;$  $\text{NuevoSueldoEmp1} = \text{SueldoEmp1} + \text{IncrementoEmp1};$ $\text{NuevoSueldoEmp2} = \text{SueldoEmp2} + \text{IncrementoEmp2};$ $\text{NuevoSueldoEmp3} = \text{SueldoEmp3} + \text{IncrementoEmp3};$	NuevoSueldoEmp1  NuevoSueldoEmp2  NuevoSueldoEmp3

Fuente: elaboración propia

Diagrama de Flujo:



Código C/C++:

```
1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      double SueldoEmp1, SueldoEmp2, SueldoEmp3;
7      double IncrementoEmp1, IncrementoEmp2, IncrementoEmp3;
8      double NuevoSueldoEmp1, NuevoSueldoEmp2, NuevoSueldoEmp3;
9
10     //ENTRADA DE DATOS
11
12     cout<<"Ingrese Sueldo de Primer Empleado: ";
13     cin>>SueldoEmp1;
14
15     cout<<"Ingrese Sueldo de Segundo Empleado: ";
16     cin>>SueldoEmp2;
17
18     cout<<"Ingrese Sueldo de Tercer Empleado: ";
19     cin>>SueldoEmp3;
20
21     //PROCESO:
22     //a. cálculo de incremento según porcentaje
23
24     IncrementoEmp1 = SueldoEmp1 * 0.1;
25     IncrementoEmp2 = SueldoEmp2 * 0.2;
26     IncrementoEmp3 = SueldoEmp3 * 0.3;
27
28     //b. cálculo de nuevo sueldo según incremento
29
30     NuevoSueldoEmp1 = SueldoEmp1 + IncrementoEmp1;
31     NuevoSueldoEmp2 = SueldoEmp2 + IncrementoEmp2;
32     NuevoSueldoEmp3 = SueldoEmp3 + IncrementoEmp3;
33
34     //SALIDA DE DATOS
35     cout<<endl;
36     cout<<"El Nuevo Sueldo de Primer Empleado es: ";
37     cout<<NuevoSueldoEmp1<<endl;
38
39     cout<<"El Nuevo Sueldo de Segundo Empleado es: ";
40     cout<<NuevoSueldoEmp2<<endl;
41
42     cout<<"El Nuevo Sueldo de Tercer Empleado es: ";
43     cout<<NuevoSueldoEmp3<<endl;
44
45     return 0;
46 }
```

Fuente: elaboración propia

## **Tema n.º 5: Estructuras de control para la programación: Estructuras de control Selectiva Simple.**

*Como se habrá dado cuenta, no siempre se tendrá casos o procesos en que la solución sea desarrollada de forma secuencial, ya que es necesario condicionar o colocar restricciones para las tareas, por esta razón que en esta sección se presentan las estructuras de control selectivas simple y compuesta.*

### **Sub tema 5.1 Definición de Estructuras de Control Selectiva**

Una Estructura de Control o Sentencia de Programación Selectiva, proporciona una condición o punto de decisión para realizar un conjunto u otro conjunto de acciones (otras sentencias, que pueden ser secuencial, selectivas o repetitivas). Sentencia de Programación Selectiva puede ser de tres tipos:

- Sentencia Selectiva Simple
- Sentencia Selectiva Compuesta
- Sentencia Selectiva Múltiple

Cada uno de estos tres tipos de sentencias se puede usar de manera conjunta, es decir colaborando entre sí, para escribir la propuesta de solución o algoritmo, formando las sentencias anidadas (una sentencia selectiva invoca a otra sentencia selectiva)

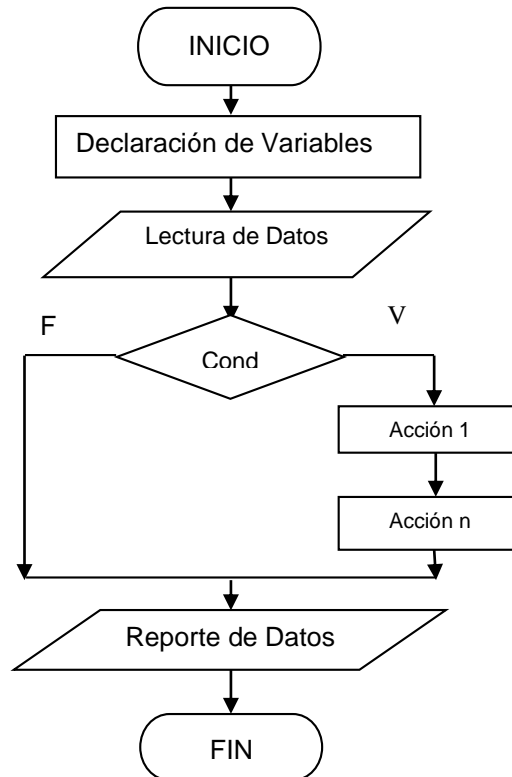
También se pueden combinar usando las sentencias repetitivas que se revisarán más adelante, permitiendo que el programa se complemente con mayor robustez para la solución del problema y con mayor flexibilidad para realizar cambios, que puedan ser solicitados por el cliente o usuario.



## Sub tema 5.2 Definición de Estructuras de Control Selectiva Simple

La Sentencia de Programación Selectiva Simple, proporciona una condición o punto de decisión y si esta es verdadera, se realiza un conjunto de acciones.

### Diagrama de Flujo para una Estructura de Control Selectiva Simple



Fuente: elaboración propia

## De la teoría a la práctica

**Ejemplo 1:** Sumar dos números enteros positivos.

$$c = a + b$$

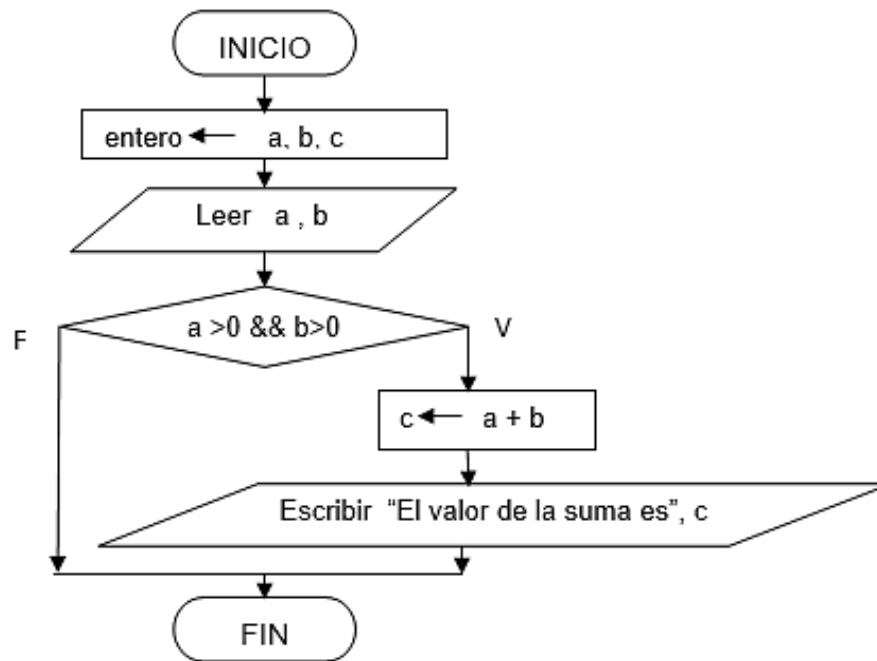
### Solución:

Se sugiere definir las variables para el proceso de entrada, asignándoles un nombre en este caso, pueden ser cualquier número, por eso se puede designar las variables:

Definición de variables:

- a:** valor del primer número entero positivo
- b:** valor del segundo número entero positivo
- c:** valor del resultado la suma de dos números a y b

Diagrama de Flujo:



Observe en el diagrama anterior, que la condición expresada en el rombo, evalúa que ambas variables sean mayores que cero (usa el operador && para la conjunción, en otros casos puede usar el operador || para la disyunción es decir o cumpla una u otra condición).

Código C/C++:

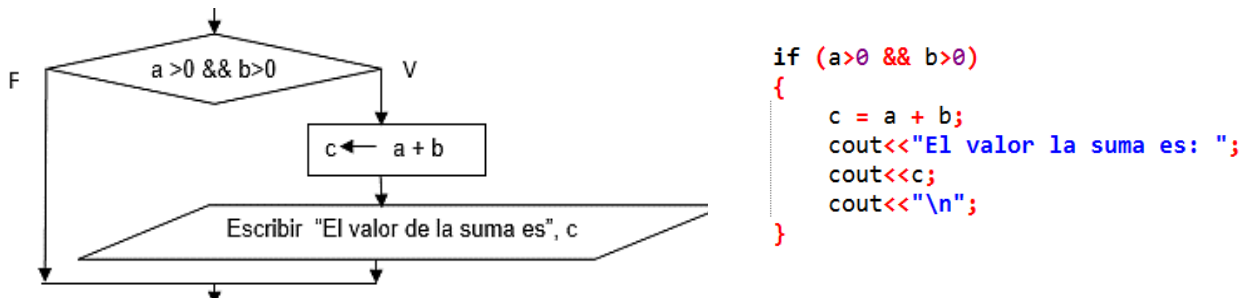
```

1  #include<iostream>
2  using namespace std;
3
4  int main( )
5  {
6      float a, b, c;
7
8      cout<<"Ingrese primer numero: ";
9      cin>>a;
10     cout<<"\n";
11
12     cout<<"Ingrese segundo numero: ";
13     cin>>b;
14     cout<<"\n";
15
16     if (a>0 && b>0)
17     {
18         c = a + b;
19         cout<<"El valor la suma es: ";
20         cout<<c;
21         cout<<"\n";
22     }
23
24     return 0;
25 }
  
```

Fuente: elaboración propia

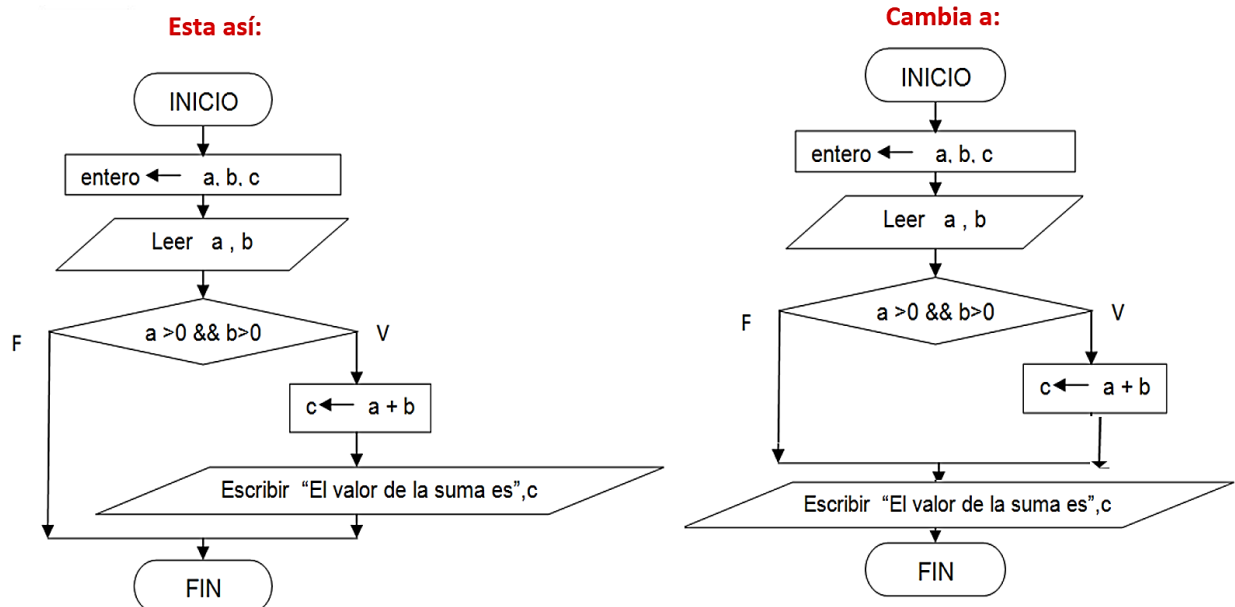
Considerando el código C/C++ mostrado en la figura anterior, podemos comparar la representación en diagrama de flujo y su expresión en código.

### Diagrama de Flujo y Código C/C++ de una Estructura de Control Selectiva Simple



Fuente: elaboración propia

En este punto, debe realizarse la pregunta, que pasaría si la sentencia de mensaje donde se muestra el valor de la suma, estuviera representada fuera de la sentencia selectiva, es decir qué valor tendría la variable **c**, creado para almacenar la suma?



Como podrá observar, si la condición  $a > 0 \ \&\& \ b > 0$  no se cumple, es decir es falso (F), la variable **c** no tendría el valor calculado de la suma, por lo que se mostraría vacía.

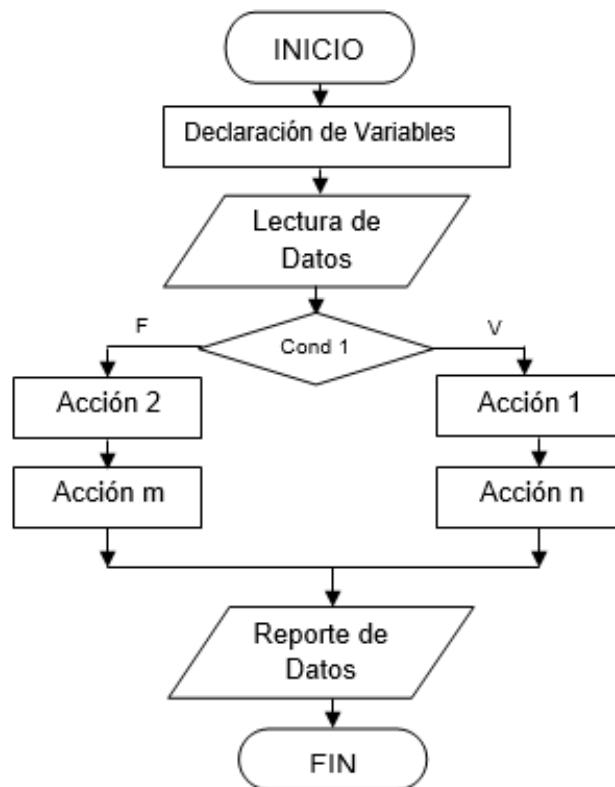
## Tema n.º 6: Estructuras de control para la programación: Estructuras de control Selectiva Compuesta.

Al revisar el diagrama y código C/C++ de la Sentencia Selectiva Simple, al obtener una condición Falsa, se hace necesario considerar otro grupo de acciones (secuenciales, selectivas y/o repetitivas) que permiten mejorar el flujo del programa.

### Sub tema 6.1 Definición de Estructuras de Control Selectiva Compuesta

La Sentencia de Programación Selectiva Compuesta, proporciona una condición o punto de decisión y si esta es verdadera, se realiza un conjunto de acciones y si es falsa (no cumple la condición), realizará otro conjunto de acciones.

#### Diagrama de Flujo para una Estructura de Control Selectiva Compuesta



Fuente: elaboración propia

## De la teoría a la práctica

**Ejemplo 1:** Sumar dos números enteros si son positivos, sino multiplicarlos.

$c = a + b$  , en caso sean positivos

$c = a * b$  , en caso no sean positivos

### Solución:

Se sugiere definir las variables para el proceso de entrada, asignándoles un nombre en este caso, pueden ser cualquier número, por eso se puede designar las variables:

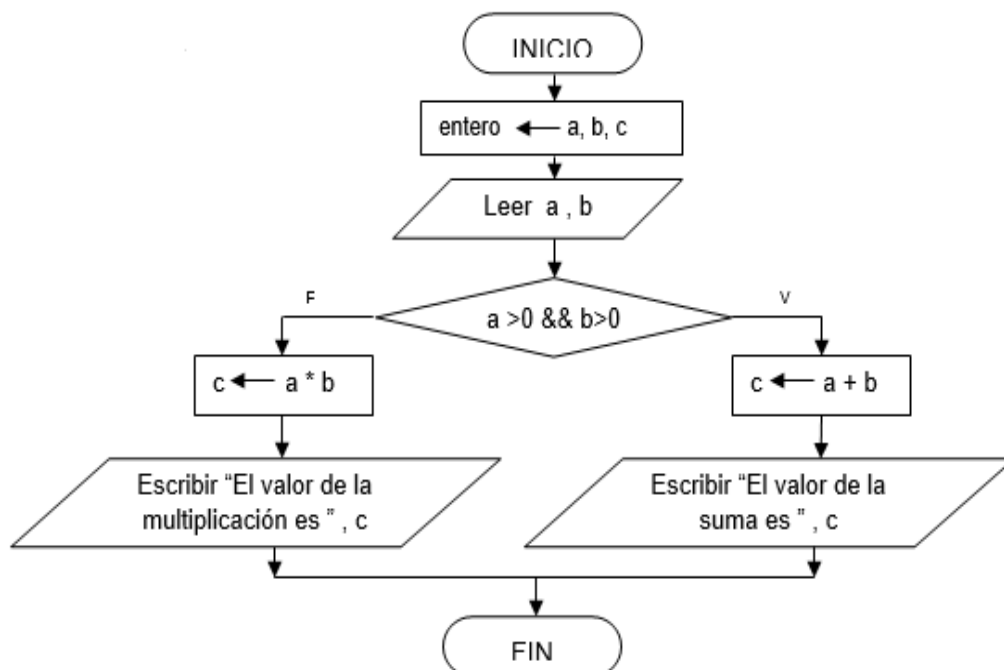
Definición de variables:

**a:** valor del primer número entero positivo

**b:** valor del segundo número entero positivo

**c:** valor del resultado la suma de dos números a y b

Diagrama de Flujo:



Observe en el diagrama anterior, que la condición expresada en el rombo, evalúa que ambas variables sean mayores que cero para sumar.

Pero si al menos una de las variables no cumple con la condición, se multiplicarán, y cada cálculo es asignado al a variable **c**.

Como el operador usado en la condición es de conjunción, nunca se realizarán los dos cálculos al mismo tiempo, por lo que ya no es necesario utilizar una variable distinta por cada cálculo de suma o de multiplicación.

Es importante considerar el adecuado uso de las variables en el programa, para que de esta manera podamos alinearlos a una de características de software que es la eficiencia en el uso de recursos de memoria del computador, además de la fácil corrección y mantenimiento del programa

Código C/C++:

```

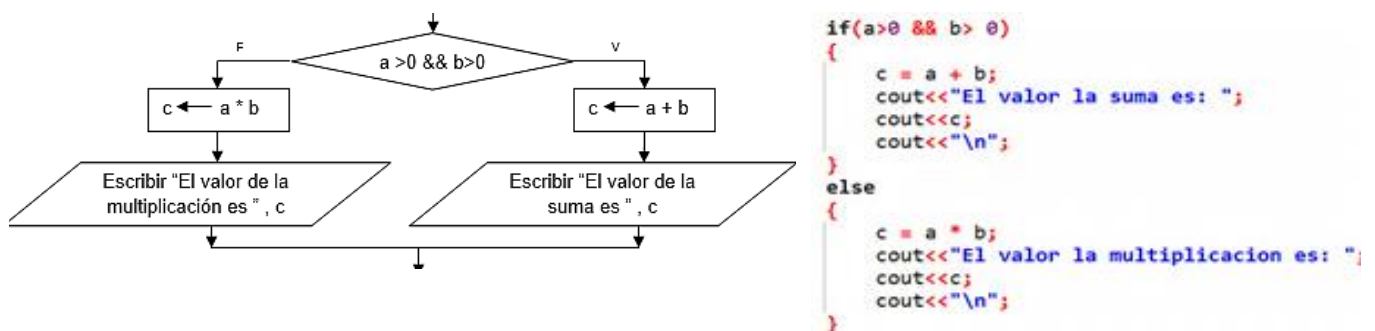
1  #include<iostream>
2  using namespace std;
3
4  void main( )
5  {
6      int a, b, c;
7
8      cout<<"Ingrese primer número: ";
9      cin>>a;
10     cout<<"\n";
11
12     cout<<"Ingrese segundo número: ";
13     cin>>b;
14     cout<<"\n";
15
16     if(a>0 && b> 0)
17     {
18         c = a + b;
19         cout<<"El valor la suma es: ";
20         cout<<c;
21         cout<<"\n";
22     }
23     else
24     {
25         c = a * b;
26         cout<<"El valor la multiplicacion es: ";
27         cout<<c;
28         cout<<"\n";
29     }
30 }

```

Fuente: elaboración propia

Considerando el código C/C++ mostrado en la figura anterior, podemos comparar la representación en diagrama de flujo y su expresión en código.

### Diagrama de Flujo y Código C/C++ de una Estructura de Control Selectiva Compuesta

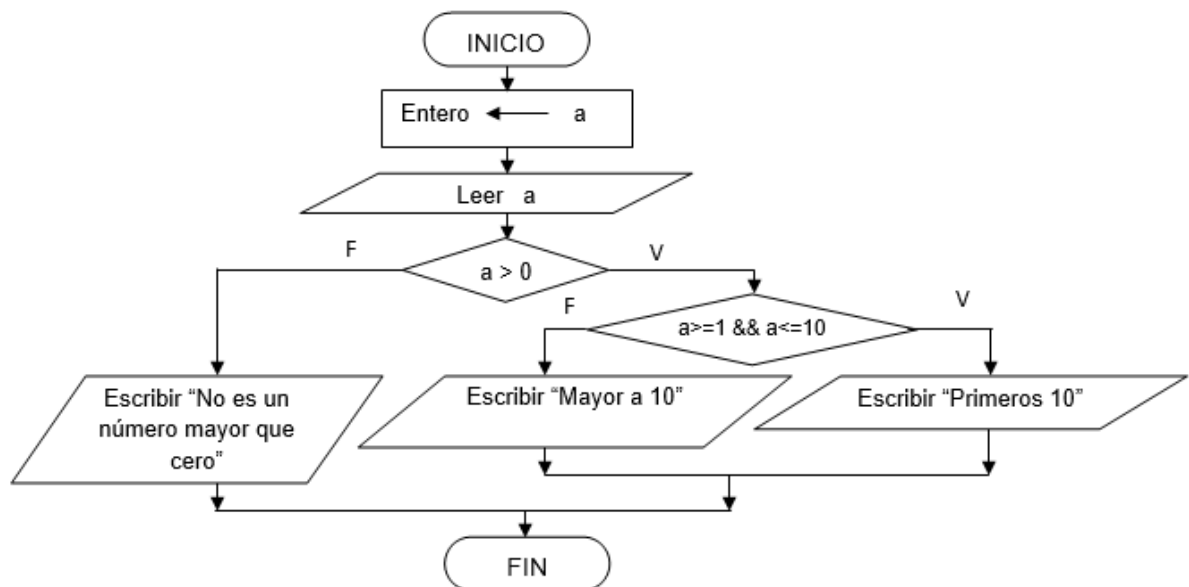


Fuente: elaboración propia

Ambas estructuras de control selectivas y compuestas, pueden combinarse en la solución de un programa, dando como resultado el anidamiento de estas estructuras.

**Ejemplo 2:** ingresar un número entero y si es positivo y diferente de cero, reportar si se encuentra entre los diez primeros números.

### Diagrama de Flujo para Ejemplo de Estructura de Control Anidada



Fuente: elaboración propia

Considerando el código C/C++ mostrado en la figura anterior, podemos comparar la representación en diagrama de flujo y su expresión en código.

### Solución en Código C/C++ para Ejemplo de Estructura de Control Anidada

```

1  #include<iostream>
2  using namespace std;
3
4  void main( )
5  {
6      int a;
7
8      cout<<"Ingrese número: ";
9      cin>>a;
10
11     if(a>0)
12         if(a>=1 && a<10)
13             cout<<"Primeros 10";
14         else
15             cout<<"Son más de los primeros 10";
16     else
17         cout<<"No es un número mayor que cero";
18 }
  
```

Fuente: elaboración propia

## **Tema n.º 7: Estructuras de control para la programación: Estructuras de control Selectiva Múltiple.**

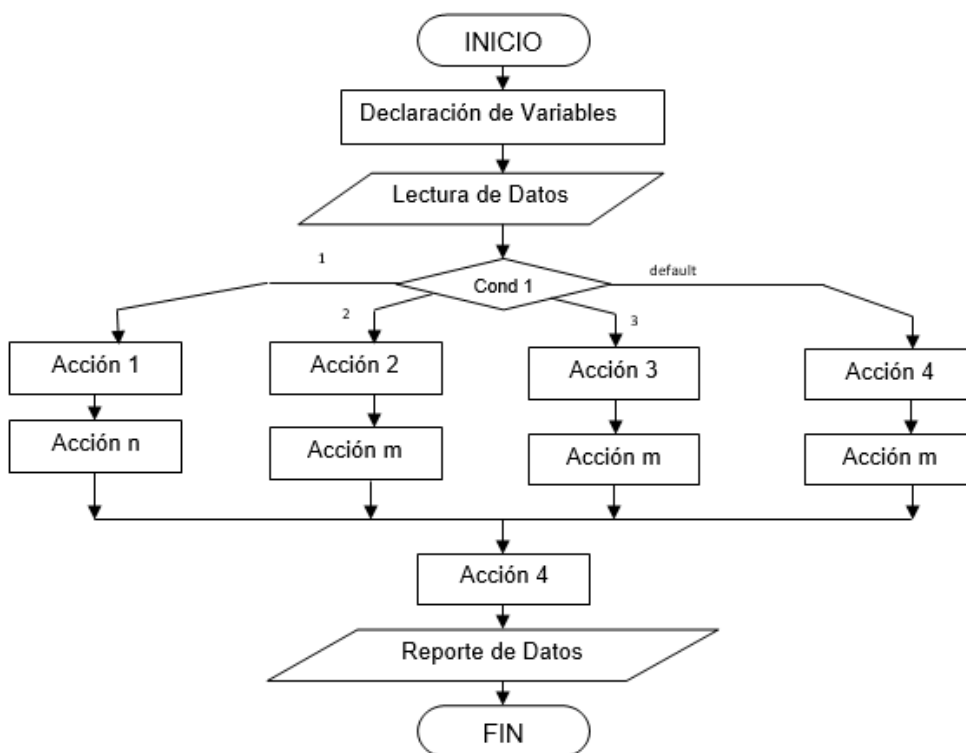
*Estimado estudiante, hasta este punto se ha evaluado condiciones que permiten resultados en verdadero o falso, pero no necesariamente siempre se tendrá es-tos casos.*

*A continuación se presenta una estructura de control de programación, que le permite evaluar un caso, de un conjunto de opciones.*

### **Sub tema 7.1 Definición de Estructuras de Control Selectiva Múltiple**

La Sentencia de Programación Selectiva Múltiple, proporciona una condición o punto de decisión que evalúa si es Verdadero, ejecuta un conjunto de acciones, según el caso.

#### **Diagrama de Flujo para una Estructura de Control Selectiva Múltiple**



Fuente: elaboración propia

### **De la teoría a la práctica**

**Ejemplo 1:** Ingresar un número del 1 al 5 y reporte la vocal que represente.

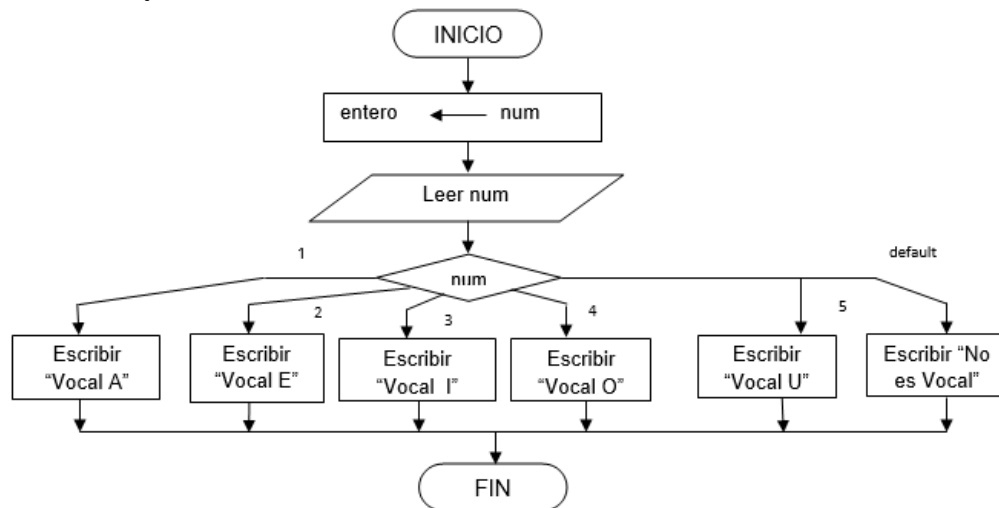


### Solución:

Definición de variables:

**num:** valor del numero entero positivo

Diagrama de Flujo:



Observe en el diagrama: la expresión en el rombo, permite el ingreso de un solo valor (no permite rango de valores por ejemplo >0), y según coincida el caso con el valor ingresado, se ejecuta el conjunto de acciones, finalizando y no permitiendo la ejecución de otro caso (usando break).

Código C/C++:

```

1  #include<iostream>
2  using namespace std;
3
4  void main( )
5  {
6      int num;
7
8      cout<<"Ingrese número: ";
9      cin>>num;
10
11     switch(num)
12     {
13         case 1: { cout<<"Vocal A";
14                     }break;
15
16         case 2: { cout<<"Vocal E";
17                     }break;
18
19         case 3: { cout<<"Vocal I";
20                     }break;
21
22         case 4: { cout<<"Vocal O";
23                     }break;
24
25         case 5: { cout<<"Vocal U";
26                     }break;
27
28         default: cout<<"No es numero para una vocal";
29     }
30 }
31
32
33
34
35

```

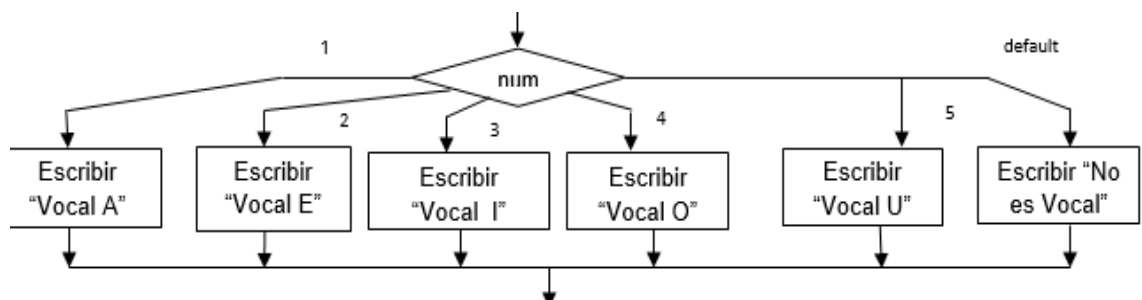
Fuente: elaboración propia

Si observa la figura, la variable "num", es de tipo entero, es decir (ejemplo: **int num;**), por lo que los casos evalúan ese valor como entero (ejemplo: **case 1:**) sin usar comilla simple.

Si se hubiese definido como caracter (ejemplo: **char num;**), la evaluación del caso hubiese sido considerando la comilla simple (ejemplo: **case '1':** )

Considerando el código C/C++ mostrado en la figura anterior, podemos comparar la representación en diagrama de flujo y su expresión en código.

**Diagrama de Flujo y Código C/C++  
de una Estructura de Control Selectiva Compuesta**



```

switch(num)
{
    case 1: {
        cout<<"Vocal A";
        }break;

    case 2: {
        cout<<"Vocal E";
        }break;

    case 3: {
        cout<<"Vocal I";
        }break;

    case 4: {
        cout<<"Vocal O";
        }break;

    case 5: {
        cout<<"Vocal U";
        }break;

    default: cout<<"No es numero para una vocal";
}
  
```

Fuente: elaboración propia

**Ejemplo 2:** Ingresar la inicial (mayúscula o minúscula) de un tipo de barco y reporte el nombre del mismo, y si no existe, reportar "TIPO DE BARCO NO ENCONTRADO".

### Solución:

Recuerde que el switch evalúa un valor dado, y ejecuta directamente el caso, ya no busca en los demás casos (break). Pero debe preguntarse, que pasaría si obvia la instrucción break de algún caso?, lógicamente no se rompe el flujo del switch, por lo que procesaría por defecto el siguiente caso con su respectivas instrucciones. Esto constituye una oportunidad para dar solución a este nuevo ejemplo, sin necesidad de emplear funciones especiales para diferenciar mayúsculas y minúsculas.

Identificar Variables a usar	Variable(s) a ingresar	Proceso (cálculo)	Variable(s) para reportar
tbarco	tbarco	<pre> switch(tbarco) {     case 'b':     case 'B': cout&lt;&lt;"El tipo es Buque"; break;      case 'c':     case 'C': cout&lt;&lt;"El tipo es Crucero"; break;      case 'f':     case 'F': cout&lt;&lt;"El tipo es Fragata"; break;      default: cout&lt;&lt;"No es un tipo de barco"; } </pre>	tbarco

Fuente: elaboración propia

Código C/C++:

```

1  #include<iostream>
2  using namespace std;
3
4  int main()
5  {
6      char tbarco;
7
8      cout<<"Ingrese tipo de barco: ";
9      cin>>tbarco;
10
11     switch(tbarco)
12     {
13         case 'b':
14         case 'B': cout<<"El tipo es Buque"; break;
15
16         case 'c':
17         case 'C': cout<<"El tipo es Crucero"; break;
18
19         case 'f':
20         case 'F': cout<<"El tipo es Fragata"; break;
21
22         default: cout<<"No es un tipo de barco";
23     }
24
25     return 0;
26 }

```

Fuente: elaboración propia

**Ejemplo 3:** Se requiere calcular el monto total de pago, luego de ingresar el nombre y género de un cliente, además de la cantidad de productos a adquirir, con su respectivo precio.

Para calcular el monto total a pagar, se asignan descuentos según el género y la cantidad de productos adquiridos, como se muestra en la siguiente tabla:

	Genero			
	M		F	
Cantidad	<=10	>10	<=10	>10
Porcentaje Descuento	0.2	0.5	0.3	0.4

### Solución:

Definimos variables, con un cuadro que le permita definir las tres partes del algoritmo de solución.

Observe que se definen todas las variables a usar tanto en el ingreso, proceso y salida.

Además, observa que en el proceso, se solicita ingresar la cantidad de producto, validando su ingreso con una estructura de control selectiva compuesta, garantizando que sea positivo.

También se solicita ingresar el género del cliente, y de igual forma, se valida su ingreso con una estructura de control selectiva compuesta, garantizando que solo se ingrese las iniciales de masculino y femenino en minúscula.

Luego, se usa una estructura de control selectiva múltiple, para validar por casos, el género del cliente y realizar la asignación del porcentaje de descuento de acuerdo a la cantidad de productos, verificada en una estructura de control selectiva compuesta.

Se procede a calcular el monto de pago, y se recomienda hacerlo por partes, a fin de tener mayor control de los datos en las variables.

Finalmente se procede a reportar o mostrar en pantalla, el valor calculado del pago, según el requerimiento del usuario.

Estimado estudiante, le recomiendo seguir este procedimiento de solución en los demás casos propuestos, es decir validando el ingreso y calculando por etapas, los valores finales de la variable.

Identificar Variables a usar	Variable(s) a ingresar	Proceso (cálculo)	Variable(s) para reportar
nombClie geneClie precioProd cantProd PorcDscto MontoBruto MontoDscto MontoPago	nombClie cantProd geneClie	<pre> Si(cantProd&gt;0) {     Leer geneClie;     si(geneClie == 'm'    geneClie == 'M'    geneClie == 'f'    geneClie == 'F')     {         Si(geneClie)         {             caso 'm':             caso 'M':                 {                     Si(cantProd &lt;= 10)                     PorcDscto = 0.2;                     Sino                         if(cantProd &gt; 10)                             PorcDscto = 0.5;                 }break;             caso 'f':             caso 'F':                 {                     Si(cantProd &lt;= 10)                     PorcDscto = 0.3;                     Sino                         if(cantProd &gt; 10)                             PorcDscto = 0.4;                 }break;         }          MontoBruto = cantProd * precioProd;         MontoDscto = MontoBruto * PorcDscto;         MontoPago = MontoBruto - MontoDscto;          Escribir "El monto bruto es: "         Escribir MontoBruto         Escribir "El monto descuento es: "         Escribir MontoDscto         Escribir "El monto Pago es: "         Escribir MontoPago      }     Sino         Escribir "ERROR. Debe ingresar F o M." } Sino     cout&lt;&lt;"ERROR. Debe ingresar mayor a cero.";</pre>	MontoBruto MontoDscto MontoPago

Código C/C++:

```

1  #include<iostream>
2  #include<string>
3  using namespace std;
4
5  int main()
6  {
7      string nombClie;
8      char geneClie;
9      double precioProd=100;
10     int cantProd;
11     double PorcDscto, MontoBruto, MontoDscto, MontoPago;
12
13     cout<<"Ingreso Nombre de Cliente: ";
14     cin>>nombClie;
15
16     cout<<"Ingreso Cantidad de Productos: ";
17     cin>>cantProd;
18
19     //Validar cantidad mayor a cero
20     if(cantProd>0)
21     {
22         cout<<"Ingreso Genero de Cliente: ";
23         cin>>geneClie;
24
25         //Validar que se ingrese genero
26         if(geneClie == 'm' || geneClie == 'M' || geneClie == 'f' || geneClie == 'F')
27         {
28             switch(geneClie)
29             {
30                 case 'm':
31                 case 'M':
32                 {
33                     if(cantProd <= 10)
34                         PorcDscto = 0.2;
35                     else
36                         if(cantProd > 10)
37                             PorcDscto = 0.5;
38                     }break;
39
40                 case 'f':
41                 case 'F':
42                 {
43                     if(cantProd <= 10)
44                         PorcDscto = 0.3;
45                     else
46                         if(cantProd > 10)
47                             PorcDscto = 0.4;
48                     }break;
49
50             }
51
52             MontoBruto = cantProd * precioProd;
53             MontoDscto = MontoBruto * PorcDscto;
54             MontoPago = MontoBruto - MontoDscto;
55
56             cout<<"El monto bruto es: ";
57             cout<<MontoBruto<<endl;
58             cout<<"El monto descuento es: ";
59             cout<<MontoDscto<<endl;
60             cout<<"El monto Pago es: ";
61             cout<<MontoPago<<endl;
62         }
63         else
64             cout<<"ERROR. Debe ingresar femenino o masculino.";
65     }
66     else
67         cout<<"ERROR. Debe ingresar mayor a cero.";
68
69     return 0;
70 }

```

Fuente: elaboración propia

## Glosario de la Unidad 1

**Acción:** sentencia de programación, puede ser secuencial, selectiva o repetitiva.

**Bloque:** en programación, un bloque es un conjunto de instrucciones o sentencias que pueden ser las estructuras de control secuencial, selectiva o repetitiva.

**Caracter:** tipo de dato de un lenguaje de programación, que solo permite un símbolo o letra.

**Módulo de Programa:** porción de código que se puede reutilizar en diferentes programas.

**Proceso:** también conocido como caja negra, representa a uno o más algoritmos de solución de un problema.

**Semántica:** el significado del conjunto de símbolos (léxico: tokens) de un lenguaje de programación, que tienen un orden particular.

**Sentencia:** también conocido como instrucción, representa la estructura de control para la programación.

**Sintaxis:** la forma, orden que se escriben el conjunto de símbolos (léxico: tokens) de un lenguaje de programación, que tienen un significado particular.

**Requerimiento:** solicitud o necesidad que tiene un cliente o usuario de programa, para ser solucionado con un programa de cómputo.

**Switch:** es la forma de escribir o sintaxis, de la estructura selectiva múltiple, y se puede entender como una compuerta que solo permite ingresar un valor y no un rango de valores.

## **Bibliografía de la Unidad 1**

Ceballos Sierra , F. (2006). C / C++ curso de programación (6ta ed.). México: AlfaOmega.

Joyanes Aguilar, L. (2008). Fundamentos de Programación (4ta ed.). España: McGraw-Hill.

Pressman, R. (2002). Ingeniería del software (5ta ed.). España: McGraw-Hill.

Rojas Moreno, C. (2013). Algorítmia y estructura de datos. Perú: Universidad Continental.





**HUANCAYO**

Av. San Carlos 1980  
Urb. San Antonio - Huancayo

Teléfono: 064 481430

**LOS OLIVOS - LIMA**

Av. Alfredo Mendiola 5210  
Los Olivos - Lima

Teléfono: 01 2132760

**MIRAFLORES - LIMA**

Jr. Junín 355  
Miraflores - Lima

Teléfono: 01 2132760

**AREQUIPA**

Av. Los Incas s/n  
Urb. Lambramani, José Luis  
Bustamante y Rivero - Arequipa

Teléfono: 054 412030

**CUSCO**

Av. Collasuyo Lote B-13  
Urb. Manuel Prado  
Wanchaq - Cusco

Teléfono: 084 480070