PRÁCTICA/LABORATORIO Nº 02

ESTRUCTURA GENERAL DE UN PROGRAMA

Indicaciones para el envío:

- Adjuntar un archivo **PDF** que contenga la solución de los problemas de la sección 1 y 2.
- Las imágenes deben ser de buena calidad/**legibles**.
- Se debe encajar **1 ejercicio por página**.
- Para la **implementación en C++,** adjuntar **captura de código** y **ejecución**.

Objetivos:

1. Repasar la teoría vista en clase.
2. Diseñar diagramas de flujo, puedes usar https://app.diagrams.net/.
3. Diseñar algoritmos en pseudocódigo.
4. Implementar algoritmos en el lenguaje de programación C++.

0. Marco Teórico

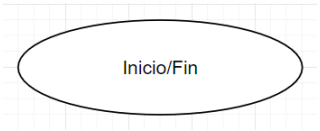
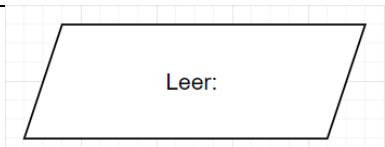
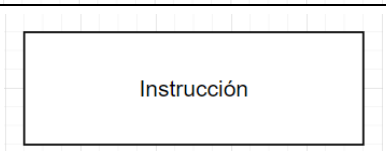
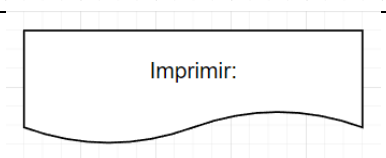
0.1 Algoritmo

Según Donald Knuth: "Un algoritmo es una secuencia **finita** de instrucciones **precisas** y **no ambiguas** para resolver un problema particular". La estructura general de un algoritmo es:

Datos de **entrada**, **procesamiento** de datos, **impresión** de resultados.

0.2 Diagramas de Flujo

Un diagrama de flujo esquematiza de forma **gráfica** un algoritmo. Muestra gráficamente las **instrucciones** para alcanzar la solución de un problema. A continuación, presentamos **solo los símbolos necesarios para resolver la presente práctica**.

Representación del Símbolo	Explicación del Símbolo
	Símbolo utilizado para marcar el **inicio** o **fin** de un diagrama de flujo.
	Símbolo utilizado para marcar la **lectura** de datos de **entrada**.
	Símbolo utilizado para representar una **instrucción**. Podemos expresar en su interior asignaciones, operaciones aritméticas o lógicas.
	Símbolo utilizado para representar la **impresión** en pantalla.

Recuerda que:

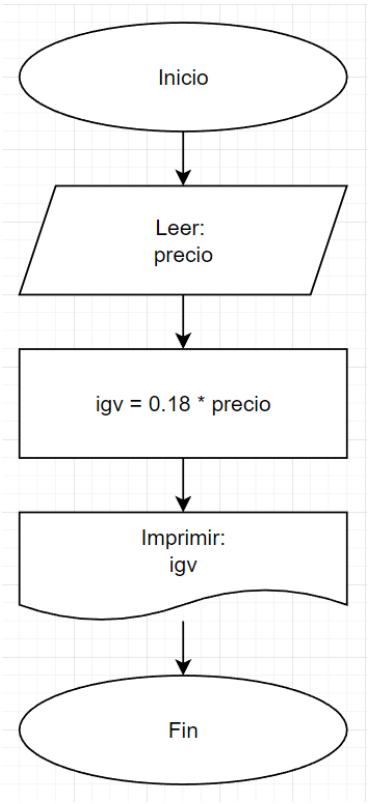
a) Todo diagrama de flujo debe tener un **inicio** y un **fin**.

b) Las líneas utilizadas para indicar dirección de flujo del diagrama deben ser rectas, **verticales u horizontales**.

c) Todas las líneas utilizadas para indicar la dirección del flujo del diagrama deben estar **conectadas**.

d) El diagrama de flujo debe ser construido de **arriba hacia abajo**.

Ejemplo: Cálculo del IGV.



0.3 Pseudocódigo

Según Donald Knuth: "El **pseudocódigo** es un lenguaje informal de propósito general que describe de manera **clara**, **precisa** y **concisa** cómo se debe llevar a cabo un algoritmo".

Podemos transformar el diagrama de flujo anterior en pseudocódigo:

```
Inicio
Leer precio
igv = 0.18 * precio
Imprimir igv
Fin
```

0.4 Tipos de Datos

Nos centraremos en los tipos de datos **simples**, los cuales pueden ser:

a) **Enteros**: Aquellos que **no tienen parte decimal**.

Ejemplo: 0, 1024, -512.

b) **Racionales**: Aquellos que tienen parte decimal **finita**.

Ejemplo: 7.5, -37.865, 16000.5, -15.0.

c) **Alfanuméricos**: Pueden ser de tipo carácter, cuyo contenido pueden ser letras (A-Z,a-z), dígitos (0-9) o símbolos especiales (#,$,^,*,%,/,!,+,-,…,etc.). Las cadenas son un conjunto de caracteres.

Ejemplo: **Carácter**: 'a', 'B', '$', '9', '-', '#', 'f'
Cadena: "abcde", "$9#7", "Pepe Paz", "052-583000"

d) **Lógicos**: Son los booleanos, que solo pueden tomar dos valores: verdadero (True) o falso (False).

Para el lenguaje de programación que estamos utilizando (C++), mencionaremos a continuación cómo están implementados estos tipos de datos:

Tipo de Dato	C++	Características
Enteros	`int`	De 4 bytes, almacena valores en el rango [-2147483648 - 2147483647]
	`long long`	De 8 bytes, almacena valores en el rango [-9,223,372,036,854,775,808 - 9,223,372,036,854,775,807]
Racionales	`float`	De 32 bits, con una precisión aproximada de 7 dígitos decimales.
	`double`	De 64 bits, con una precisión aproximada de 15-16 dígitos decimales.
Alfanuméricos	`char`	Permite representar un solo carácter. C++ utiliza para codificación UTF-8.
	`string`	Permite representar un conjunto de caracteres (char).
Booleanos	`Bool`	Permite representar verdadero (1) o falso (0).

0.5 Identificadores

Los identificadores permiten asignar un **nombre** a una casilla de memoria que almacena un **dato**. Tal y como hicimos en el punto 0.2, **igv** es un identificador que almacena el resultado de una expresión. A este espacio de memoria con identificador le llamamos **variable** si puede ser modificado (o constante si no).

En **C++**, las reglas para los nombres de las variables (identificadores) son las siguientes:

a) Los nombres de las variables deben **comenzar** con una letra o un guión bajo "_".

b) Los nombres de las variables pueden **contener** letras, números y guiones bajos "_".

c) Los nombres de las variables **distinguen entre mayúsculas y minúsculas**.

d) Los nombres de las variables **no pueden ser palabras clave reservadas** por el lenguaje de programación (por ejemplo, "if", "while", "int", "double", entre otros).

e) Los nombres de las variables deben ser únicos dentro del ámbito donde se definen.

Además de estas reglas básicas, también es importante seguir algunas **convenciones de estilo** para mejorar la **legibilidad** del código. Algunas convenciones comunes incluyen:

a) Utilizar nombres **descriptivos** que indiquen el **propósito** o **función** de la variable.

b) **Utilizar mayúsculas y minúsculas de forma consistente** para mejorar la legibilidad.

c) Utilizar nombres de variables en **singular** para representar objetos **datos simples** y en **plural** para representar **datos estructurados**.

d) **Evitar abreviaturas o nombres demasiado cortos** que puedan ser confusos o difíciles de entender.

Recuera que, en el lenguaje de programación C++, es necesario primero declarar la **variable** junto con el **identificador** y **tipo de dato** antes de hacer uso de esta.

Se puede declarar una **variable** con la siguiente sentencia:

```
<tipo de dato> <identificador>;
```

Si deseamos darle un **valor inicial** a la variable al momento de declararla, podemos hacer lo siguiente:

```
<tipo de dato> <identificador>=<expresión>;
```

Lo cual es equivalente a las siguientes sentencias:

```
<tipo de dato> <identificador>;

<identificador>=<expresión>;
```

También, en una sola sentencia, se pueden declarar **dos o más variables**:

```
<tipo de dato> <identificador1>,<identificador2>,…;
```

Si deseamos darle un valor inicial a una o más variables a declararse, podemos hacerlo en la misma sentencia separándolas por comas (,).

```
<tipo de dato> <identificador1>=<expresión>,<identificador2>,…;
```

Ejemplo:

Para el diagrama de flujo y pseudocódigo de los puntos 0.2 y 0.3 respectivamente:

```
#include<bits/stdc++.h>
using namespace std;
int main()
{
      float precio,igv;
      cout<<"Ingresar el precio: ";
      cin>>precio;
      igv=0.18*precio;
      cout<<"El IGV es: "<<igv;
      return 0;
}
```

0.5 Operaciones aritméticas

Las operaciones aritméticas requieren de operadores aritméticas y operan con números (Ej. 10,-0.9) y variables enteras o racionales (x, igv). A continuación, presentamos los operadores aritméticos, su implementación en C++ y un ejemplo con resultado.

Orden	Operación	Pseudocódigo	C++	Ejemplo	Resultado
1	Potencia	a^b ó a**b	`pow(a,b)`	3^2	9
1	Raíz cuadrada	√a	`sqrt(a)`	√4	2
1	Raiz n	$\sqrt[n]{b}$	`pow(b,1/n.)`	$\sqrt[3]{8}$	2
2	Multiplicación	*	*	6*4	24
2	División	/	/	8/2	4
2	Residuo	mod	%	8%3	2
3	Suma	+	+	6+3	9
3	Resta	-	-	6-3	3

La jerarquía de las operaciones sigue un orden, el **operador asociativo ()** tiene el orden **0**. En caso de existir **() anidados**, se evalúan **primero los de último nivel** de anidamiento. Se opera primero las operaciones de **menor orden a mayor orden**. De existir dos operaciones de igual orden, se opera de **izquierda a derecha**.

Ejemplo:

7+5-6 = 12-6 = 6

9+7*8-36/5 = 9+56-36/5 = 9+56-7.2 = 65-7.2 = 57.8

7*5**3/5%3 = 7*125/5%3 = 875/5%3 =175%3 = 1

7-8*(2+1)-28 = 7-8*3-28 = 7-24-28 = -45

Nota: En C++:

a) Una **división** de **enteros** devuelve un **número entero**, si la división no es exacta, devuelve la **parte entera** del resultado. Ejemplo: 3/2 = 1

b) Una **división** que incluye **un número racional**, devuelve un **número racional**.

Ejemplo: 3/2. = 1.5, 3./2 = 1.5, 3.0/2 = 1.5, 3/2.0 = 1.5, 3.0/2.0 = 1.5

c) La operación **residuo** o **módulo** solo está definido para operaciones entre números **enteros**.

0.6 Operadores relacionales

Son operadores que permiten comparar dos operandos, que deben ser del **mismo tipo**. El resultado es verdadero (1) o falso (0). A continuación, presentamos los operadores relacionales y un ejemplo.

Operación	Operador	C++	Ejemplo	Resultado
Igual que	=	==	12 = 11	Falso (0)
Diferente a	<>	!=	‘a’ <> ‘b>	Verdadero (1)
Menor que	<	<	7 < 15	Verdadero (1)
Mayor que	>	>	22 > 11	Verdadero (1)
Menor igual que	≤	<=	14 ≤ 11	Falso (0)
Mayor igual que	≥	>=	13 ≥ 10	Verdadero(1)

0.7 Operadores lógicos

Son operadores que permiten trabajar con expresiones lógicas. Estos son de conjunción (**and**), disyunción (**or**) o negación (**!**). Los resultados de estas operaciones se corresponden con las tablas de verdad de los operadores lógicos, donde la mayor jerarquía la tiene la operación de negación; además, tienen menor jerarquía que los operadores aritméticos.

1. Realice el <u>diagrama de flujo</u> e implementación en C++ de los siguientes problemas: NR=número racional, NE=número entero, ST=string

1.1 Dados los valores enteros A y B (leer), calcule (e imprima) la expresión (A-B)^2/3.

1.2 Dados los valores el código de postulante (ST), primer nombre (ST) y nota1 (NR) y nota2 (NR) de un estudiante, calcular el promedio de las notas.

1.3 Dados los valores el código de postulante (ST), primer nombre y nota1 (NR) y nota2 (NR) de un estudiante, calcular el promedio ponderado de las notas (nota1=60%, nota2=40%).

1.4 Dado el precio de 3 artículos (NR), calcular el precio del total.

1.5 Dado el precio sin igv (18%) de un artículo (NR), calcular el precio con igv.

1.6 Dado un número (NE), calcular el cuadrado y el cubo del número.

1.7 Dado un número (NE), calcular la raíz cuadrada y la raíz cúbica.

1.8 Dadas la base (NE) y altura (NE) de un triángulo, calcular el perímetro y área.

1.9 Dadas la base (NE) y altura (NE) de un cuadrado, calcular el perímetro y área.

1.10 Dado el precio de un artículo (NR) y la cantidad de dinero entregada por un cliente (NE), calcular el vuelto o cambio a entregarse.

1.11 Dado el nombre, peso en kg (NR). y estatura en cm (NE) de un estudiante, calcular su peso en libras y altura en pulgadas.

1.12 Dados los tres lados de un triángulo (NE), calcular su perímetro y área.

2. Realice el pseudocódigo e implementación en C++ de los siguientes problemas:

2.1 Dado el radio (NE) y altura de un cono (NE), calcular su área y volumen.

2.2 Dado un número de días (NE), calcular el número de segundos que hay en este.

2.3 Dados los puntos (x1, y1) y (x2, y2) (NE), calcular la distancia euclidiana entre ambos puntos.

2.4 Dados los puntos (x1, y1, z1) y (x2, y2, z2) (NE), calcular la distancia euclidiana entre ambos puntos.

2.5 Dada una cantidad de minutos (NE), convertir el valor a horas:minutos.

2.6 Dada una cantidad de horas (NE) y un máximo de horas de trabajo por día (NE), calcular cuántos días de trabajo se requiere.

2.7 Dado el valor de x (NR), calcular x^4-3x^3+2x^2-9.

2.8 Dados los valores de los catetos de un triángulo rectángulo (NE), calcular su hipotenusa.

2.9 Dado un número n (NE), calcular la suma desde 1 hasta n.

2.10 Dado un número n (NE), calcular la suma de los n primeros números impares.

2.11 Dado x e y (NR), calcular el resultado de ((4^x - 3^x) / (4^y + 4*x + 3^y)).

2.12 Dado el precio en galones del petróleo (NR) y el tamaño del depósito en litros (NE), calcular el precio a pagar por llenar el depósito por completo.

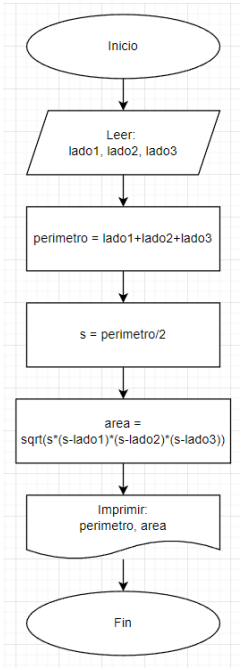
PRÁCTICA Nª 02

ESTRUCTURA GENERAL DE UN PROGRAMA

Nombres y Apellidos: _____Código: _____

1.12 Dados los valores de A y B (leer), calcule (e imprima) la expresión (A-B)^2/3.

1.12.1 Diagrama de flujo:



1.24.1 Implementación en C++:

```
1  #include<bits/stdc++.h>
2
3  using namespace std;
4
5  int main()
6  {
7      int lado1,lado2,lado3;
8      int perimetro,s;
9      float area;
10     cout<<"Ingrese el lado 1: "; cin>>lado1;
11     cout<<"Ingrese el lado 2: "; cin>>lado2;
12     cout<<"Ingrese el lado 3: "; cin>>lado3;
13     perimetro=lado1+lado2+lado3;
14     s=perimetro/2.;
15     area=sqrt(s*(s-lado1)*(s-lado2)*(s-lado3));
16     cout<<"El perimetro es: "<<perimetro<<endl;
17     cout<<"El area es: "<<area<<endl;
18     return 0;
19 }
```

```
C:\Users\Israel Chaparro\Desktop\1.12.exe
Ingrese el lado 1: 6
Ingrese el lado 2: 7
Ingrese el lado 3: 11
El perimetro es: 24
El area es: 18.9737

--------------------------------
Process exited after 1.927 seconds with return value 0
Presione una tecla para continuar . . .
```

2.9 Dado un número n (NE), calcular la suma desde 1 hasta n.

2.9.1 Pseudocódigo:

```
Inicio
Leer n
suma=n*(n+1)/2
Imprimir suma
Fin
```

2.9.1 Implementación en C++:

```cpp
#include<bits/stdc++.h>
using namespace std;
int main()
{
    int n,suma;
    cout<<"Ingrese el valor de n: ";
    cin>>n;
    suma=n*(n+1)/2;
    cout<<"La suma desde 1 hasta n es: "<<suma;
    return 0;
}
```

```
C:\Users\Israel Chaparro\Desktop\2.9.exe
Ingrese el valor de n: 100
La suma desde 1 hasta n es: 5050
--------------------------------
Process exited after 0.9066 seconds with return value 0
Presione una tecla para continuar . . .
```