TRABAJO DE PROGRAMACION NUMERICA

PRESENTADO POR

ALBERTO AFRICANO

JONATHAN CAMPBELL

JORGE DONADO

OSCAR HERNANDEZ

DOCENTE

NICOLAS RUEDA PIMIENTO

UNIVERSIDAD DE LA COSTA

CUC

BARRANQUILLA /COLOMBIA

2021

TALLER DE PROGRAMACION NUMERICA
2021-2

Profesor: Nicolás Rueda Pimiento.

Presentar de forma escrita en **grupos de máximo 3 integrantes** para el día **Lunes 30 de agosto de 2021.** Por favor LEER y CUMPLIR las siguientes condiciones de entrega de la actividad en el aula virtual:

1. VER el video adjunto en esta tarea donde se muestra cómo debe crear el PDF de la actividad, a partir de llenar un archivo WORD y luego convertirlo a PDF. SOLAMENTE RECIBO EL TALLER PRESENTADO DE ESTA FORMA EXPLICADA EN EL VIDEO, UN ÚNICO PDF.
2. Debe indicar tal como se muestra en el video una hoja de presentación y aparezcan allí los nombres de los integrantes. En caso de que no los anote allí, NO SE TENDRÁ EN CUENTA COMO INTEGRANTE DEL GRUPO QUIEN NO APAREZCA EN ESA HOJA BAJO NINGUNA EXCUSA.
3. Ejercicios referentes a preguntas (teoría, justificar) puede redactar allí mismo en su documento word las respuestas, pegando la captura del enunciado o simplemente haciendo referencia al ejercicio al redactar su respuesta y pegando luego las imágenes de su justificación.

EJERCICIOS

1. Dar definición y ejemplos para los siguientes conceptos:
   (a) Diagramas de flujo para la representación de algoritmos.
   (b) Tipos de variables: variable booleana, variable tipo string (cadena).
   (c) En C++ , ¿cuál es la jerarquía de las operaciones aritméticas?
2. Escriba un algoritmo y luego su respectivo código en `lenguaje C++` para cada uno de los siguientes problemas:
3. Para cada uno de los siguientes problemas debe realizar: pseudocódigo, diagrama de flujo y el correspondiente código en lenguaje (`C++`):

   (a) Dada distancia recorrida en millas por un vehículo y el tiempo empleado para ello en segundos, convertir a kilómetros y horas respectivamente y retornar en pantalla el valor de las conversiones y la velocidad calculada con dichos datos.
   (b) Ingresar el costo de una vivienda nueva diseñada por una constructora e indicar que según los siguientes acabados agregados con costo adicional cuánto deberá pagar de monto total, si:
   - Piso en porcelanato incrementa el costo de la vivienda en 15%.
   - Baños completamente terminados, incrementa en 10% el costo.
   - Cocina con horno y enchape, aumenta en un 9% el costo.

   (c) Convertir pulgadas a: pies y a millas. Mostrar como salida esas dos conversiones calculadas.
   (d) Si una persona se gana el premio de 100 millones de pesos en una lotería, la DIAN le aplica impuesto al patrimonio por 16%. Luego, si la secretaría de hacienda del 9% del valor anterior, muestre en pantalla cuánto dinero realmente recibirá dicha persona como premio.

DESARROLLO

1.Dar definición y ejemplos para cada uno de los siguientes conceptos:

(a) Diagramas de flujos para la representación de algoritmos.

DIAGRAMA DE FLUJOS

El **diagrama de flujo** o **diagrama de actividades** es la representación gráfica del algoritmo o proceso. Se utiliza en disciplinas como programación, economía, procesos industriales y psicología cognitiva.

En Lenguaje Unificado de Modelado (UML), un diagrama de actividades representa los flujos de trabajo paso a paso de negocio y operacionales de los componentes en un sistema. Un diagrama de actividades muestra el flujo de control general.

En SysML el diagrama de actividades ha sido extendido para indicar flujos entre pasos que mueven elementos físicos (e.g., gasolina) o energía (e.g., presión). Los cambios adicionales permiten al diagrama soportar mejor flujos de comportamiento y datos continuos.

Estos diagramas utilizan símbolos con significados definidos que representan los pasos del algoritmo, y representan el flujo de ejecución mediante flechas que conectan los puntos de inicio y de fin de proceso.

CARACTERISTICAS

Un diagrama de flujo siempre tiene un único punto de inicio y un único punto de término.

Las siguientes son acciones previas a la realización del diagrama de flujo:

- Identificar las ideas principales a ser incluidas en el diagrama de flujo. Deben estar presentes el autor o responsable del proceso, los autores o responsables del proceso anterior y posterior y de otros procesos interrelacionados, así como las terceras partes interesadas.
- Definir qué se espera obtener del diagrama de flujo.
- Identificar quién lo empleará y cómo.
- Establecer el nivel de detalle requerido.
- Determinar los límites del proceso a describir.

Los pasos a seguir para construir el diagrama de flujo son:

- Establecer el alcance del proceso a describir. De esta manera quedará fijado el comienzo y el final del diagrama. Frecuentemente el comienzo es la salida del proceso previo y el final la entrada al proceso siguiente.

- Identificar y listar las principales actividades/subprocesos que están incluidos en el proceso a describir y su orden cronológico.
- Si el nivel de detalle definido incluye actividades menores, listarlas también.
- Identificar y listar los puntos de decisión.
- Construir el diagrama respetando la secuencia cronológica y asignando los correspondientes símbolos.
- Asignar un título al diagrama y verificar que esté completo y describa con exactitud el proceso elegido

DESCRIPCIÓN

En UML, un diagrama de actividades es una variación del diagrama de estado UML donde los «estados» representan operaciones, y las transiciones representan las actividades que ocurren cuando la operación es completa.

El diagrama de actividades UML 2.0, mientras que es similar en aspecto al diagrama de actividades UML ahora tiene semánticas basadas en redes de Petri. En UML 2.0, el diagrama general de interacción está basado en el diagrama de actividades. El diagrama de actividad es una forma especial de diagrama de estado usado para modelar una secuencia de acciones y condiciones tomadas dentro de un proceso.

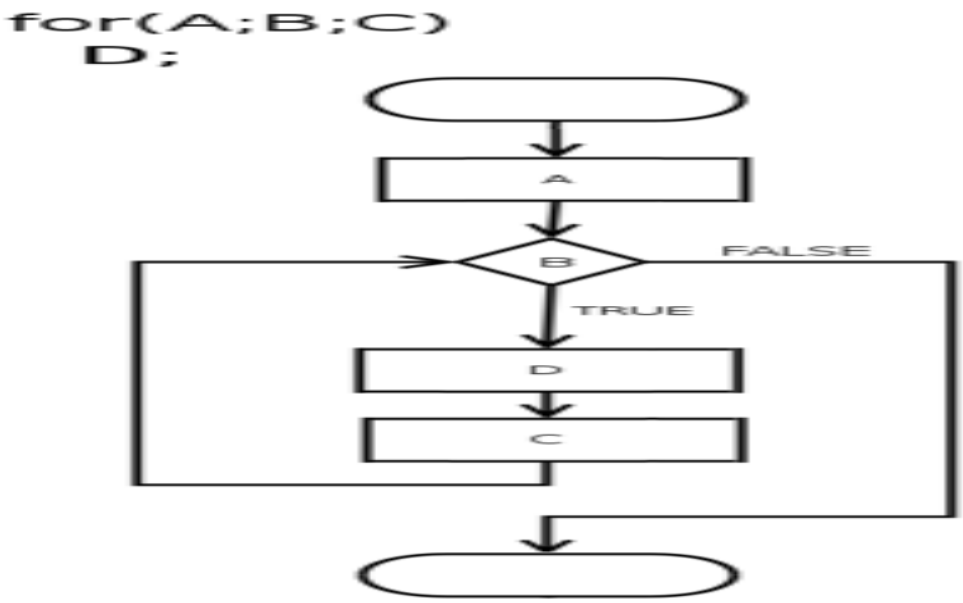
La especificación del Lenguaje de Modelado Unificado (UML) define un diagrama de actividad como:

"… una variación de una máquina estados, lo cual los estados representan el rendimiento de las acciones o subactividades y las transiciones se provocan por la realización de las acciones o subactividades."

El propósito del diagrama de actividad es modelar un proceso de flujo de trabajo (workflow) y/o modelar operaciones.

Una Operación es un servicio proporcionado por un objeto, que está disponible a través de una interfaz.

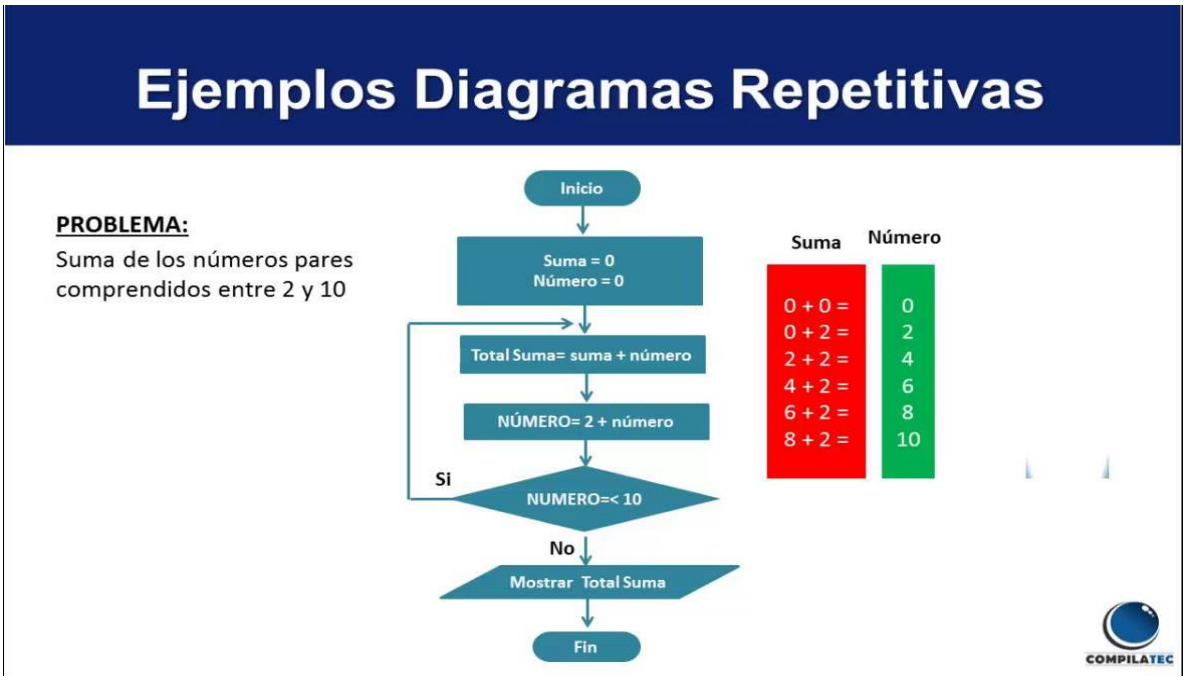
Una Iterfaz es un grupo de operaciones relacionadas con la semántica



TIPOS DE DIAGRAMAS DE FLUJOS

- Formato vertical: En él, el flujo y la secuencia de las operaciones, va de arriba hacia abajo. Es una lista ordenada de las operaciones de un proceso con toda la información que se considere necesaria, según su propósito.
- Formato horizontal: En él, el flujo o la secuencia de las operaciones, va de izquierda a derecha.
- Formato panorámico: El proceso entero está representado en una sola carta y puede apreciarse de una sola mirada mucho más rápido que leyendo el texto, lo que facilita su comprensión, aún para personas no familiarizadas. Registra no solo en línea vertical, sino también horizontal, distintas acciones simultáneas y la participación de más de un puesto o departamento que el formato vertical no registra.Formato Arquitectónico: Describe el itinerario de ruta de una forma o persona sobre el plano arquitectónico del área de trabajo. El primero de losflujogramas es eminentemente descriptivo, mientras que los utilizados son fundamentalmente representativos

EJEMPLO



(b) TIPOS DE VARIABLES; VARIABLE BOOLEANA,VARIABLE TIPO STRING(CADENA)

Las variables booleanas
Nos referimos a variables booleanas a aquellas que utilizamos para indicar que algo es verdadero o falso.

Lo importante de estas variables es que sólo pueden tomar dos valores: verdadero o falso, no hay más posibilidades y esta propiedad la utilizaremos para tomar decisiones, por ejemplo: tengo una variable que se llama bTarifaMenor y que es verdadera si tengo una variable edad de tipo entero <= 12 y en caso contrario será falsa. Podríamos decir que si la variable bTarifa Menor es verdadera aplicaremos un descuento del 20% en el precio de un producto.

La definición de estas variables dependerá del lenguaje de programación que estemos utilizando. No podemos encontrar con los tipos: booblean, bool, Boolean, Bool. En cuanto tengamos claro cual de los tipos es el que utiliza el lenguaje con el que estamos programando definiremos la variable:

Para c# definiríamos así:
bool bTarifaMenor; //definimos la variable booleana
bTarifaMenor = False; //inicializamos la variable con valor falso

int edadCliente = 13; //definimos e inicializamos la variable edad en 13

Para aplicar la lógica de decisión que evalúa el valor de la variable edadCliente utilizamos

la instrucción if que sigue la siguiente sintaxis:

```
if (condición que se evalúa)
{
  //Entre estas llaves va el conjunto de instrucciones que se ejecutan si se cumple
  //la condición que se evalúa
}
```

aplicando la instrucción de evaluación if en el ejemplo:

```
//la condición de evaluación es que la edad del cliente sea menor o igual que 12
//dejamos la referencia de los operadores disponibles que iremos aplicando en los
ejemplos
if (edadCliente <= 12)
{
  bTarifaMenor = True; //asignamos el valor de verdadero a la variable booleana
}
```

Las llaves "{" y "}" contienen un conjunto de instrucciones que sólo se ejecutarán si se cumple la condición que se evalúa dentro del paréntesis, si ese conjunto de instrucciones consta de una sola instrucción como es el caso del ejemplo, se pueden obviar las llaves:

```
if (edadCliente <= 12)
  bTarifaMenor = True; //asignamos el valor de verdadero a la variable booleana
```

Sin embargo, si Ud. está empezando a programar le recomiendo que utilice siempre las llaves, así podrá distinguir claramente cuáles son las instrucciones que se ejecutarán en el caso que se cumpla la condición del if.

Para terminar el ejemplo, aplicaremos el descuento a los menores utilizando como condicional a la variable booleana:

```
if (bTarifaMenor)
{
 //aquí hace las instrucciones necesarias para aplicar el 20% de descuento en el precio
}
```

En el caso de este if la condición que se evalúa es simplemente una variable booleana, por que en sí esta variable es True o False, en el caso que su valor sea True se ejecutará el bloque de instrucciones que se encuentra entre las llaves (descuento en el precio).

Variables de tipo string

Un string en realidad es un conjunto ordenado de letras (caracteres). Supongamos que tienes:

```
C++
string nombre = "Juan";
```

ombre es una variable que contiene 4 caracteres: la J, la u, la a y la n.

Se puede acceder a cada uno de los elementos de este conjunto refiriéndose a la posición que ocupan, empezando por el *0*. Es decir:
Puedes acceder a la J con *nombre[0]*
Puedes acceder a la u con *nombre[1]*
Puedes acceder a la a con *nombre[2]*
Puedes acceder a la n con *nombre[3]*

También puedes modificar las posiciones, por ejemplo, "*nombre[0] = 'X'*" hará que *nombre* pase a ser "*Xuan*".

Podemos modificar letras, pero no podemos añadir ni eliminar letras, es decir, si hago "*nombre[4] = 'a';*" el programa probablemente fallará, porque la variable nombre es un conjunto de 4 elementos (numerados del *0* hasta al *3*), no existe ningún elemento en la posición *4* y por tanto no podemos hacer una modificación en esa posición.

Fíjate que para poner una letra ponemos comillas simples, no dobles. El motivo es que las comillas dobles son para *string*s, y las simples para caracteres individuales. Es decir, cuando pongo comillas dobles estoy diciendo que lo que hay entre comillas es un conjunto de elementos, mientras que cuando pongo comillas simples estoy diciendo que lo que hay entre comillas es un único elemento, que es un carácter.

Para saber cuantas letras tiene un string debes de usar su subinstrucción "*size()*", por ejemplo, *nombre.size()* es *4* porque "*Juan*" tiene 4 letras. Dicho de otra manera, cuando una variable contiene un conjunto, puedes utilizar el nombre de la variable seguido de "*.size()*" para conocer cuántos elementos contiene este conjunto.

Podemos usar una variable de tipo *int* para indicar una posición:

C++

```
string nombre = "juan";
int posicion;
cout << "Indica una posicion y la covnertire en mayuscula: ";
cin >> posicion;
nombre[posicion] = nombre[posicion] - 32;
cout << nombre;
```

(C) EN C++, ¿CUAL ES LA JERARQUIA DE LAS OPERACIONES ARITMETICA?

jerarquia de operadores C++
OPERADORES ARITMÉTICOS.

Sirven para realizar operaciones aritméticas básicas. Los operadores aritméticos C siguen las reglas algebraicas típicas de jerarquía o prioridad.

Tabla de Operadores Aritméticos.

OPERACIÓN EN C	OPERADOR ARITMÉTICO	EXPRESIÓN ALGEBRAICA	EXPRESIÓN EN C
Suma	+	f + 7	f + 7
Resta	-	p - c	p - c
Multiplicación	*	bm	b * m
División	/	x/y o $\frac{x}{y}$	x / y
Módulo	%	r mód s	r % s
Potencia	pow	x^2	pow (a , b)

Los paréntesis se utilizan en las expresiones de C de manera muy similar a como se usan en las expresiones algebraicas, sirven para indicar que la expresión dentro de ellos se debe realizar primero. Por ejemplo, para multiplicar a por la cantidad b+c escribimos:

a * (b + c)

	Operador (es)	Operacion (es)	Orden de cálculo (precedencia)
MAYOR ↓ MENOR	()	Paréntesis	Se calculan primero. Si los paréntesis están anidados, la expresión en el par más interno se evalúa primero. Si existen varios pares de paréntesis "en el mismo nivel" (es decir no anidados), se calcularán de izquierda a derecha.
	pow	Función Pow	Se escribe la función y luego entre paréntesis la(s) que servirán como base, luego se escribe una coma (,) y a continuación el exponente.
	*, /, %	Multiplicación, División y Módulo	Se evalúan en segundo lugar. Si existen varias, se calcularán de izquierda a derecha.
	+, -	Suma o Resta	Se calculan al último. Si existen varios, serán evaluados de izquierda a derecha.

Pow es la función que permite realizar las potencias. Por ejemplo x=23 se podría presentar en C de las maneras siguientes: x=pow (2 , 3); o x=2*2*2;

Las reglas de precedencia o jerarquía de operadores son guías de acción, que le permiten a C calcular expresiones en el orden correcto. Cuando decimos que una evaluación o cálculo avanza de izquierda a derecha, nos estamos refiriendo a la asociatividad de los operadores.

Ejemplo 1:

El siguiente ejemplo contiene módulo (%), multiplicación, división, adición y sustracción.

Algebraicamente: z= pr mod q + w ÷ / x - y

En C: z = p * r % q + w / x – y;

1, 2, 4, 3, 5

Los números de las operaciones indican el orden en el cual C valorará o calculará los operadores. La multiplicación, el módulo y la división, serán evaluadas primero, en un orden de izquierda a derecha (es decir, se asocian de izquierda a derecha), en vista de que tienen precedencia mayor que la suma y la resta. La suma y la resta serán calculadas después. También ellas se evaluarán de izquierda aderecha.

No todas las expresiones con varios pares de paréntesis contienen paréntesis anidados. La

expresión a * (b + c) + c * (d + e) no contiene paréntesis anidados. En vez de ello, se dice que estos paréntesis están “en el mismo nivel”. En ese caso, se calcularán las expresiones del paréntesis primero y en un orden de izquierda a derecha

Ejercicio 3.

3. Para cada uno de los siguientes problemas debe realizar: pseudocódigo, diagrama de flujo y el correspondiente código en lenguaje (C++):

Ejercicio 3. A.

pseudocódigo:

```
var
      v:numerico
inicio
      imprimir("Ingrese la velocidad en millas/segundo: ")
      leer(v)
      v=v*5794
      imprimir("\nLa velocidad en kilometros/hora es: ",v)
fin
```

diagrama de flujo:

código en lenguaje (C++):

```
#include <iostream>
using namespace std;
int main() {
    float v;
    cout << "DIngrese la velocidad en millas/segundo: ";
    cin>>v;
    v=v*5794;
    cout << "\nLa velocidad en kilometros/hora es: "<<v;
    cout<<endl;
    return 0;
}
```

Ejercicio 3. B.

pseudocódigo:

```
var
      v:numerico
inicio
      imprimir("Ingrese el precio base: ")
      leer(v)
      v=v*1.34
      imprimir("\nEl precio final es: ",v)
fin
```

diagrama de flujo:

código en lenguaje (C++):

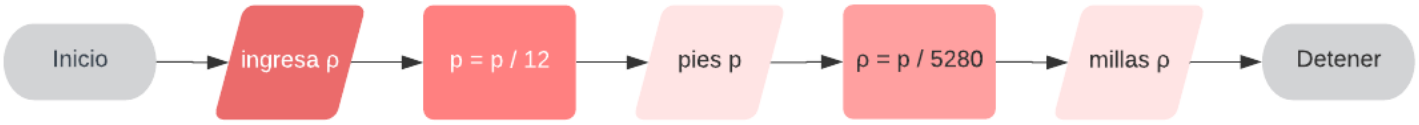
```
#include <iostream>
using namespace std;
int main() {
    float v;
    cout << "Ingrese el precio base: ";
    cin>>v;
    v=v*1.34;
    cout << "\nEl precio final es: "<<v;
    cout<<endl;
    return 0;
}
```

Ejercicio 3. C.

pseudocódigo:

```
var
      p:numerico
inicio
      imprimir("Ingrese la longitud en PULGADAS para convertir a PIES y a MILLAS: ")
      leer(p)
      imprimir("Pulgadas: ",p)
      p=p/12
      imprimir("\nPies: ",p)
      p=p/5280
      imprimir("\nMillas: ",p)
fin
```

diagrama de flujo:



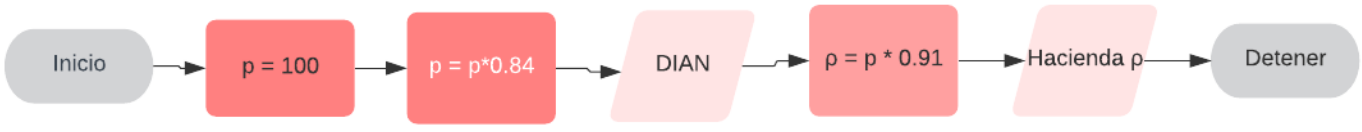
código en lenguaje (C++):

```
#include <iostream>
using namespace std;
int main() {
    float p;
    cout << "Ingrese la longitud en PULGADAS para convertir a PIES y a MILLAS: ";
    cin>>p;
    cout << "\nPulgadas: "<<p<<endl;
    p=p/12;
    cout << "\nPies: "<<p<<endl;
    p=p/5280;
    cout << "\nMillas: "<<p<<endl;
    return 0;
}
```

pseudocodigo:

```
var
      p:numerico
inicio
      p=100
      imprimir("Dinero ganado: ",p)
      p=p*0.84
      imprimir("\nDinero despues de la DIAN: ",p)
      p=p*0.91
      imprimir("\nDinero despues de Hacienda: ",p)
      imprimir("\nTotal dinero recibido: ",p)
fin
```

diagrama de flujo:



código en lenguaje (C++):

```
#include <iostream>
using namespace std;
int main() {
    float p=100;
    cout << "Dinero ganado: "<<p;
    p=p*0.84;
    cout << "\nDinero despues de la DIAN: "<<p;
    p=p*0.91;
    cout << "\nDinero despues de Hacienda: "<<p;
    cout << "\nTotal dinero recibido: "<<p;
    return 0;
}
```