



Programación I

Elementos básicos de un programa computacional

Retroalimentación

Características de un algoritmo:

Retroalimentación

Características de un algoritmo:

- Debe ser **finito**, posee un número determinado de pasos.
- Debe ser **preciso**, no puede contener ambigüedades.
- Debe ser **definido**, si se sigue dos veces, se debe llegar al mismo resultado.
- Se estructura en tres partes, **Entrada**, **Proceso** y **Salida**.

Retroalimentación

Fases en la resolución de problemas

Los puntos más importantes de fases son:

Retroalimentación

Fases en la resolución de problemas

Los puntos más importantes de fases son:

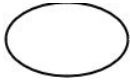




- **Análisis:** Se analiza el problema tomando en cuenta todas las especificaciones de los requisitos dado por el cliente o por la persona que encarga el programa.
- **Diseño:** se diseña la solución, el cual dará como resultado un algoritmo que resuelve el problema.
- **Codificación:** La solución se escribe detalladamente las instrucciones generando el código fuente, este es escrito en un lenguaje de programación de alto nivel.
- **Ejecución, verificación y depuración:** Se realiza la ejecución del programa, se comprueba rigurosamente y se eliminan todos los errores que puedan aparecer.
- **Documentación:** Se realiza la escritura del ciclo de vida del software, donde se detalla el análisis, diseño y codificación, junto con manuales y referencias.
- **Mantenimiento:** Se realizan actualizaciones al programa, cada vez que sea necesario.

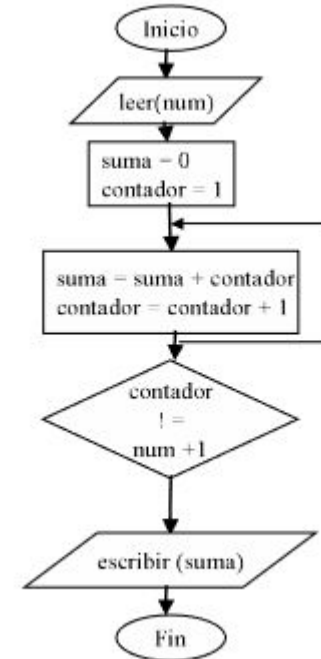
Pseudocódigo

- Leer permite asignar valores de **entrada**.
- el símbolo <- es de **asignación**, donde la variable área se le asigna el resultado de la expresión $\text{base} * \text{altura}$.
- Escribir realiza la operación de **salida** que puede ser un mensaje un valor.
- FinAlgoritmo define el fin del algoritmo.

```
Algoritmo calcular_area  
  Leer base, altura  
  area←base*altura  
  Escribir area  
FinAlgoritmo
```

Diagrama de flujo

	Inicio y fin de algoritmo
	Secuencia
	Operaciones de entrada y salida
	Procesos
	Bifurcaciones



Ejercicios

Problema 1: Se necesita un programa que indique si es mayor de edad el usuario o no, la edad la debe ingresar el usuario.

Problema 2: Realice un programa donde se solicita al usuario ingresar 2 números, luego debe devolver el número mayor, en el caso de ser iguales muestre un mensaje por pantalla indicando que son iguales.

Problema 3: Realiza un programa donde el usuario ingrese la edad del cliente y muestre el monto que éste debe pagar para poder entrar a una tienda de videojuegos, si el cliente es menor a 8 años entran gratis, si tiene entre 8 y 12 debe pagar \$800 y si su edad es mayor a 12 debe pagar \$1500.

Problema 4: Realiza un programa donde el usuario ingrese 4 notas y debe calcular su promedio ponderado, la primera nota tiene ponderación de 20%, la segunda tiene una ponderación de 30%, la tercera nota tiene una ponderación de 30% y la cuarta nota tiene una ponderación de 20%.

Ejercicios

Problema 1: Se necesita un programa que indique si es mayor de edad el usuario o no, la edad la debe ingresar el usuario.

Ejercicios

Problema 1: Se necesita un programa que indique si es mayor de edad el usuario o no, la edad la debe ingresar el usuario.

Análisis

- Entrada: edad
- salida: mensaje_mayor

Operaciones:

Diseño

```
Algoritmo mayor_edad
  escribir "Ingrese su edad"
  leer edad
  si edad >=18 entonces
    mensaje_mayor <- "Es mayor de edad"
    escribir mensaje_mayor
  sino
    mensaje_mayor <- "No es mayor de edad"
    escribir mensaje_mayor
finalgoritmo
```

Ejercicios

Problema 2: Realice un programa donde se solicita al usuario ingresar 2 números, luego debe devolver el número mayor, en el caso de ser iguales muestre un mensaje por pantalla indicando que son iguales.

Ejercicios

Problema 2: Realice un programa donde se solicita al usuario ingresar 2 números, luego debe devolver el número mayor, en el caso de ser iguales muestre un mensaje por pantalla indicando que son iguales.

Análisis

- Entrada: numero_1, numero_2
- salida: mensaje_numero_mayor

Operaciones:

Diseño

Algoritmo numero_mayor

```
    leer numero_1, numero_2
    Si numero_1 > numero_2 entonces
        escribir "El numero mayor es", numero_1
    sino
        si numero_2 > numero_1 entonces
            escribir "El numero mayor es:", numero_2
        sino
            escribir "ambos numero son iguales"
finalgoritmo
```

Ejercicios

Problema 3: Realiza un programa donde el usuario ingrese la edad del cliente y muestre el monto que éste debe pagar para poder entrar a una tienda de videojuegos, si el cliente es menor a 8 años entran gratis, si tiene entre 8 y 12 debe pagar \$800 y si su edad es mayor a 12 debe pagar \$1500.

Ejercicios

Problema 3: Realiza un programa donde el usuario ingrese la edad del cliente y muestre el monto que éste debe pagar para poder entrar a una tienda de videojuegos, si el cliente es menor a 8 años entran gratis, si tiene entre 8 y 12 debe pagar \$800 y si su edad es mayor a 12 debe pagar \$1500.

ANÁLISIS

entrada: edad

salida: mensaje_costo

Operaciones:

DISEÑO

Algoritmo costo_entrada

 leer edad

 si edad < 8 entonces

 escribir "entra gratis"

 sino

 si edad >= 8 y edad <= 12 entonces

 escribir "Debe pagar \$800 para poder entrar"

 sino

 escribir "Debe pagar \$1500 para poder entrar"

finalgoritmo

Ejercicios

Problema 4: Realiza un programa donde el usuario ingrese 4 notas y debe calcular su promedio ponderado, la primera nota tiene ponderación de 20%, la segunda tiene una ponderación de 30%, la tercera nota tiene una ponderación de 30% y la cuarta nota tiene una ponderación de 20%.

Ejercicios

Problema 4: Realiza un programa donde el usuario ingrese 4 notas y debe calcular su promedio ponderado, la primera nota tiene ponderación de 20%, la segunda tiene una ponderación de 30%, la tercera nota tiene una ponderación de 30% y la cuarta nota tiene una ponderación de 20%.

ANÁLISIS

entrada: nota1, nota2, nota3, nota4

salida: prom_pond

Operaciones:

```
nota1_pond <- nota1*0.2
nota2_pond <- nota2*0.3
nota3_pond <- nota3*0.3
nota4_pond <- nota4*0.2
prom_pond <- nota1_pond+nota2_pond+nota3_pond+nota4_pond
```

DISEÑO

Ejercicios

Problema 4: Realiza un programa donde el usuario ingrese 4 notas y debe calcular su promedio ponderado, la primera nota tiene ponderación de 20%, la segunda tiene una ponderación de 30%, la tercera nota tiene una ponderación de 30% y la cuarta nota tiene una ponderación de 20%.

DISEÑO

Proceso calc_ponderado

 Escribir "Ingrese Primera nota:"

 Leer nota1

 Escribir "Ingrese Segunda nota:"

 Leer nota2

 Escribir "Ingrese Tercera nota:"

 Leer nota3

 Escribir "Ingrese Cuarta nota:"

 Leer nota4

 nota1_pond <- nota1*0.2

 nota2_pond <- nota2*0.3

 nota3_pond <- nota3*0.3

 nota4_pond <- nota4*0.2

 prom_pond <- nota1_pond+nota2_pond+nota3_pond+nota4_pond

 escribir "Su nota ponderada es de: ", prom_pond

FinProceso

Escritura de algoritmos/programas

La escritura de algoritmo mediante una herramienta de programación debe ser lo más clara posible y estructurada. Esto facilita el entendimiento del algoritmo y su posterior codificación en un lenguaje de programación.

Dependiendo el lenguaje con el que se trabaje el algoritmo, en pseudocódigo tendrá una estructura similar debido a que requerirá la lógica de este. Para el caso de python se deben indentar para estructurar el pseudocódigo

Problema: Calcular la media de tres números pedidos por teclado

```
Algoritmo calculo_perimetro
....leer numero_1
....leer numero_2
....leer numero_3
....media <- (numero_1+numero_2+numero_3)/3
....escribir media
Fin
```

Tipos de datos

Los tipos de datos son la propiedad de un valor que determina su dominio, indicando que valores puede tomar, qué operaciones se le pueden aplicar y cómo este puede ser representado internamente por el computador.

Los tipos de datos se pueden clasificar en:

- **Datos Numéricos:** El tipo de dato numérico es el conjunto de valores numéricos, donde se representan números enteros y los reales. Este tipo de datos permiten realizar operaciones aritméticas comunes.
- **Datos Lógicos:** Los datos de tipo lógico o también llamados booleano, solo pueden tomar dos valores, verdadero y falso
- **Datos Alfanuméricos(String):** Los tipos alfanuméricos o de tipo carácter y tipo cadena, se encuentra dentro del conjunto finito y ordenado de caracteres que la computadora reconoce. Un dato del tipo carácter sólo contiene un carácter.

Datos de tipo numérico

Los datos de tipo numérico se pueden representarse de dos formas:

Enteros: Este se encuentra en el subconjunto finito de los números enteros. Estos son números completos, por lo cual no poseen componentes fraccionarios o decimales, estos pueden ser negativos o positivos.

Reales: El tipo real está en subconjunto de los números reales, estos siempre poseen un punto decimal, y pueden ser positivos o negativos. estos siempre constan de una parte entero y una parte decimal.

Datos de tipo lógicos

Los datos de tipo lógico o también llamados booleanos como solo pueden contener verdadero y falso se utilizan para representar alternativas si o no, a determinadas condiciones

Datos de tipo alfanuméricos

Los caracteres para las computadoras no son estándar, pero por lo general se reconocen los caracteres alfabéticos y numéricos:

- Caracteres alfabéticos
- Caracteres numéricos
- Caracteres especiales

Las cadenas de caracteres son una sucesión de caracteres que se encuentran delimitados por comillas (apóstrofo) o dobles comillas “, dependiendo del tipo de lenguaje de programación. **La longitud** de una cadena es el número de caracteres que existe dentro de la cadena.

Asignación

Esta es la operación que le permite darle a una variable un determinado valor. Entre estos valores se puede asignar:

- Un valor constante
- El valor de otra variable
- El valor de una constante
- El resultado de evaluar una expresión.

Al asignar el valor a una variable, si esta ya posee una variable se pierde y se reemplaza por el mismo valor, esto quiere decir que:

```
a = 2
print(a) → Mostrará 2
a = 4
print(a) → Mostrará 4
```

En otros lenguajes como Java, c, c++, etc., Normalmente los valores asignados deben ser mismo tipo que la variable, pero en python a las variables se le pueden asignar otro tipo de variables sin problemas.

Operaciones de entrada y salida

Las operaciones de entrada y salida, nos permiten intercambiar información con el medio externo.

Entrada: Se se le asigna a una variable un valor dado desde el exterior, esto puede ser a través un teclado, o recibir variables desde otra función, etc

Salida: Se le transfiere el valor de una variable a un dispositivo de salida esto se puede ver a través de la pantalla, o enviar el valor a otro programa.

Constantes y variables

Durante la ejecución de los programas existen valores que no deben cambiarse durante la ejecución del programa, estos valores se llaman constante, mientras tanto existen otros valores que cambian durante la ejecución estos valores se les llama variable.

Constante: Es un dato que permanece sin cambios durante todo el algoritmo o durante la ejecución del programa.

Variable: Es un objeto o tipo de datos cuyo valor puede cambiar durante el desarrollo del algoritmo o ejecución del programa. Para poder reconocer una variable en la memoria de la computadora, es necesario darle un nombre con el cual podamos identificarla dentro de un algoritmo.

Clasificación por contenido

Las variables también poseen otro tipo de clasificación se pueden clasificar por tipo de uso:

Variable Numéricas: Son variables en la que almacenan valores numéricos, positivos o negativos. En ellas se encuentran los tipos de datos como short, int, long, long long, float, double, long double.

pi=3.1416 valor=12000

Variables lógicas: Son variables que pueden tener solo dos tipos de valores, verdadero o falso.

Variables Alfanuméricas: Las variables alfanuméricas se forman por caracteres alfanuméricos.

Nombre="Juan" Direccion="Calle falsa 123"

Clasificación de variables por uso

Las variables también poseen otro tipo de clasificación se pueden clasificar por tipo de uso:

Variables de trabajo: Este tipo de variable recibe el resultado de una operación matemática completa y se usan normalmente dentro de un programa.

`area = a * b`

Contadores: Se utilizan para llevar un control del número de ocasiones en que se realiza una operación o se cumple una condición. Normalmente el incremento del contador es uno en uno.

Acumuladores: Forma que toma una variable y que sirve para llevar una suma acumulativa de una serie de valores que se van leyendo o calculando progresivamente.

Nombrando y usando variables

Restricciones y recomendaciones para nombrar una variable:

- Los nombres de las variables solo pueden contener letras, números y guiones bajos.
- Pueden comenzar con una letra o guión bajo, pero no con un número.
- No se permiten espacios en los nombre de las variables, pero se pueden usar guiones bajos
- No se pueden utilizar palabras que python haya reservado para un programa en particular
- Se recomienda utilizar nombre breves pero descriptivos.
- Tener cuidado al utilizar l minúscula y la letra O mayúscula ya que se podrían confundir con 1 y 0.
- Un ejemplo correcto: `variable_1`, un ejemplo erróneo: `1_variable`.
- Un ejemplo correcto: `costo_total`, un ejemplo erróneo: `costo total`.
- Al nombrar es mejor `nombre_estudiante` que nombrarlo como `n_e`.

Operadores y operandos

Los operadores: Los operadores permiten la manipulación de valores de una o más variables.

Estos se clasifican en:

- Aritméticos
- Relacionales
- Lógicos

Operadores aritméticos

+	Suma	$2+3$	5
-	Resta	$2-3$	-1
*	Multiplicación	$2*3$	6
**	Exponenciación	$2**3$	8
/	división	$2/3$	0.6666
//	división entera	$2//3$	0
%	módulo	$2\%3$	2

Operadores aritméticos

Prioridades de los operadores:

- Todas las expresiones que están entre paréntesis se evalúan primero. En el caso de tener paréntesis anidados se evalúan desde dentro a fuera, el paréntesis mas interno se evalúa primero.
- Dentro de la expresión los operadores evalúan con el siguiente orden:
 - Exponenciación.
 - Multiplicación, división, módulo.
 - Suma y resta.
- En el caso que los operadores tengan el mismo nivel de prioridad se evalúan de izquierda a derecha.

Operadores relacionales

Para los ejemplos se asumen que $a=2$, $b=10$, $c=12$

$>$	Mayor que	$a+b > c$	Falso
$<$	Menor que	$a-b < c$	Verdadero
\geq	Mayor o igual que	$a+b \geq c$	Verdadero
\leq	Menor o igual que	$c-b \leq a$	Verdadero
\neq	Diferente	$a+b \neq c$	Falso
$==$	Igual	$b==c$	Falso

Operadores lógicos

Las operaciones lógicas están compuestas por:

- Conjunción lógica and
- Disyunción lógica or
- Negación lógica not

Operadores lógicos

p	q	p and q	p or q	not p
Verdadero	Verdadero	Verdadero	Verdadero	Falso
Verdadero	Falso	Falso	Verdadero	
Falso	Verdadero	Falso	Verdadero	Verdadero
Falso	Falso	Falso	Falso	

Estructuras de control

Las estructuras de control las podemos dividir en:

- Secuencial, una acción sigue a otra sin romper la secuencia
- Condicional se realiza una acción u otra dependiendo del resultado de la evaluación de una expresión lógica
- Repetitiva, iterativa o en bucle se repite un conjunto de acciones 0 o más veces.

Comentarios y documentación

La capacidad de comentar el código fuente se encuentra en todos los lenguajes de programación, un comentario es una línea de código que no es ejecutable, esto quiere decir que el compilador o intérprete no la tomara como una línea de código.

Los comentarios sirven para dar explicaciones sobre el programa, detallando aspectos como significados de variables, instrucciones de uso de las funciones, esto sirve principalmente cuando trabajas sobre proyectos con gran cantidad de código.

En python se puede comentar de dos formas:

- La primera es con el símbolo `#` el cual declara que desde el `#` en adelante será nuestro comentario, pero solo dentro de esa línea
- La otra manera es escribiendo triple comilla `'` al principio y al final del comentario, este permite tomar más de una sola línea.

Comentarios y documentación

```
# Comentario de una sola linea
print("hola mundo") # Comentario

'''
Este es un comentario
con mas de una linea
'''

'''
Tambien se puede comentar
en mas linea de esta manera
'''
```

Indentación en python.

Los diseñadores del lenguaje python trataron de evitar la necesidad de usar caracteres como llaves “{” y “}” o palabras claves como “Begin” y “end”, simultáneamente que el código fuera escrito de manera más ordenada y menos elementos y por tanto más fácil de leer para un humano.

La indentación en otros lenguajes:

C	Matlab	VBA(excel)
<pre>for (pos=0; pos<n; pos++) { printf("pos = %d\n", pos); }</pre>	<pre>for pos=0:n-1 fprintf('pos = %d\n', pos) end</pre>	<pre>For pos = 0 to n-1 Cells(pos, 1).value = pos Next</pre>
Python		
<pre>for pos in range(n): print('pos =', pos)</pre>		

Entrada y salida en python

Para la entrada y salida de datos las funciones más básicas dentro de python `input()` y `print()`, La función `input()`, permite obtener texto escrito por el teclado. Al llegar a la función, el programa se detiene esperando que se escriba algo y se pulse la tecla Intro. En cambio la función `print()`, permite la salida a través de la consola, si se quiere mostrar un texto o una variable se utiliza la función `print()`.

```
a = input()
print(a)
print("Hola mundo")
```

Tarea N°1

Para los problemas presentados genere el paso de análisis donde incluye variables entrada, salida y proceso, luego genere el pseudocódigo del problema. Puede guiarse con la presentación de la semana 1, la presentación de la semana 2. Luego realice la codificación en python de 3 problemas entre el 1 al 5.

- 1.- Genere un programa que recibe una cantidad de minutos, y me muestra la cantidad de horas y minutos que corresponde.
- 2.-Calcular el perímetro y área de un rectángulo dada su base y su altura.
- 3.-Dados los catetos de un triángulo rectángulo, calcular su hipotenusa.
- 4.-Dados dos números, mostrar la suma, resta, división y multiplicación de ambos.
- 5.- Se reciben 2 números distintos, se debe mostrar la distancia entre ellos.

Tarea N°1

6.- Genere el pseudocódigo del siguiente diagrama de flujo

