

Operaciones básicas con datos y variables y cadena de caracteres

Sitio: Agencia de Habilidades para el Futuro

Curso: Técnicas de Programación 1° D

Libro: Operaciones básicas con datos y variables y cadena de caracteres

Imprimido por: RODRIGO PINTO

Día: martes, 26 de noviembre de 2024, 07:19

Tabla de contenidos

1. Operaciones básicas con datos y variables.

1.1. Expresiones aritméticas y lógicas

1.2. Tablas de verdad para las expresiones lógicas

2. Cadenas de caracteres

2.1. PSeInt para nuestro análisis

1. Operaciones básicas con datos y variables.

¡Antes de empezar!



Vamos a recordar los principios básicos de la representación gráfica de algoritmos a través de diagramas de flujo y las bases de PSeInt, un programa que te permitirá desarrollar algoritmos y diagramas.

Diagramas de Flujo



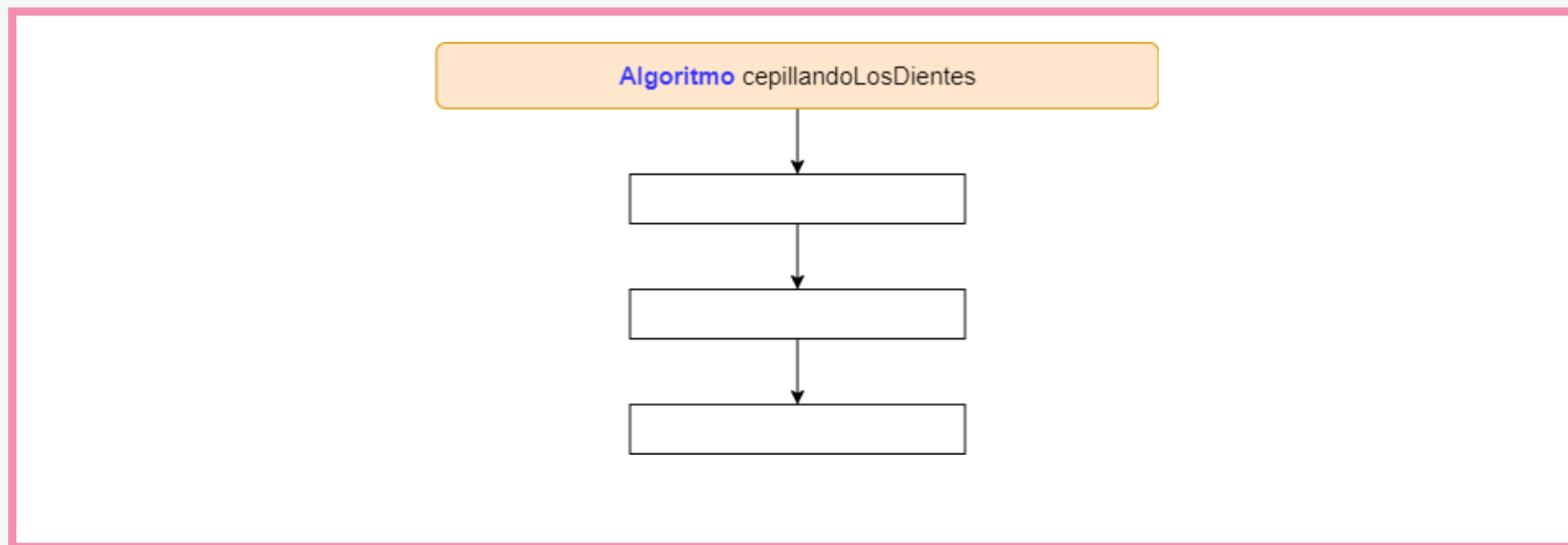
¿Escuchaste hablar alguna vez de diagramas de flujo? ¡Seguro que sí!, cuando realizaste el curso de Pensamiento Computacional.

En nuestra vida cotidiana utilizamos diferentes formas de representación, que funcionan como modelos ideales: simplifican y muestran rasgos esenciales de un elemento o proceso. El modo de representar una serie de pasos es a partir de este tipo de diagramas.

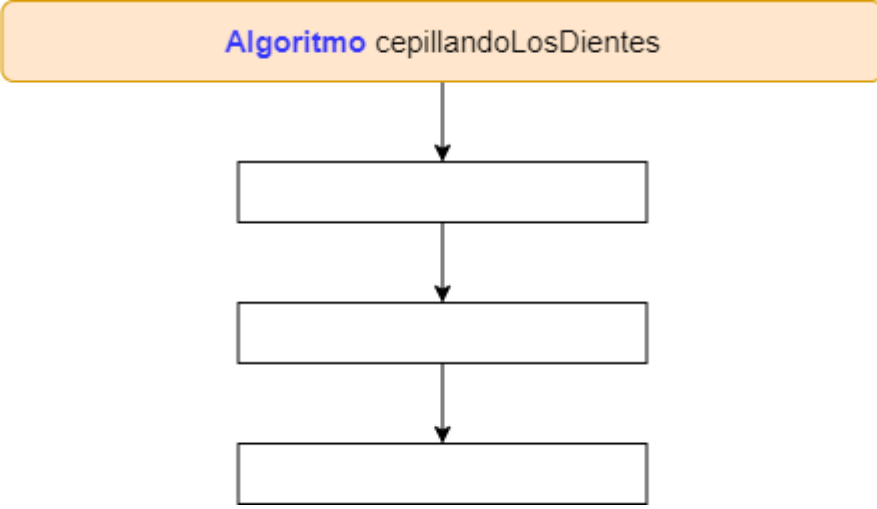
Es la representación gráfica de un algoritmo o proceso. Se utiliza en disciplinas como programación, economía, procesos industriales y psicología cognitiva. Estos diagramas utilizan símbolos con significados definidos que representan los pasos del algoritmo, y representan el flujo de ejecución mediante flechas que conectan los puntos de inicio y de fin del proceso.

El diagrama de flujo no lleva números que indiquen el orden de los pasos. Este orden se estructura a partir de la secuencialidad del proceso de arriba hacia abajo.

Un diagrama de flujo puede ser simple o muy complejo. Por eso, no alcanza solamente con la organización de la información de arriba hacia abajo. Hay símbolos que nos ayudan a determinar el tipo de proceso que estamos realizando en ese momento en el marco de nuestro algoritmo. Te mostramos los símbolos del diagrama que estamos utilizando... hay otros que ya vas a conocer más adelante y que no son necesarios para este algoritmo.



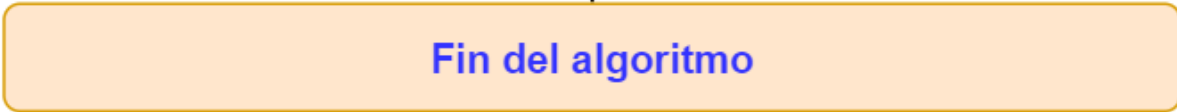
Siempre el diagrama **comienza con un símbolo de caja de bordes redondeados** que contiene el nombre de nuestro algoritmo. Esto es importante, porque marca el comienzo del proceso.



Algoritmo cepillandoLosDientes

En este símbolo de recuadro se escribe un paso (nunca más de uno) del proceso del algoritmo que implica una acción; por ejemplo, tapar la pasta: algo que hay que hacer. Siempre tiene forma rectangular, para diferenciarse de otro tipo de pasos.

Como ves, cada recuadro que contiene un paso está relacionado con los otros pasos a través de un símbolo flecha que indica el próximo paso a ejecutarse en nuestro proceso.



Fin del algoritmo

Cuando llegamos al último paso y termina el proceso del algoritmo, se indica con el símbolo de caja de bordes redondeados con las palabras Fin del algoritmo.

PSint



Accede al enlace de un tutorial básico de PSint haciendo clic en el botón:

[Tutorial](#)

Operaciones básicas con datos y variables.

Las variables son porciones de memoria a las que definimos con un determinado tipo de dato según el valor que necesitemos que almacene y poder guardar en ellas datos o información.

Sobre las variables, o mejor dicho sobre los datos de la variable, podemos realizar distintas operaciones dependiendo su tipo.

Las operaciones se clasifican en:

- Aritmética
- Lógicas
- Cadena de caracteres

1.1. Expresiones aritméticas y lógicas

Expresiones aritméticas

Se componen de operandos y operadores de tipo aritmético, siendo los más habituales suma (+), resta (-), multiplicación (*), división (/) y resto de la división (%).



Por ejemplo, las expresiones al igual que en matemática se construyen a partir de operadores y operandos. La mayoría de los operadores requiere dos operandos, aunque existen otros operadores que aplican a un solo operando.

Operador	Operación realizada	Ejemplo	Resultado
+	Suma	6+4	10
-	Sustracción	12-6	6
*	Multiplicación	3*4	12
/	División	25/3	8.3333333333
%	Módulo (resto de la división entera)	25 % 3	1

Expresiones lógicas

Son aquellas que tienen como resultado verdadero o falso. Los operadores que permiten construirlas son los relacionales y los booleanos. La simbología utilizada es la referente al lenguaje PSeInt. Los operadores relacionales se utilizan para comparar operando entre sí (comparación de caracteres, números, fecha, etc.).

OPERADOR	DESCRIPCIÓN	EJEMPLO	RESULTADO
==	Igual a. Comprueba si dos operandos son iguales.	(3 == 3)	Es True
		(num == 3)	Es True si num almacena el valor de 3.
!=	Diferente a. Comprueba si dos operandos no son iguales.	(3 != 3)	Es False
		(num != (n * 3))	Es False si la variable num es diferente el valor al cálculo de n * 3
>	Mayor que. Comprueba si el primer operando es mayor al segundo operando.	(3 > 5)	Es False
		(valor > n)	Es False si la variable n tiene un valor mayor a valor
<	Menor que. Comprueba si el primer operando es menor que el segundo operando.	(7 < 8)	Es True
		((num + x) < n)	Es True si la variable n tiene un valor mayor a (num + x)
>=	Mayor o igual. Comprueba si el primer operando es mayor o igual que el segundo operando.	(9 >= 16)	Es False
		(num >= (n - 5))	Es True si la variable num tiene un valor mayor a n - 5
<=	Menor o igual. Comprueba si el primer operando es menor o igual que el segundo operando.	(21 <= 21)	Es True
		(num <= n)	Es True si la variable num tiene un valor menor o igual a n

OPERADOR	DESCRIPCIÓN	EJEMPLO	RESULTADO
&&	AND (Y lógico)	((3 == 3) && (5 == 5))	Es True, Verdadero y verdadero da verdadero
		((num > 3) && (num < 5))	Es True si num almacena un valor igual a 4
		((num > 0) && (num < 1000))	Es True si num almacena un valor mayor a 0 y menor que 1000
	OR (O lógico)	((3 == 3) (8 != 6))	Es True, porque las condiciones son verdaderas
		((num > 3) (num < 5))	Si num = 5, El resultado es True, porque num es mayor a tres y no es menor a 5
		((num * 2 > 0) (num / 4 < 45))	Si num = 3, El resultado es True, porque num es mayor a tres y no es menor a 5
!	NOT lógico) (NO)	!(3 == 3)	El resultado es False, porque estamos negando el condicional.
		!(num * 2 > 0)	Si num = 5, el condicional es True y como lo estamos negando, el resultado final es False.
		!(x * 2 > 0)	Si x = -2, el condicional es False y como lo estamos negando, el resultado final es True.

A	B	A && B
V	V	V
V	F	F
F	V	F
F	F	F

A	B	A B
V	V	V
V	F	V
F	V	V
F	F	F

A	!A
V	F
F	V

1.2. Tablas de verdad para las expresiones lógicas

Los operadores lógicos

Permiten agrupar expresiones lógicas. Se usan para combinar dos valores booleanos y devolver un resultado verdadero o falso. Se denominan operadores booleanos porque hacen uso de los principios del álgebra de Boole.

Para la conjunción AND $\rightarrow \&\& \rightarrow Y \rightarrow ^$

- Podemos decir que la disyunción será verdadera si al menos uno de los operandos es verdadero.

Los operadores básicos que utilizaremos son: and (y), or (o) y ! (no).

En las tablas siguientes se explica cada uno.

El operador and (y) tiene resultado cierto cuando las condiciones sobre las que se aplica son ciertas.

Por ejemplo,

si para ir al cine las condiciones son A) tener tiempo y B) tener dinero. Sólo se irá al cine cuando ocurran las 2 condiciones simultáneamente.

VARIABLES		EXPRESION
A	B	A & B
Falso	Falso	Falso
Falso	Verdadero	Falso
Verdadero	Falso	Falso
Verdadero	Verdadero	Verdadero

Podemos definir que la conjunción será verdadera si y solo si todos los operandos son verdaderos.

El operador or (o) tiene resultado cierto cuando alguna de las condiciones son ciertas.

Por ejemplo,

podemos enterarnos de una noticia a través de A) la radio B) la TV. Basta con escuchar la radio, ver la TV o ambas cosas para enterarnos.

Para la disyunción OR → || -> 0 -> v

VARIABLES		EXPRESION
A	B	A B
Falso	Falso	Falso
Falso	Verdadero	Verdadero
Verdadero	Falso	Verdadero
Verdadero	Verdadero	Verdadero

Para la negación NOT → !

La operación NOT negará el valor del operando. Siempre aplicará a un solo operando, pudiendo ser este una expresión booleana.

Por ejemplo, tenemos dos variables y le asignamos valores a cada una de ellas:

```
varNro1 = 100, varNro2 = 200
```

- Si (varNro1 == varNro2)→ Falso no son iguales
Si (!(varNro1==varNro2))→ Verdadero
Esto es porque varNro1 no es igual a varNro2, por lo que la comparación es falsa.

Entonces reemplazando → ! (Falso) voy a negar lo falso, quedando como resultado verdadero.

Ejemplo: Si hace calor y tengo plata, voy a ir a la pileta.

Condiciones
Conclusión

Condición 1	Condición 2	Conclusión
V	V	V
F	V	F
V	F	F
F	F	F

Ejemplo: Si no llueve, voy a ir al parque.

Condición
Conclusión

Condición	!Condición	Conclusión
V	F	F
F	V	V

Ejemplo: Si me invitan o tengo plata, voy a ir al cine.

Condiciones
Conclusión

Condición 1	Condición 2	Conclusión
V	V	V
F	V	V
V	F	V
F	F	F

Precedencia

Así como en matemáticas se sabe que las operaciones de multiplicar y dividir se ejecutan antes que la suma y la resta, la precedencia de los operadores lógicos es: **not**, **and**, **or**. Primero se evaluará el not, luego el and y finalmente el or. Se recomienda, a los efectos de clarificar, utilizar paréntesis.

Ejemplos de expresiones lógicas:

Sean:

$x = 3$

$y = 4$

$z = 2$

$f = \text{false}$

¿Cuánto vale cada una de las siguientes expresiones?

a) $(x > z) \ \&\& \ (y > z)$

b) $(x + y / 2) \leq 3.5$

c) $!f$

d) $!f \ || \ ((x < z) \ \&\& \ (z \geq 3 * y))$

e) $f \ \&\& \ (x > z + y)$

Respuestas:

- a) Verdadero
- b) Falso, 5 no es menor que 3.5
- c) Verdadero
- d) Verdadero. Ya al evaluar "not f" se sabe que la expresión entera es verdadera pues es un "or".
- e) Falso. El primer término es falso, por lo cual, dado que es un "and", la expresión entera es falsa.

2. Cadenas de caracteres

Las cadenas de caracteres se comparan comenzando por el primer carácter a la izquierda hasta encontrar la primera diferencia, resultado la comparación de acuerdo al número de orden del carácter en que difieren.

`'a' < 'b' → Verdadero`
`'Emma' > 'Eva' → Falso`
`'mesa' < 'mesada' → Verdadero`

La operación exclusiva que puede realizarse con estos tipos de datos es la concatenación que se representará con el símbolo +, en el siguiente ejemplo utilizaremos dos variables y compondremos el valor de una tercera concatenando los valores de las otras dos.

Ejemplo:

`sNombre = 'Juan'`

`sApellido = 'Martinez'`

`sNombreYApellido ← sNombre + ',' + sApellido`

`sNombreYApellido → 'Juan,Martinez'`

Agregando Instrucciones de Entrada y Salida

Consigna:



Una consultora en sistemas paga a sus programadores \$2850.50 la hora. Dado el nombre y el apellido de un/a programador/a (se solicitan los datos por separado) y la cantidad de horas trabajadas, informar el nombre completo (apellido, nombre) y el importe total a abonarle.

Veamos cómo encarar una posible solución.

Problemática:

-Debo pagarle al empleado y no sé cuánto abonarle.

Datos de entrada:

- Nombre del programador
- Apellido del programador
- Cantidad de horas trabajadas
- Valor por hora

Datos de salida:

- Nombre y apellido
- Importe total a abonar

Antes de abordar cualquier solución debemos comprender bien la problemática planteada y los datos de entrada, es decir, con qué datos contamos para poder obtener los datos de salida o información que me solicitan.

Una vez obtenidos estos datos, debemos analizar si con estos datos es suficiente para llevar a cabo el desarrollo. Si no lo es, entonces debemos plantear una o varias hipótesis para completar el enunciado.

En un pseudocódigo de análisis básico nos quedaría de la siguiente manera:

1. Solicitar el nombre del programador.
2. Solicitar el apellido del programador.
3. Solicitar la cantidad de horas trabajadas.
4. Multiplicar la cantidad de horas trabajadas por 2850.50.
5. Armar el nombre completo según formato solicitado.
6. Mostrar por pantalla el nombre completo.
7. Mostrar por pantalla el importe a abonar.

2.1. PSeInt para nuestro análisis

Pasemos a PSeInt nuestro análisis y veamos que sucede línea por línea:



Aclaración: la doble barra // indica que ese renglón es para poder realizar comentarios y no es parte de la resolución del problema.

Algoritmo honorarios:

```
//Definimos las variables de nuestro programa cada una.  
// con su tipo de dato correspondiente.  
Definir nombre, apellido Como Caracter.  
Definir cantidadHorasTrabajadas Como Entero.  
Definir valorHora Como Real.  
Definir nombreCompleto Como Caracter.  
Definir importeACobrar Como Real.  
  
// La sentencia o comando Escribir muestra por pantalla el mensaje que  
// le indiquemos  
Escribir "Ingresa el nombre del programador: "  
  
// La sentencia o comando Leer lee por teclado lo que el usuario ingrese y lo  
// almacena en la variable indicada  
Leer nombre
```

Escribir "Ingresa el apellido del programador: "

Leer apellido

Escribir "Ingresa la cantidad de horas trabajadas: "

Leer cantidadHorasTrabajadas

// a la variable importeACobrar le asignamos (con el símbolo =) el valor contenido en

// la variable cantidadHorasTrabajadas multiplicado por 2850.50

importeACobrar = cantidadHorasTrabajadas * 2850.50

nombreCompleto = apellido + " , " + nombre

Escribir nombreCompleto

//En la siguiente sentencia concatenamos un valor literal con el valor de una variable

Escribir "Importe a cobrar: \$", importeACobrar

FinAlgoritmo

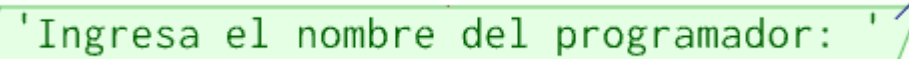
Algoritmo honorarios**Definir** nombre, apellido **Como Caracter****Definir** cantidadHorasTrabajadas **Como Entero****Definir** valorHora **Como Real****Definir** nombreCompleto **Como Caracter****Definir** importeACobrar **Como Real****Escribir** "Ingresa el nombre del programador: "**Leer** nombre**Escribir** "Ingresa el apellido del programador: "**Leer** apellido**Escribir** "Ingresa la cantidad de horas trabajadas: "**Leer** cantidadHorasTrabajadas

importeACobrar = cantidadHorasTrabajadas * 2850.50

nombreCompleto = apellido + ", " + nombre

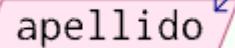
Escribir nombreCompleto**Escribir** "Importe a cobrar: \$", importeACobrar**FinAlgoritmo**

En diagrama de flujo las salidas por pantalla se representan de la siguiente manera:



```
'Ingresar el nombre del programador: '
```

Los ingresos desde el teclado se representan de la siguiente manera:



```
apellido
```

Para asignarle un dato a una variable, se debe poner a la izquierda de la sentencia el nombre de la variable que recibirá el valor, seguido una flechita apuntándole (<-) o bien un signo igual (=) y el valor a asignarle.

Este valor a asignar puede ser:

- Un valor propiamente dicho.
- Una variable.
- El resultado de una operación aritmética o lógica.

Por ejemplo en nuestro caso:

La variable importeACobrar recibe el resultado del producto del valor de la variable cantidadHorasTrabajadas por el valor 2850.50.

```
importeACobrar ← cantidadHorasTrabajadas*2850.50
```

Nuestro diagrama de flujo quedará de la siguiente manera:

