Escuela Normal Superior N°40 “Mariano Moreno”

Datos y Algoritmos para la Programación de Computadoras

Cuadernillo Propedéutico 2017 Nivel Superior

INDICE

Datos

Identificación de datos

MARCO TEÓRICO: Los **datos** son todos aquellos elementos de **información** presentes en cualquier situación a resolver. Sin la existencia de ellos, ni siquiera podría plantearse un problema, o bien, el mismo no tendría una solución. En la mayoría de los casos, la información con la que contamos es excesiva y mucha de ella innecesaria, de ahí que también es importante saber desechar aquella que no nos es útil y quedarnos sólo con la que realmente vamos a utilizar. Se considera información útil sólo a los datos necesarios para resolver lo que se nos pida en el problema o ejercicio planteado.

Ejemplo

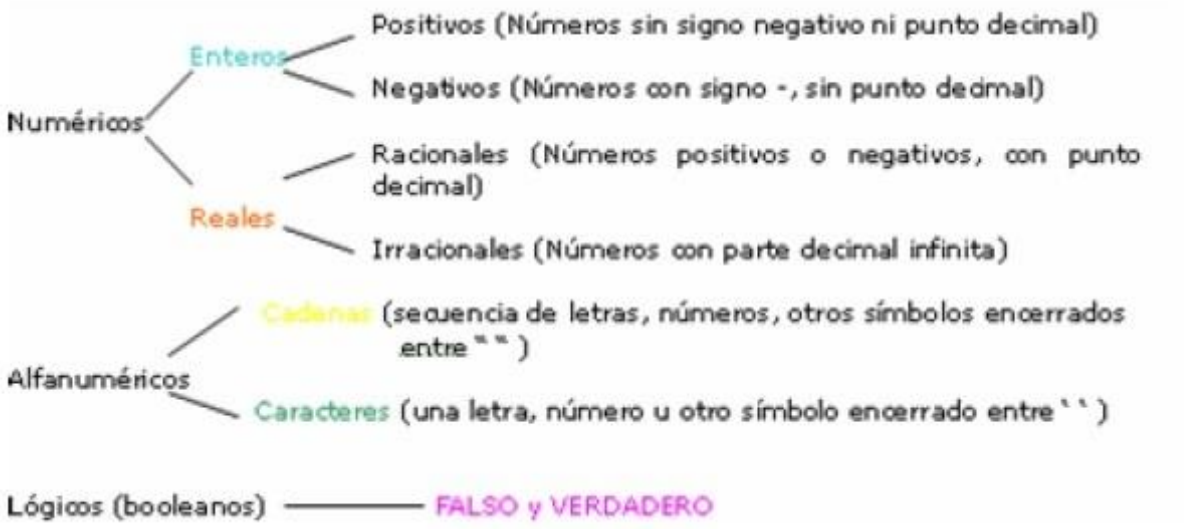
En el siguiente caso se identifican los datos importantes para su solución.

Juan Pérez, casado, vive en la calle Norte 3245, tiene que pagar sus cuentas mensuales de Alquiler, Luz y Teléfono (su teléfono es el 2345567). El monto de cada recibo es de $3520,50 $998,70 y $548,99 respectivamente, y él percibe un sueldo mensual de $12380,00, pero en este momento solo dispone de $6000.00 para hacer sus pagos. ¿Cuánto dinero tendrá después de hacer sus pagos para gastarlos en familia?

Datos	¿El dato es relevante para resolver el problema?	
	SI	NO
Nombre de la persona (Juan Pérez)		X
Estado Civil (Casado)		X
Domicilio (Calle Norte 3245)		X
Teléfono (2345567)		X
Monto Recibo Alquiler	X	
Monto Recibo de Luz	X	
Monto Recibo Teléfono	X	
Sueldo Mensual		X
Dinero Disponible	X	
Dinero Disponible posterior a los pagos	X	

Tipos de datos

MARCO TEÓRICO: Otro reto al que nos enfrentamos cuando analizamos un problema, es el de saber **clasificar** la información, esta clasificación se hace de acuerdo al tipo de datos que maneja, pues de ello dependerán las operaciones que sobre tales datos podamos realizar. Por ejemplo, no podemos considerar que el nombre de una persona sea un número o que la altura de un rectángulo sea un color, ni siquiera que el número de páginas que tiene un libro sea una fracción. Así, una vez que logremos identificar los datos necesarios dentro de un problema, vamos a asignarles un **tipo**. Los tipos que debemos considerar para nuestros datos son: los **numéricos**, (aquellos que tienen que ver con valores medibles o cantidades), **alfanuméricos** (formados por símbolos o letras) y **lógicos** (los que se refieren a valores que se pueden calificar como SI o No o como FALSO y VERDADERO), Consideremos la existencia de los siguientes tipos:



Es importante señalar que para que un dato sea usado de manera correcta por una computadora debe tener una "representación especial ", observe la siguiente tabla:

Tipo de Dato	Dato (Valor)	Representación en Computadora
Cadena de caracteres	Rodrigo González	"Rodrigo González"
caracter	@	'@'
Número entero	245	245
Número real	45309,87	45309,87
Lógico (booleano)	Si - NO	Verdadero (true) – falso(false)

Variables

MARCO TEÓRICO: Una variable es un elemento que permite el almacenamiento de un dato, para ser utilizado en representación del mismo. Las variables, como el nombre lo indica, pueden tomar distintos valores es decir, cambian su valor cuantas veces sea necesario y en diferentes situaciones, así, pueden utilizarse para expresar fórmulas que una vez evaluadas arrojen algún valor que dependa de los datos sobre los que se aplica (el resultado de la fórmula, dependerá de sus variables). En el ámbito de la programación, las variables se crean en la memoria de la computadora.

Las variables se definen a través de 3 atributos: **Identificador** o nombre de la variable, **Tipo de Dato** o rango de valores que puede almacenar, y **Valor** que en ese momento se encuentra almacenado en ella.

Por ejemplo, si nos encontramos ante el siguiente caso:

Se desea emitir un mensaje de alerta cada vez que una persona menor de edad se identifique mediante el uso de un lector de huella digital.

Entonces el dato importante para que se emita la alerta sería la edad de la persona, y podría proponerse una variable para manejar este dato cuyo identificador se llamaría Edad, y el Tipo de Dato de la variable sería entero, lo que significa que solo podría almacenar valores numéricos enteros.

Dato extraído	Identificador	Tipo	Valor
Edad de la Persona	Edad	Numérico Entero	Valores numéricos enteros

Identificadores

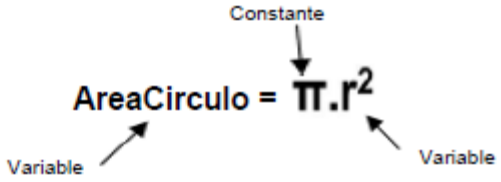
MARCO TEÓRICO: Los identificadores son los nombres simbólicos que se asignan a los diferentes objetos que se utilizan en los programas de cómputo, estos nombres son introducidos por el programador para hacer referencia a una variable, constante entre otros y deben obedecer ciertas reglas al momento de formarse, además de ellas, es conveniente utilizar nombres apropiados con el uso de la variable, esto es, que exista relación entre el dato y su nombre, por ejemplo, si se necesita manejar una variable para almacenar una calificación, un identificador apropiado será ***calificacion*** y en lugar de utilizar ***impuesto***, pues este último hace referencia a otra cosa (observe que el identificador *calificacion* no lleva acento es una de las restricciones que se deben cumplir.)

Los identificadores de las variables son cadenas de caracteres que deben seguir las convenciones:

- Empezar con una letra o el símbolo de guión bajo (_).
- No deben incluir operadores (+ - * / % & =).
- No deben incluir signos de puntuación ni comillas o apóstrofes (. , ; : ¿ ¡ “ ‘).
- Pueden contener números combinados con otros caracteres (no exclusivamente números).
- No deben llevar espacios en blanco.

Constantes

MARCO TEÓRICO: Una constante es un dato de cualquier tipo: numérico, alfanumérico o lógico que a diferencia de una variable, su valor no cambia ante ninguna variante de la solución del problema. Por ejemplo: pi = 3.1416, es un dato útil en el cálculo del área de un círculo, que, independientemente del valor de un radio, mantiene siempre el mismo valor. Las constantes al igual que las variables tienen **identificador** o nombre, **tipo de dato** y **valor** que almacena.



En la expresión anterior, que representa la fórmula del área de un círculo, se puede identificar a la Constante pi=3.1416, que es un valor que se va a mantener fijo, y a la variable r que va a almacenar el radio del círculo y variable AreaCirculo que va a contener el resultado del cálculo respectivamente.

Operadores y Expresiones

Todos los símbolos que representan enlaces entre cada uno de los argumentos u operandos que intervienen en una operación se llaman operadores, y se utilizan para construir expresiones.

Las expresiones son combinaciones de operadores y operandos, estos últimos pueden ser variables o constantes. En función del tipo de operadores, las expresiones se clasifican en aritméticas (Ej. a + (b * 3) / c), relacionales (Ej. x>=20), y lógicas (Ej. Not(x>5)).

Los operadores pueden ser:

ARITMÉTICOS

Son operadores utilizados para realizar cálculos matemáticos. Para operar con números se utilizan operadores aritméticos, que junto con las variables numéricas forman expresiones aritméticas.

- + Suma
- - Resta
- * Multiplicación
- ^ Potenciación
- / División real
- \ División entera
- MOD Resto de la división entera

Ejemplo de expresión aritmética: a + (b * 3) / c

Donde +, * y / son operadores aritméticos; a, b y c son variables, y 3 es un valor constante.

Nota: Es importante enfatizar que el único operador reconocido para representar la multiplicación es el asterisco (*).

Una vez comprendidas las operaciones aritméticas, debemos considerar una operación que muchas veces no es tomada en cuenta pero su importancia es fundamental, esta operación es la **ASIGNACIÓN.**

La función de la asignación consiste en poner a una variable algún valor, el cual puede ser obtenido de un dato constante, por ejemplo 3, "Juan", falso, etc. o de una expresión por ejemplo: 2+4, a*2, etc.
El operador que vamos a utilizar para representar la asignación es una flecha ← que será utilizado de derecha a izquierda, es decir, si queremos ponerle a la variable nombre el valor "Sofía Juárez", la expresión quedará de la siguiente manera:

Nombre ← "Sofía Juárez"

Nota: En algunos casos, la computadora también reconoce el símbolo = para denotar la operación de asignación.

RELACIONALES O DE COMPARACIÓN

Son operadores utilizados para efectuar comparaciones; permiten formar expresiones booleanas, es decir, expresiones que al ser evaluadas generan un valor lógico Verdadero o Falso.

- < Menor que
- = Igual
- > Mayor que
- <= Menor o igual que
- >= Mayor o igual que
- <> Distinto de

Ejemplo: (Num > 5)

Donde > es el operador relacional Mayor que; Num es una variable, y 5 es un valor constante. La expresión booleana o lógica del ejemplo, va a devolver un valor Verdadero o Falso en función del valor almacenado en la variable Num.

LÓGICOS O BOOLEANOS

Son operadores usados para realizar operaciones lógicas.

- OR Suma lógica
- AND Producto lógico
- NOT Negación

OR (O)

Es un operador binario, es decir, afecta a dos operandos. La expresión que forma es verdadera cuando al menos uno de sus operandos es verdadero. Da como resultado falso si todos los operandos tienen un valor falso. Es el operador lógico de disyunción.

Ejemplo: *(5>10)* ***Or*** *(3<2) Devuelve Falso*

AND (Y)

Es un operador binario. La expresión formada es verdadera cuando ambos operandos son verdaderos. Da como resultado falso, si al menos uno de los operandos tiene un valor falso. Es el operador lógico de conjunción.

Ejemplo: *(3<5)* ***And*** *(30>20) Devuelve Verdadero*

NOT (NO)

Es un operador unario, es decir, sólo afecta a un operando. Afecta a la expresión cambiando su estado lógico: si era verdadero lo transforma en falso, y viceversa.

Ejemplo: ***Not****(18>5) Devuelve Falso*

ALFANUMÉRICOS O DE CONCATENACIÓN

Son operadores utilizados para combinar cadenas de caracteres, para unir datos alfanuméricos.

- + ; & : Concatenación

La concatenación consiste en unir expresiones alfanuméricas como si fueran eslabones de una cadena.

TABLA RESUMEN DE OPERADORES

Operadores Aritméticos	Operadores de Comparación	Operadores de Concatenación	Operadores Lógicos
Potencia ^	< (Menor que)	&	And
Multiplicación *	<= (Menor o igual que)	+	Or
División /	> (Mayor que)		Not
Resto División Mod	>= (Mayor o igual que)		
Suma +	= (Igual a)		
Resta -	< > (Distinto de)		

ORDEN DE EVALUACION

La prioridad a la hora de evaluar los operadores en cualquier expresión es:

- Paréntesis (empezando por los más internos)
- Potencias
- Productos y divisiones
- Sumas y restas
- Concatenación
- Relacionales
- Lógicos

PARÉNTESIS

Los paréntesis se utilizan para anidar expresiones.

- ( ) Anidar expresiones

Algoritmos

MARCO TEÓRICO: Un algoritmo es un procedimiento a seguir, para resolver un problema en términos de: las acciones por realizar y el orden en que dichas acciones deben llevarse a cabo.

Un algoritmo nace en respuesta a la aparición de un determinado problema. Un algoritmo está compuesto de una serie finita de pasos que convergen en la solución de un problema, pero además estos pasos tienen un orden específico.

Entenderemos como problema a cualquier acción o evento que necesite cierto grado de análisis, desde la simpleza de cepillarse los dientes hasta la complejidad del ensamblado de un automóvil. En general, cualquier problema puede ser solucionado utilizando un algoritmo, en este sentido podemos utilizar los algoritmos para resolver problemas de cómputo.

Un algoritmo para un programador es una herramienta que le permite resaltar los aspectos más importantes de una situación y descartar los menos relevantes. Todo problema de cómputo se puede resolver ejecutando una serie de acciones en un orden específico.

Por ejemplo, considere el algoritmo que se elaboraría para el problema o situación de levantarse todas las mañanas para ir al trabajo:

1. Salir de la cama
2. quitarse el pijama
3. ducharse
4. vestirse
5. desayunar
6. arrancar el vehículo para ir al trabajo o tomar transporte.

Algoritmos Descriptivos

MARCO TEÓRICO: Existen dos maneras de expresar los algoritmos, en pseudocódigo y como diagramas de flujo de datos. Los algoritmos en pseudocódigo, también denominados algoritmos descriptivos tienen un formato más natural y entendible para detallar los pasos a seguir en la solución de un problema.

Para construir algoritmos descriptivos, es necesario identificar 3 grandes partes:

a) Qué **datos** requerimos para resolver el problema (Datos de Entrada)

b) Qué **operaciones** debemos realizar sobre los datos para ir resolviendo el problema (Proceso)

c) Qué **resultados** se van a proporcionar al finalizar el proceso (Datos de Salida).

Para iniciar la construcción del algoritmo, es importante detectar todos los datos involucrados en él, tratando de clasificarlos de acuerdo a su tipo y su uso, además de su categoría en variables o constantes, otra cuestión importante que se debe atender es ver si el dato, contiene ya alguna información o esta va a ser calculada según progrese el algoritmo.

Una segunda etapa consiste en formular aquellas expresiones que permitan calcular los resultados o que determinen el comportamiento del algoritmo como la toma de decisiones.

Finalmente debe escribirse el algoritmo sin omitir ningún paso y tomando en cuenta las recomendaciones anteriores.

Ejemplo

Se desea calcular la suma de dos números

Definición del problema: Elaborar un algoritmo para calcular la suma de dos números

Análisis del problema	Algoritmo	Diagrama de flujo
Entrada: *A y B representan los dos números.* **Proceso:** *Suma = A + B* **Salida:** *Resultado es Suma*	INICIO 1 LEER A y B 2 SUMA = A + B 3 ESCRIBIR("EL RESULTADO ES:"; SUMA) FIN	

Datos de Entrada y Salida en los Algoritmos

MARCO TEÓRICO: Al elaborar algoritmos es importante identificar los datos que se presentan y son necesarios para poder resolver el problema planteado. Estos datos se conocen como de entrada y son aquellos que la computadora va a procesar. Los datos de salida son datos derivados, es decir, obtenidos a partir de los datos de entrada. Por esta razón, a los datos de salida se les considera más significativos que a los datos de entrada. A los datos de entrada se les considera la materia prima de los datos de salida, considerados estos como la verdadera información.

Por ejemplo

Se desea realizar el algoritmo para calcular el área de distintos terrenos que poseen una forma rectangular.

Detectamos como datos: frente, fondo y área; los tres datos son de tipo numérico, y ninguno es constante ya que como se trata de distintos terrenos las dimensiones pueden variar. Se requieren los valores del frente y del fondo del terreno como Datos de Entrada, y el área como Dato de Salida será calculada en en función de los datos de entrada.

Por lo tanto, proponemos 3 variables: fr, fo y area, para almacenar las medidas del frente, el fondo del terreno, y el área del mismo respectivamente.

La expresión que permite resolver el problema es la siguiente:

area = fr * fo

A continuación se presenta una solución a través de un algoritmo descriptivo.

DATOS	**Identificadores**
Entrada	
Longitud del frente del terreno	fr
Longitud del fondo del terreno	fo
Salida	
Area del Terreno	area

ALGORITMO

Inicio

Leer fr

Leer fo

area ← fr * fo

Escribir area

Fin

De acuerdo al ejemplo, tomemos las siguientes convenciones: Las asignaciones (que indican cuando una variable toma algún valor), se denotarán por flechas ←.
Para los Datos de Entrada se utilizará la palabra Leer, y para mostrar los resultados o Datos de Salida la palabra Escribir.

Datos Intermedios en los Algoritmos

MARCO TEÓRICO: Existen datos que solo se utilizan para almacenar algún valor temporalmente y se les conoce como datos intermedios.

Ejemplo

Elabore un algoritmo que solicite el número de respuestas correctas, incorrectas y en blanco, correspondientes a postulantes, y muestre su puntaje final considerando, que por cada respuesta correcta tendrá 4 puntos, respuestas incorrectas tendrá -1 y respuestas en blanco tendrá 0.

Solución:

DATOS	**Identificadores**
Entrada	
Número de Respuestas Correctas	RC
Número de Respuestas Incorrectas	RI
Número de Respuestas en Blanco	RB
Salida	
Puntaje Final	PF
Intermedio	
Puntaje de Respuestas Correctas	PRC
Puntaje de Respuestas Incorrectas	PRI

ALGORITMO

Inicio

Leer RC

Leer RI

Leer RB

PRC ← RC * 4

PRI ← RI * -1

PF ← PRC + PRI

Escribir PF

Fin

Estructuras Condicionales Simples

MARCO TEÓRICO: Una acción muy frecuente y útil en la construcción de algoritmos es la toma de decisiones, que se lleva a cabo a través de estructuras condicionales. En los puntos donde aparecen estas decisiones, el algoritmo tiene dos posibilidades de funcionar, es decir puede optarse por realizar ciertas instrucciones del algoritmo o por no hacerlas.

Las condiciones se pueden incluir en el algoritmo de la misma manera en que las utilizamos en nuestro actuar cotidiano, por ejemplo: Voy a salir , si está nublado"; "Si paso el examen de admisión, estudio en la universidad"; "Si tengo dinero, me compro un celular"

La implementación de estas sentencias dentro de los algoritmos se llevará a cabo bajo las siguientes convenciones:

La estructura condicional tiene la forma:

Si (condicion) Entonces

instrucciones

Fin Si

Y su interpretación es: Se evalúa la condición (que está formada por expresiones relacionales o lógicas) y que puede tener un valor VERDADERO o FALSO, dependiendo de los datos involucrados en la expresión. Sólo en el caso de que su valor sea VERDADERO, se realizarán las instrucciones que se encuentren después del "Entonces" hasta el "Fin si", si por el contrario la expresión resulta FALSA, estas instrucciones no serán tomadas en cuenta y el algoritmo continuará con la instrucción que aparezca después del "Fin si".

Ejemplo: Determinar con base a la edad de una persona si puede obtener una licencia para conducir.

DATOS

	Identificadores
Entrada	
Edad de la persona	edad
Salida	
Mensaje de salida	no requiere

ALGORITMO

Inicio

Escribir "Proporcione su edad"

Leer edad

Si (edad > 18) Entonces

Escribir "Licencia autorizada"

Fin si

Fin

Estructuras Condicionales Dobles

MARCO TEÓRICO: Hay situaciones en la que no solo existe una opción de decisión, sino que se nos presentan dos opciones o alternativas posibles, dependiendo si se cumple la condición planteada en un problema. Por ejemplo: "Una persona planea ir a un balneario si el día esta soleado, de lo contrario, visitará un museo"; esta situación presenta 2 alternativas de acción excluyentes entre si dependiendo del estado del clima"

Para representar este tipo de problemas se utilizan estructuras condicionales dobles. Su estructura es:

Si (condicion) Entonces

Instrucción Verdadera (V)

Sino

Instrucción Falsa (F)

Fin Si

Ejemplo: Se desea comparar dos números diferentes y mostrar el número mayor.
El caso anterior lo podemos expresar por medio del siguiente algoritmo:

DATOS	**Identificadores**
Entrada	
Primer número	numa
Segundo número	numb
Salida	
Número mayor	No requiere

ALGORITMO

Inicio

Leer numa

Leer numb

Si (numa > numb) entonces

Escribir numa

Sino

Escribir numb

Fin si

Fin

Estructuras Condicionales Anidadas

Se dice que una estructura condicional es anidada cuando por la alternativa del Verdadero o del Falso de la estructura condicional, hay otra estructura condicional.

```
Si (condicion1) Entonces
    Si (condicion2) Entonces
        Instrucción/es
    Sino
        Instrucción/es
    Fin Si
Sino
    Si (condicion3) Entonces
        Instrucción/es
    Sino
        Instrucción/es
    Fin Si
Fin Si
```

Ejemplo

Escribir un algoritmo que permita evaluar la edad de una persona, y que informe en un mensaje si la edad ingresada es mayor, menor, o igual a 21.

DATOS	Identificadores
Entrada	
Edad de la persona	edad
Salida	
Mensaje de salida	No requiere

ALGORITMO

```
Inicio
Leer edad
Si (edad > 21) Entonces
    Msgbox "La edad ingresada es mayor a 21"
Sino
    Si (edad < 21) Entonces
        Msgbox "La edad ingresada es menor a 21"
    Sino
        Msgbox "La edad ingresada es igual a 21"
    Fin Si
Fin Si
Fin
```

Estructuras Condicionales Múltiples

MARCO TEÓRICO: Cuando se presentan situaciones en las que no solo se tienen dos posibilidades al tomar una decisión, se dispone de instrucciones condicionales múltiples que tienen una estructura muy sencilla basada en **casos** para ser realizadas. Por ejemplo, pensemos en un sistema de conversión de calificaciones cuantitativas (numéricas) a cualitativas (dictámenes), entonces puede aplicarse una especie de "tabla de valores que relacione todos los posibles casos con una acción similar a la siguiente:

Calificación	Dictamen
0-6	"No suficiente"
7	"Suficiente"
8	"Bien"
9	"Muy bien"
10	"Excelente"

La notación o forma de expresar instrucciones de este tipo dentro de un algoritmo sería de la siguiente manera:

```
Selecciona (valor ordinal)
   Caso valor1: instrucciones
   Caso valor2: instrucciones
   Caso valor3: instrucciones
      .
      .
   Otros casos: instrucciones
Fin Casos
```

La interpretación de una estructura de este tipo es:

Se verifica el valor de la expresión localizada dentro de los paréntesis, es importante saber que esta expresión (que puede ser incluso una simple variable) debe tener un valor ordinal, es decir un número entero, un caracter, o una cadena de caracteres.

De acuerdo la evaluación de esta expresión ordinal, lo siguiente a realizarse es la instrucción o instrucciones delante del caso con el valor que corresponde a la expresión, una vez terminado el bloque de instrucciones correspondiente, brincará hacia la siguiente instrucción que aparezca después del "Fin Casos"

Se muestra el ejemplo de un algoritmo con la información definida en la tabla anterior.

DATOS	Identificadores
Entrada	
Nombre del alumno	nom
Calificación numérica	calif
Salida	
Nombre del alumno	nom
Dictamen cualitativo	No requiere

ALGORITMO

Inicio

Escribir “Proporcione su nombre: “

Leer nom

Escribir “Proporcione su calificación”

Leer calif

Seleccionar (calif)

Caso 10: Escribir nom, “tiene una calificación excelente”

Caso 9: Escribir nom, “tiene una calificación muy buena”

Caso 8: Escribir nom, “tiene una calificación buena”

Caso 7: Escribir nom, “tiene una calificación suficiente”

Otros casos: Escribir nom, “tiene una calificación no suficiente”

Fin Si

Fin