ALGORITMIA BÁSICA

CONTENIDO

# CAPÍTULO I: INTRODUCCIÓN

## Desarrollo de aplicaciones

## Elaboración de algoritmos y representación de datos

# CAPÍTULO 2: ESTRUCTURAS BÁSICAS EN EL DESARROLLO DE ALGORITMOS

## Estructuras para la construcción de algoritmos e instrucciones de lectura y escritura.

## Instrucción de asignación y expresiones aritméticas.

## Manipulación de expresiones algebraicas.

## Expresiones relacionales y lógicas.

## Aplicaciones del capítulo.

CAPÍTULO 3: ESTRUCTURAS DE DECISIÓN

## Estructura de decisión

## Componente DE\_LO\_CONTRARIO

## Decisiones anidadas

## Selección múltiple

## Ejemplo de uso de la instrucción CASOS.

CAPÍTULO 4: ESTRUCTURA CÍCLICAS

## Estructura ciclo e instrucción MIENTRAS

## Contadores y acumuladores

## Aplicaciones de ciclos en matemáticas

## Ciclos anidados con instrucción MIENTRAS

## Instrucción PARA

## Ciclos anidados con instrucción PARA

## Instrucción HAGA

CAPÍTULO 5: SUBPROGRAMAS

## Concepto e identificación

## Clases y aplicación

## Parámetros y variables

## Ejemplos de uso

## Ejemplos de uso mejorados

CAPÍTULO 6: VECTORES

## Identificar el uso de un vector

## Definición y construcción

## Operaciones básicas: mayor dato, menor dato e intercambio de datos

## Operaciones básicas: procesos de inserción en un vector ordenado ascendentemente.

## Operaciones básicas: procesos de borrado en un vector

## Operaciones básicas: búsqueda binaria y ordenamiento por selección

## Operaciones básicas: ordenamiento por burbuja

CAPÍTULO 7: ESTRUCTURA ESTÁTICA DE DOS DIMENSIONES: MATRIZ

## Definición, identificación y construcción de una matriz

## Recorridos sobre matrices

## Suma de los datos de fila y columnas de una matriz

## Clases de matrices

## Intercambio de filas y columnas de una matriz

## Ordenamiento de los datos de una matriz con base en los datos de una columna

## Construcción de la transpuesta de una matriz y suma de matrices

## Multiplicación de matrices

CAPÍTULO 8: PROGRAMACIÓN ORIENTADA A OBJETOS

## Introducción a la programación orientada a objetos

## Desarrollo de los métodos de la clase vector

## Herencia y polimorfismo dinámico en vectores

## La clase matriz

**Objetivo general**

Desarrollar la habilidad para elaborar algoritmos correspondientes a soluciones que serán implementadas como programas de computador.

**Objetivos específicos**

1. Ubicar la elaboración de algoritmos dentro del desarrollo de soluciones utilizando el computador como herramienta.
2. Reconocer la estructura secuencial con sus instrucciones de lectura, escritura, asignación, decisión y ciclo.
3. Reconocer la estructura de decisión y utilizar sus instrucciones: SI, su componente opcional DE\_LO\_CONTRARIO, y CASOS.
4. Reconocer la estructura ciclo y utilizar sus instrucciones: MIENTRAS, PARA y HAGA.
5. Identificar las diferentes tareas que se ejecutan en un algoritmo para diseñar cada una de ellas como un subprograma y poder elaborar algoritmos más legibles, más sencillos y reutilizables.
6. Identificar y usar la estructura arreglo en sus formas básicas: vectores y matrices.

**CAPÍTULO I: INTRODUCCIÓN**

**Desarrollo de aplicaciones**

Cuando se va a desarrollar una aplicación usando el computador como herramienta se tiene establecida cierta metodología para garantizar que la aplicación desarrollada sea de buena calidad. Los pasos que establece dicha metodología son:

1. Análisis del problema
2. Diseño de la solución
3. Implementación de la solución planteada
   1. Elaboración de algoritmos
      1. Análisis del problema
      2. Diseño de la solución
      3. Construcción del algoritmo
      4. Prueba de escritorio
   2. Codificación en algún lenguaje de programación
   3. Compilación
   4. Pruebas de algoritmo
4. Pruebas del sistema
5. Puesta en marcha

* El análisis del problema consiste en estudiar el problema al cual se le desea dar solución utilizando el computador como herramienta. Este estudio incluye entrevistas con los usuarios y con todas las personas que conocen cómo resolver el problema externamente al computador; se investigan los datos que se requieren, la información que se desea producir y los casos especiales que se puedan presentar. El análisis es fundamental. Un buen análisis tendrá como consecuencia un buen diseño.
* El diseño de la solución consiste en elaborar un modelo en el cual se consideran todas las situaciones que se identificaron en el análisis: se definen bases de datos, archivos, procesos, recursos y todo lo necesario para solucionar el problema.
* La implementación consiste en conseguir los recursos necesarios, construir las bases de datos y los archivos y elaborar todos los programas para que el sistema funcione apropiadamente.
* Las pruebas consisten en suministrar, al modelo desarrollado, datos ficticios con el fin de comprobar que el sistema responde correctamente a todas las situaciones que se hallaron en el análisis y que se implementaron como programa de computador. Esta fase de pruebas es muy importante puesto que allí se pueden detectar fallas y situaciones que no se consideraron, por alguna razón, en el análisis y en el diseño. Las pruebas deben estar supervisadas en forma exhaustiva por los usuarios del sistema y por sus desarrolladores.
* La puesta en marcha consiste en liberar el sistema desarrollado, para que el usuario lo utilice en su labor diaria. En esta fase de puesta en marcha se hace seguimiento al funcionamiento de la aplicación para efectuar los últimos ajustes en caso de que sea necesario.

Durante este trimestre nos enfocaremos en el punto Elaboración de algoritmos.

**Elaboración de algoritmos y representación de datos**

**Un algoritmo es un conjunto de instrucciones que son tan precisas y tan bien hechas que basta con que una persona sepa leer para ejecutar correctamente una tarea que nunca había ejecutado**. En este punto se presentan los pasos que se siguen en la elaboración de algoritmos, explicando las acciones que deben efectuarse en cada uno de ellos y los elementos necesarios para construir algoritmos. Se presenta además la jerarquía de conceptos en la representación de datos, explicando en detalle cada uno de dichos conceptos con sus correspondientes códigos y usos.

Objetivos

1. Identificar los pasos que se siguen en la construcción de un algoritmo.
2. Reconocer la jerarquía de representación de datos en un computador.

Preguntas básicas

1. ¿Cuáles son los pasos en la construcción de un algoritmo?
2. ¿Cuáles son los elementos con los que se construye un algoritmo?
3. ¿En qué consiste el análisis de un problema?
4. ¿Qué es una prueba de escritorio?
5. ¿Cuál es la jerarquía de representación de datos en un computador?

Contenidos

1. Pasos en la construcción de un algoritmo
2. Elementos para la construcción de un algoritmo

**Pasos en la construcción de un algoritmo**

Los pasos que se siguen en la construcción de un algoritmo son:

1. Análisis del problema.
2. Diseño de la solución.
3. Construcción del algoritmo.
4. Prueba de escritorio.

El análisis consiste en determinar exactamente cuáles son los datos de entrada que se requieren, cuál es la información que se desea producir y cuál es el proceso que se debe efectuar sobre los datos de entrada para producir la información requerida. Se debe indagar por todas las situaciones especiales que se puedan presentar para tenerlas en cuenta en el diseño.

Con base en el análisis se elabora el diseño del algoritmo: se asignan nombres a las variables, se define el tipo de cada una de ellas, se definen las operaciones y/o subprocesos que hay que efectuar y el método para resolver cada uno de ellos.

Los elementos para la construcción de un algoritmo son: datos, estructuras e instrucciones. Más adelante hablaremos en detalle de cada uno de ellos.

La prueba de escritorio consiste en asumir la posición del computador y ejecutar el algoritmo que se ha elaborado para ver cómo es su funcionamiento. Esta parte es muy importante puesto que permite detectar errores de lógica sin haber usado aún el computador. Aunque no garantiza que el algoritmo está bueno en el 100%, ayuda mucho en la elaboración de algoritmos correctos.

Habiendo superado los pasos anteriores, se elige un lenguaje de programación, se codifica el algoritmo en dicho lenguaje y se pone en ejecución en el computador disponible. En nuestro caso, vamos a usar el lenguaje de programación Java.

**Elementos para la construcción de un algoritmo**

Los elementos con los cuales se construye un algoritmo son las estructuras lógicas y los datos. Comencemos con los datos.

Para efectos de representación de datos en un computador, éstos se clasifican en numéricos y no numéricos, y los datos numéricos se clasifican en enteros y reales.

En términos de computación esta clasificación se denomina tipo, y se habla entonces de datos de tipo entero, de tipo real, de tipo no numérico, etc.

Cuando se trabajan datos numéricos en un computador es muy importante considerar si el tipo es entero o real, puesto que, dependiendo de ello, los resultados que se obtienen al efectuar operaciones aritméticas pueden variar sustancialmente. Cuando veamos la evaluación de expresiones haremos notar esta diferencia.

**Representación de datos en un computador**

La unidad básica de representación es el bit, que significa **binary digit**. Un bit es un elemento biestable, que solo puede tener dos valores, como un bombillo, está prendido o apagado. Por convención se ha definido que un estado representa el 0 y el otro estado representa el 1. Y esa es la razón por la cual se denomina dígito binario.

Siguiente orden de jerarquía, seguimos con el **byte**, que es un conjunto de 8 bits. Ese conjunto de 8 bits, esta numerado desde el bit 0 hasta el bit 7.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **7** | **6** | **5** | **4** | **3** | **2** | **1** | **0** |
| 0 | 1 | 1 | 1 | 0 | 0 | 1 | 0 |

Como cada elemento es un bit, en cada elemento estará almacenado un 0 o un 1.

Si tenemos un conjunto de 8 bit, podemos tener muchas diferentes combinaciones de 1 y 0. Entonces, por convención se ha establecido una combinación de 1 y 0 me va a representar la A mayúscula, esta otra combinación de 1 y 0, me va a representar la b minúscula, esta otra la P mayúscula, esta otra el número 1, y así sucesivamente.

Esta convención que se ha establecido para identificar cuál es el carácter que representa cada combinación, se denomina código ASCII.

American Standard Code Information Interchange: Código Estándar Americano para Intercambio de Información.

Esta convención es la que ha sido aceptada mundialmente, y en la actualidad en casi todos los computadores, los bytes se rigen por el código ASCII.

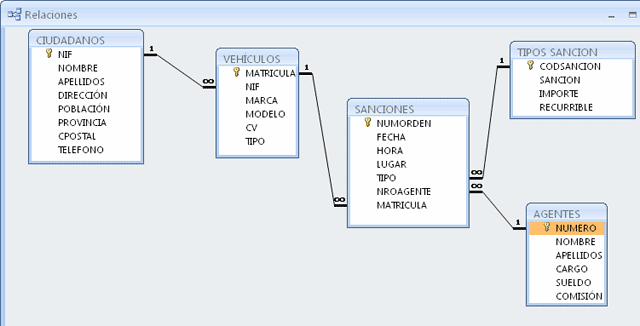
Siguiendo en la jerarquía, a continuación del byte, tenemos el **campo**, que es un conjunto de bytes.

Si queremos representar el nombre de una persona dentro de un computador (Pedro), necesitamos un byte para la letra “p”, otro para la letra “e”, otro para la letra “d”, otro para la letra “r” y otro para la letra “o”.

Luego tenemos el **registro**, que es un conjunto de campos. Si queremos describir a una persona totalmente, necesitamos un campo para almacenar el nombre, otro para la dirección, otro para el teléfono, otro para fecha de nacimiento, dirección electrónica. Todo eso que identifica plenamente a una persona, es lo que se denomina un **registro**.

**Archivo**: conjunto de registros. Si queremos describir a un conjunto de personas, como a los estudiantes de un curso, necesitamos datos como: nombre, edad, dirección. Para cada estudiante se necesita un registro. Todo ese conjunto de registros con lo que estoy describiendo a todos los estudiantes, es lo que se conoce como archivo.

**Base de datos**: Es un conjunto de Archivos relacionados entre sí.



**CAPÍTULO 2: ESTRUCTURAS BÁSICAS EN EL DESARROLLO DE ALGORITMOS**

El concepto de algoritmo, como un conjunto de instrucciones con sus datos, constituye la parte fundamental del capítulo.

En este capítulo se estudian los importantes conceptos de: datos, constantes, variables, expresiones, instrucciones de asignación, instrucciones de entrada e instrucciones de salida.

**Estructuras para la construcción de algoritmos e instrucciones de lectura y escritura.**

Aquí se presentan las estructuras básicas para la construcción de algoritmos (cada una de ellas con sus respectivas instrucciones), la forma general en que se escribirán los algoritmos que se elaboren en este texto, la instrucción de lectura y la instrucción de escritura.

Objetivos

1. Identificar las estructuras lógicas, con sus correspondientes instrucciones, para la construcción de algoritmos.
2. Reconocer la forma general de un algoritmo.
3. Aplicar la instrucción de lectura.
4. Aplicar la instrucción de escritura.

Preguntas básicas

1. ¿Cuáles son las estructuras lógicas necesarias para construir un algoritmo?
2. ¿En qué consiste la estructura secuencia?
3. ¿Cuál es la forma general de un algoritmo?
4. ¿En qué consiste la instrucción de lectura?
5. ¿En qué consiste la instrucción de escritura?

Contenidos

1. Estructuras lógicas para la construcción de algoritmos
2. Forma general de un algoritmo
3. Instrucción de lectura
4. Instrucción de escritura

**Estructuras lógicas para la construcción de algoritmos**

Las estructuras lógicas para la construcción de algoritmos son:

1. Estructura de secuencia.
2. Estructura de decisión.
3. Estructura ciclo.

La estructura básica en la construcción de un algoritmo es la estructura de secuencia. Esta estructura consiste en que las instrucciones se ejecutan exactamente en el orden en que han sido escritas: primero se ejecuta la primera instrucción, luego la segunda instrucción, luego la tercera instrucción y por último la última instrucción.

*El orden en el cual se escriben las instrucciones es fundamental para el correcto funcionamiento de un algoritmo.*

Cada estructura consta de un conjunto de instrucciones.

Las instrucciones correspondientes a la estructura de secuencia son:

1. Instrucciones de asignación.
2. Instrucciones de lectura.
3. Instrucciones de escritura.
4. Las instrucciones correspondientes a la estructura de decisión.
5. Las instrucciones correspondientes a la estructura ciclo.

Las instrucciones correspondientes a la estructura de decisión son:

1. La instrucción SI y su componente opcional DE\_LO\_CONTRARIO.
2. La instrucción CASOS.

Las instrucciones correspondientes a la estructura ciclo son:

1. La instrucción MIENTRAS.
2. La instrucción PARA.
3. La instrucción HAGA

**Forma general de un algoritmo**

La forma general de nuestros algoritmos será:

**Algoritmo** nombreDelAlgoritmo

Definición de variables

INICIO

Instrucciones del programa

FIN

Fin (nombreDelAlgoritmo)

**Instrucción de lectura**

Para que el computador pueda procesar datos, éstos deben estar en la memoria principal (RAM). La instrucción de lectura consiste en llevar los datos con los cuales se desea trabajar, desde un medio externo hacia la memoria principal.

Los medios externos en los cuales pueden residir los datos son: disco duro, disco removible, etc; los datos también pueden entrase directamente a través del teclado.

La forma general de la instrucción de lectura es:

LEA (lista de variables, separadas por comas)

En nuestro caso, en los algoritmos consideraremos que los datos entran directamente desde el teclado.

**Instrucción de escritura**

La instrucción de escritura consiste en llevar datos desde la memoria hacia un medio externo, el cual puede ser un disco duro, una impresora o la pantalla.

La forma general de la instrucción de escritura es:

ESCRIBA (lista de variables y/o mensajes, separados por comas)

Los mensajes son para instruir al usuario acerca de los datos que se le están presentando. Si el dato que se imprime es la estatura de una persona, es conveniente que esté precedido por un mensaje que diga: estatura. Si el dato que se está presentando es una edad, es conveniente que esté precedido por un mensaje que diga: edad. Y así sucesivamente.

Cuando vayamos a escribir un mensaje en una instrucción de lectura, dicho mensaje lo escribiremos encerrado entre comillas.

Vamos a hacer nuestro primer algoritmo utilizando sólo las instrucciones de lectura y de escritura.

1 Algoritmo Uno

Variables:

2 nom: alfanumérica

3 Tel: numericaEntera

4 INICIO

5 ESCRIBA (“Teclee nombre y teléfono”)

6 LEA (nom, tel)

7 ESCRIBA (“nombre: “, nom, “teléfono: “, tel)

8 FIN

9 Fin (Uno)

En la instrucción 1 estamos definiendo el nombre del algoritmo: lo estamos llamando *Uno*.

En la instrucción 2 estamos definiendo las variables que vamos a utilizar en nuestro programa. Ponemos el título *variables* y definimos la variable **nom**, que podrá almacenar datos alfanuméricos, y en la instrucción 3 definimos la variable **tel**, que podrá almacenar datos numéricos enteros.

En la instrucción 4 ponemos nuestra palabra clave INICIO, la cual indica que a partir de ahí están las instrucciones de nuestro algoritmo.

En la instrucción 5 ponemos nuestra instrucción de escritura, la cual escribe el mensaje “Teclee nombre y teléfono” instruyendo al usuario acerca de los datos que debe teclear.

En la instrucción 6 ponemos nuestra instrucción de lectura, en la cual los datos tecleados por el usuario se almacenarán en las posiciones de memoria que el computador identificará con los nombre **nom** y  **tel**.

En la instrucción 7 ponemos la instrucción de escritura con la cual se escriben los datos tecleados por el usuario, cada uno con su respectivo título.

La instrucción 8 es nuestra palabra clave FIN, que cierra el inicio de las instrucciones del algoritmo, y en la instrucción 9 ponemos el fin del algoritmo.

**Ejercicios propuestos**

* + 1. ¿Cómo se diferencia la escritura de mensajes de la escritura de datos en una instrucción de escritura?
    2. Detecte y describa los errores que hay en el siguiente algoritmo. Proponga soluciones.

Algoritmo jabón

Variables: a, b: numericaEntera

INICIO

ESCRIBA (“teclee los datos para a y b”)

ESCRIBA (“dato a: “, a, “”dato b: “, b)

LEA (a, b)

FIN

Fin (jabón)

* + 1. ¿Cuáles son las instrucciones correspondientes a la estructura ciclo?
    2. ¿Cuáles son las instrucciones correspondientes a la estructura de decisión?
    3. ¿Cuáles son las instrucciones correspondientes a la estructura de secuencia?

**Instrucción de asignación y expresiones aritméticas**

Aquí se tratará lo correspondiente a la instrucción de asignación, la cual consiste en llevar datos hacia alguna posición de la memoria principal del computador, y todo lo correspondiente a las expresiones aritméticas: construcción, operadores, tipos y evaluación.

Objetivos

* + 1. Emplear la instrucción de asignación.
    2. Construir y manipular expresiones aritméticas.

Preguntas básicas

1. ¿En qué consiste una instrucción de asignación?
2. ¿Cómo se evalúa una expresión aritmética?
3. ¿Cuál es el orden de evaluación de los operadores en una expresión aritmética?
4. ¿En qué consiste la asociatividad de un operador?
5. ¿Cómo afecta el tipo de operando el resultado de evaluar una expresión?

Contenidos

1. Instrucción de asignación
2. Expresiones aritméticas

**Instrucción de asignación**

La instrucción de asignación consiste en llevar algún dato a una posición de memoria, la cual está identificada con el nombre de una variable.

La forma general de una instrucción de asignación es:

|  |  |  |
| --- | --- | --- |
|  |  | -constante numérica entera |
|  | -constante numérica real |
| Variable= | -variable |
|  | -mensaje |
|  | -expresión |

Ejemplos:

1. a = 316
2. b = 3.14
3. c = «hola mundo»
4. d = a
5. e = a + b \* a

En los ejemplos 1 y 2, a las variables a y b les estamos asignando una constante numérica: entera en el primer ejemplo, real en el segundo. En el ejemplo 3, a la variable c le estamos asignando un mensaje. En el ejemplo 4, a la variable d le estamos asignando el contenido de otra variable. En el ejemplo 5, a la variable e le estamos asignando el resultado de evaluar una expresión.

Vamos a ocuparnos de lo que son expresiones.

Expresiones aritméticas

En general, una expresión es una sucesión de operandos y operadores, la cual puede ser de tres clases:

1. Expresión aritmética.
2. Expresión relacional.
3. Expresión lógica.

Comencemos tratando lo que son las expresiones aritméticas:

Una *expresión aritmética* es una sucesión de operandos y operadores aritméticos, en la cual los operadores actúan sobre los operandos.

Los operadores aritméticos son (tabla 4.1):

Tabla 4.1. Operadores aritméticos y su significado

|  |  |
| --- | --- |
| **Operador** | **Operación** |
| ^ | Potenciación |
| \* | Multiplicación |
| / | División |
| % | Módulo |
| + | Suma |
| ― | Resta |

De los operadores mostrados vale la pena indicar en qué consiste el operador módulo (%).

*El operador módulo calcula el residuo de una división entera.*

Si tenemos la instrucción

a = 12 % 7

en la posición de memoria identificada con la variable a quedará almacenado el valor de 5.

Cuando se escribe una expresión aritmética es necesario tener en cuenta las características de evaluación de dicha expresión. Dichas características incluyen lo que es la prioridad de ejecución de los operadores y la asociatividad de cada uno de ellos. La prioridad se refiere a cuáles operaciones se ejecutan primero y cuáles de últimas, y la asociatividad se refiere al orden en el cual se ejecutan operaciones consecutivas con la misma prioridad.

En expresiones aritméticas de computador la prioridad de ejecución de las operaciones es la siguiente:

1. Primero se ejecutan las operaciones de potenciación.
2. En segundo lugar se ejecutan las operaciones de multiplicación, división y módulo.
3. En tercer lugar se ejecutan las operaciones de sumas y restas.

La asociatividad se refiere a:

1. Cuando se hallan operaciones consecutivas de potenciación, éstas se ejecutan de derecha a izquierda, es decir, la operación de potenciación es asociativa por la derecha.
2. Las demás operaciones son asociativas por la izquierda, es decir, se ejecutan en el orden que aparezcan, de izquierda a derecha. En otras palabras, las operaciones multiplicación, división, módulo, suma y resta son asociativas por la izquierda.

Considerando estas dos características vamos a presentar algunos ejemplos que las ilustren.

Consideremos que en las variables a, b, c, d, e y f están almacenados los siguientes datos

(tabla 4.2):

Tabla 4.2. Variables con sus valores.

|  |  |
| --- | --- |
| **Variable** | **Valor** |
| a | 4 |
| b | 5 |
| c | 6 |
| d | 3 |
| e | 2 |
| f | 4 |

y consideremos la expresión

a + b \* c

Al evaluar la expresión, primero ejecuta el producto b \* c, y se obtiene como resultado 30.

La expresión queda:

a + 30

Luego ejecuta la suma y el resultado es 34.

Consideremos algunos ejemplos:

1. Si tenemos la expresión

a + b \* c / d – e ^ f

el resultado de evaluar dicha expresión, considerando los valores mostrados en la tabla 4.2, es:

primero ejecuta e ^ f, y la expresión queda:

a + b \* c / d – 16

luego ejecuta b \* c, y la expresión queda:

a + 30 / d – 16

luego ejecuta 30 / d, y la expresión queda:

a + 10 – 16

luego ejecuta a + 10, y la expresión queda:

14 – 16

por último ejecuta 14 – 16, y se obtiene un resultado de –2.

**Manipulación de expresiones algebraicas**

En el punto anterior se trataron las expresiones aritméticas como parte de la instrucción de asignación y se estudiaron las características que tienen dichas expresiones. Aquí se verá la relación que existe entre lo que son las expresiones aritméticas de computador frente a las expresiones algebraicas, cómo se establece la correspondencia entre ellas y cómo convertir una expresión de computador en expresión algebraica y viceversa.

Objetivos

1. Construir una expresión lineal de computador con base en una expresión algebraica.
2. Construir una expresión algebraica con base en una expresión lineal de computador.

Preguntas básicas

1. ¿Para qué se utilizan paréntesis en expresiones de computador?
2. ¿Cómo se construye una expresión de computador para operaciones de potenciación?

Contenidos

1. Conversión de una expresión algebraica a expresión lineal de computador
2. Conversión de una expresión lineal de computador a expresión algebraica

**Conversión de una expresión algebraica a expresión lineal de computador**

Cuando se elabora un algoritmo es muy común tener que escribir expresiones, sobre todo si se trata de algoritmos de carácter científico o matemático. Para escribir expresiones de computador, según el módulo anterior, es necesario tener en cuenta la forma como el computador evalúa dichas expresiones.

Vamos a ver cómo convertir una expresión algebraica en expresión de computador. Comencemos considerando un primer ejemplo. Sea la expresión algebraica

|  |
| --- |
| a |
| b \* c |

Si queremos escribir esta expresión algebraica como expresión de computador, tenemos varias formas de hacerlo:

1. a / b \* c
2. a / b / c
3. a / (b \* c)

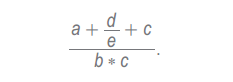
La primera forma es incorrecta, porque de acuerdo a lo visto en el módulo anterior, primero ejecuta la división del valor de a entre el valor de b, y el resultado lo multiplica por el valor de c. O sea que si a vale 36, b vale 6 y c vale 2 el resultado de evaluar dicha expresión es 12, lo cual es erróneo.

La segunda y tercera formas son correctas: en la segunda forma, primero ejecuta la división del valor de a por el valor de b y el resultado lo divide por el valor de c, obteniendo como resultado 3. En la tercera forma primero multiplica el valor de b por el valor de c y el resultado divide al valor de a, obteniendo como resultado también 3.

Es supremamente importante entender este primer ejemplo. En la tercera forma hemos utilizado paréntesis para alterar el orden de ejecución de las operaciones; sin embargo, en la segunda forma no los hemos utilizado y el resultado también es correcto.

Pasemos a considerar ejemplos más complejos, algunos de los cuales exigen el uso de paréntesis.

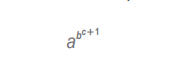
1. Veamos la siguiente expresión algebraica:



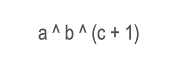
La expresión de computador correcta puede ser:

1. (a + d / e + c) / b / c o
2. (a + d / e + c) / (b \* c)

Veamos esta otra expresión algebraica



La forma correcta de escribir esta expresión algebraica como expresión de computador es:



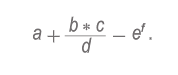
**Conversión de una expresión lineal de computador a expresión algebraica**

Pasemos ahora a considerar el caso contrario: dada una expresión de computador, escribir la expresión algebraica correspondiente.

Veamos los siguientes ejemplos:

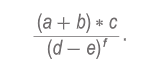
a + b \* c/d – e ^ f

La expresión algebraica es:



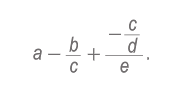
(a + b) \* c/ (d – e) ^ f

La expresión algebraica es:



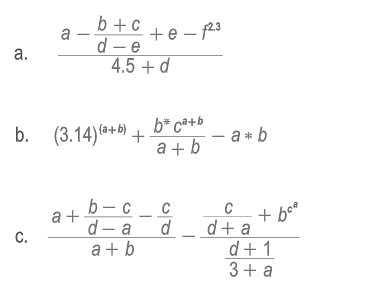
a – b / c + (b – c / d) / e

La expresión algebraica es:

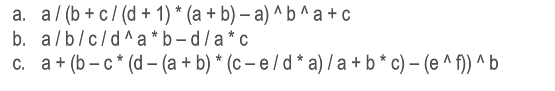


**Ejercicios propuestos**

1. Convierta las siguientes expresiones algebraicas en expresiones de computador:



1. Convierta las siguientes expresiones de computador en expresiones algebraicas:



**Expresiones relacionales y lógicas**

En este punto se tratará lo correspondiente a expresiones relacionales y expresiones lógicas, las cuales, al ser evaluadas, arrojan un resultado lógico, es decir, verdadero o falso. Con base en estas expresiones se podrán detectar diferentes situaciones que se presentan cuando se construyen algoritmos y se podrá instruir a la máquina para que ejecute las operaciones apropiadas según el resultado de una condición.

Objetivos

1. Construir y manipular expresiones relacionales.
2. Construir y manipular expresiones lógicas.

Preguntas básicas

1. ¿Qué es una expresión relacional?
2. ¿Qué resultado se obtiene al evaluar una expresión relacional?
3. ¿Cuáles son los operadores relacionales?
4. ¿Qué es una expresión lógica?
5. ¿Cuáles son los operadores lógicos?

Contenidos

1. Expresiones relacionales
2. Expresiones lógicas

**Expresiones relacionales**

Las expresiones relacionales producen un resultado lógico: verdadero o falso. Se construyen con base en operandos y operadores relacionales.

|  |  |  |
| --- | --- | --- |
| Operandos |  | - Variables |
| -Constantes |
| -Expresiones aritméticas |

|  |  |  |  |
| --- | --- | --- | --- |
| Operadores relacionales |  | > | Mayor que |
| >= | Mayor o igual que |
| < | Menor que |
| <= | Menor o igual que |
| == | Igual |
| <> | Diferente |

Cuando se trata de evaluar una expresión relacional primero se evalúan las expresiones aritméticas y luego se evalúan los operadores relacionales.

Ejemplos de expresiones relacionales:

1. a>b
2. 3.14<= b + c
3. (b - c) / 2.54 == a – b \* c / d

En el ejemplo 1 simplemente compara el valor almacenado en a con el almacenado en b. Si el valor almacenado en a es mayor que el almacenado en b, el resultado de evaluar la expresión es verdadero; de lo contrario es falso.

En el ejemplo 2 primero ejecuta la suma de b con c, y el resultado lo compara con 3.14. Si 3.14 es menor o igual que la suma de b con c, el resultado de evaluar la expresión es verdadero; de lo contrario es falso.

En el ejemplo 3 primero evalúa la expresión aritmética del lado izquierdo del operador == de acuerdo al orden de evaluación de expresiones aritméticas, luego la expresión aritmética del lado derecho del operador ==, y por último compara los resultados obtenidos. Si dichos resultados son iguales, el resultado de evaluar la expresión será verdadero; de lo contrario, falso.

**Expresiones lógicas**

Son expresiones que también producen un resultado lógico. Están construidas con operandos

y operadores lógicos.

|  |  |  |
| --- | --- | --- |
| Operandos |  | -Expresiones relacionales |
| -Variable lógicas |

|  |  |  |  |
| --- | --- | --- | --- |
| Operadores lógicos |  | ! | Negación |
| && | Conjunción |
| || | Disyunción |

Variable lógicas son variables que sólo pueden tomar dos valores: verdadero o falso.

En general, una variable lógica, en el ambiente de computadores, es una variable de un solo bit, el cual puede ser 0 o 1. Por convención se ha adoptado que el 0 representa falso y el 1 verdadero.

Cuando se trata de evaluar expresiones lógicas primero se evalúan las expresiones aritméticas, luego las expresiones relacionales y por último los operadores lógicos, los cuales también tienen cierta prioridad en el momento de efectuar la evaluación.

Prioridad de los operadores lógicos:

1. Negación
2. Conjunción
3. Disyunción

Ejemplos de expresiones lógicas:

1. a && b
2. a > b || c < d
3. 3.14 \* radio >= a ^ 2 || (b - c) == (3.1 + c) && (c + d) \* 2 <= 1

En el ejemplo 1, si **a** y **b** tienen estado de verdad, el resultado de evaluar la expresión es verdadero; de lo contrario es falso.

En el ejemplo 2, si **a** es mayor que **b**, o **c** es menor que **d**, el resultado de evaluar la expresión es verdadero; de lo contrario es falso.

En el ejemplo 3, primero se evalúan las expresiones aritméticas:

Llamemos **r1** el resultado de multiplicar 3.14 por el valor almacenado en la variable **radio**.

Llamemos **r2** el resultado de elevar el contenido de la variable **a** al cuadrado.

Llamemos **r3** el resultado de restarle a **b** lo que hay almacenado en **c**.

Llamemos **r4** el resultado de sumar 3.1 con el contenido de **c**.

Llamemos **r5** el resultado de multiplicar por 2 la suma de **c** con **d**.

Nuestra expresión quedará:

r1 >= r2 || r3 == r4 && r5 <= 1

Llamemos **r6** el resultado lógico obtenido de comparar **r1** con  **r2**.

Llamemos **r7** el resultado de comparar **r3** con **r4**.

Llamemos **r8** el resultado de comparar **r5** con 1.

Nuestra expresión queda:

r6 || r7 && r8

Luego evalúa **r7** && **r8**

Llamemos **r9** este resultado: si **r7** y **r8** son verdaderos, entonces **r9** será verdadero; de lo contrario **r9** será falso.

Nuestra expresión queda:

r6 || r9

en la cual, con uno de los dos operandos que sea verdadero, el resultado de evaluar la expresión será verdad.

**Ejercicios propuestos**

Dada la siguiente definición de variables con su respectivo tipo y contenido:

Numéricas enteras Numéricas reales Lógicas

a = 3 x = 2 m = T

b = 5 y = 3.5 n = F

c = 2 z = 5 o = F

d = 4 w = 1.2 p = T

donde T es verdad y F es falso, determine el resultado de evaluar cada una de las siguientes expresiones:

1. a + b <> 8 ^ d \* a / c + 2
2. a + b <> 8 ^ d \* (a / c) + 2
3. m && n || m && p
4. z / 2 < 2.5
5. m && (n || m) && p
6. c \* d – b ^ c >= y \* x / b || m && p && a + b > x + y

Ejercicios de algoritmos

Escriba los algoritmos que den solución a cada enunciado, teniendo en cuenta la metodología previamente estudiada y el ejemplo siguiente.

Elaborar un algoritmo que lea un dato correspondiente al lado de un cuadrado y que calcule e imprima el área de dicho cuadrado.

Análisis:

Datos de entrada:

Lado del cuadrado:lado

Datos de salida:

Área del cuadrado: área

Cálculos:

Determinar el área del cuadrado aplicando la fórmula lado al cuadrado.

El algoritmo será:

INICIO

Variables: lado, area: numéricaEntera

ESCRIBA (“Ingrese el lado del cuadrado: ”)

LEA (lado)

area= lado ^ 2

ESCRIBA (“El lado ingresado es: “, lado, “el área es: “ , area)

FIN

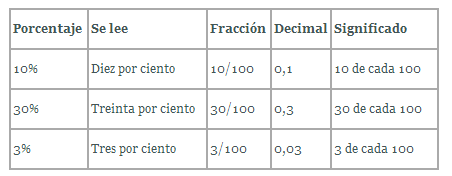
1. Elaborar un algoritmo que lea el salario actual de un empleado y el porcentaje de aumento y que calcule e imprima el salario actual, el aumento y el nuevo salario.
2. Elabore un algoritmo que lea el nombre de una persona y que imprima el mensaje “Hola” seguido del nombre de la persona leída.
3. Elabore un algoritmo que lea dos datos enteros correspondientes a los catetos de un triángulo y que calcule e imprima el valor de la hipotenusa de dicho triángulo.
4. Elabore un algoritmo que lea el código de un artículo, el precio unitario del artículo y la cantidad vendida. Su algoritmo debe calcular e imprimir el total de la venta, el IVA y el total a pagar, sabiendo que el impuesto es diecinueve por ciento.
5. Elabore un algoritmo que lea dos números enteros y que produzca como salida la suma, resta, multiplicación, división y módulo del primero por el segundo.
6. Elabore un algoritmo que lea una temperatura en grados Fahrenheit y la convierta y la imprima en grados centígrados. Los grados Fahrenheit se convierten a grados centígrados restándoles 32 y multiplicando por cinco novenos.
7. Elabore un algoritmo que lea un dato, el cual es el valor de un ángulo en grados, y que lo convierta e imprima en radianes.

**Aplicaciones de los porcentajes**

Los porcentajes se usan para:

* Relacionar una parte con el todo: Ejemplo: "El 58% de los aspirantes a ingresar en la Universidad son mujeres".
* Determinar una proporción entre dos cantidades: Ejemplo: "La proporción de levadura y harina para el bizcocho es del 3%".
* Describir a la población, indicando el peso relativo de una magnitud sobre ella. Ejemplo: "El 16% de la población tiene estudios superiores". Gran parte de la estadística se expresa en porcentajes.
* Determinar la variación relativa de una cantidad: Ejemplo: "El nivel del agua almacenada en los embalses ha subido un 8% en lo que va de año".

Cualquier porcentaje se puede expresar en forma de fracción o número decimal y, a su vez, cualquier número decimal o fracción se puede expresar en porcentaje:



**Cálculo de porcentajes**

Existen dos formas para hallar un porcentaje o tanto por ciento.

* Para calcular el porcentaje de una cantidad, multiplicamos la cantidad por el número que indica el porcentaje y dividimos el resultado entre 100.

Ejemplo:

El 20% de los estudiantes de un colegio, que tiene 240 alumnos, practica deporte. ¿Cuántos estudiantes practican deporte?

Para hallar la respuesta multiplicamos 240 por 20 y dividimos el resultado entre 100:

(240\*20)/100 = 48. Por tanto, el 20% de 240 alumnos = 48 alumnos.

* Para calcular el porcentaje de una cantidad, multiplicamos la cantidad por la expresión decimal de dicho porcentaje.

Ejemplo: Observa esta igualdad:

Para calcular el 20% de 240, basta con multiplicar 240 por 0,2:

240 x 0,2 = 48

**Incrementos**

Un incremento se produce cuando a una cantidad se le suma un porcentaje de la misma para obtener una cantidad mayor.

Ejemplo: Si una camiseta cuesta 12.000 para saber cuánto cuesta con incremento hay que:

*Calcular el incremento* que sufre el precio de la camiseta. Para ello, hallamos el porcentaje de la cantidad (19% de 12.000): 12.000 x 0,19 = 2.280 (0,19 es la expresión decimal del porcentaje 19%)

*Sumar la cantidad* (12.000) y su incremento (2.280) para obtener el precio final: 12.000 + 2.280 = 14.280.

El precio de la camiseta tiene un incremento y, por tanto, es necesario disponer de un total de 14.280 pesos para comprarla.

**Descuentos**

Un descuento se produce cuando a una cantidad se le resta un porcentaje de la misma para obtener otra cantidad menor.

Ejemplo: Vamos a calcular el precio de un libro que antes costaba 42.000 pesos y ahora tiene el 5% de descuento:

Calculamos el descuento que sufre el precio del libro. Para ello, hallamos el porcentaje de la cantidad (5% de 42,00): 42.000 x 0,05 = 2.100 (0,05 es la expresión decimal del porcentaje 5%).

Restamos la cantidad (42.000) menos su descuento (2.100) para obtener el precio final: 42.000 – 2.100 = 39.900 pesos. El precio del libro tiene un recuento y, por tanto, habría que disponer de 39.900 pesos para comprarlo.

**Ejercicios**

Ahora vamos a resolver los siguientes enunciados aplicando el concepto previamente estudiado.

* En un almacén todos los artículos tienen el 25% de descuento. ¿Cuánto pago por los siguientes artículos?

1 camisa cuesta $65.000, *el valor con el descuento es…*

1 pantalón cuesta $86.000*, el valor con el descuento es…*

1 bóxer cuesta $19.000, *el valor con el descuento es…*

* Los siguientes artículos deben incrementarse de precio, de acuerdo con la siguiente tabla. Calcular el valor final luego del incremento.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Artículo | Valor actual | % incremento | Incremento | Nuevo valor |
| Camiseta | $25.000 | 3% |  |  |
| Falda | $48.200 | 5% |  |  |
| Corbata | $15.900 | 12% |  |  |
| Zapato | $135.600 | 8% |  |  |
| Pantalón | $130.000 | 10% |  |  |
| Camisa | $56.000 | 3.5% |  |  |
| Sandalias | $75.000 | 4% |  |  |

**Ejercicios de algoritmos Secuenciales**

Hasta este punto hemos desarrollado con claridad: la estructura de secuencia con sus correspondientes instrucciones (asignación, lectura, escritura), la forma general de escribir un algoritmo y demás conceptos que nos permiten resolver un problema. Además, como complemento, vimos la aplicación de los porcentajes, temática que dio claridad en la solución de problemas cotidianos que impliquen ciertos cálculos básicos.

Con todo lo anterior ya abarcado, deberá escribir los algoritmos correspondientes para dar solución a los siguientes planteamientos.

1. A un empleado le retienen el 12% de su salario básico. Calcular el salario neto, sabiendo que le entregan un bono del 2.3% del salario básico. Se debe leer el salario básico del empleado.
2. Elaborar un algoritmo que luego de leer la edad de una persona, muestre la cantidad de meses que ha vivido.
3. Se dice que en el comercio el precio de una pelota de béisbol es el 15% del precio de un balón de fútbol. Escriba un algoritmo que lea el precio de un balón de fútbol y determine el precio de la pelota de béisbol.
4. Una persona recibe una cantidad cualquiera en dólares. Realice un algoritmo que determine el valor equivalente en pesos, sabiendo que el dólar tiene un valor de $2.886.
5. Calcular el salario neto de un trabajador sabiendo que se le debe hacer una retención del 3% para pensión y 5% para salud. El salario se calcula a partir del valor de cada hora y el número de horas trabajadas por el empleado.
6. Escriba un pseudocódigo que luego de leer los siguientes datos de un trabajador (Nombre y Sueldo mensual), genere un comprobante de pago con los siguientes detalles:

\*\*\*\*\*COMPROBANTE DE PAGO\*\*\*\*\*

Nombre: xxxxxxxxxxxxxx

Sueldo: $xxxxx

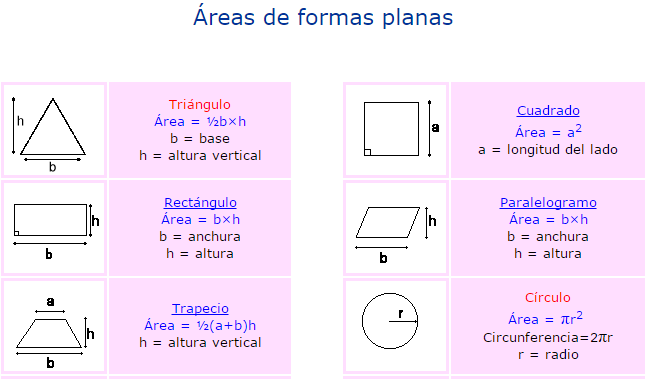
DETALLES DEL PAGO

Aportes a Salud 4%: $xxxxxx

Aportes a Pensión 4%: $xxxxxx

Total a pagar: $xxxxxxxxx

1. Escriba un pseudocódigo que permita calcular el área de las siguientes formas planas a partir del ingreso de las variables identificadas en cada fórmula.



1. Tres personas deciden invertir su dinero para fundar una empresa. Cada una de ellas invierte una cantidad distinta. Obtener el porcentaje que cada quien invierte con respecto a la cantidad total invertida.
2. Un vendedor recibe un sueldo base más un 10% extra por comisión de sus ventas, el vendedor desea saber cuánto dinero obtendrá por concepto de comisiones por las tres ventas que realiza en el mes y el total que recibirá en el mes tomando en cuenta su sueldo base y comisiones.
3. Un alumno desea saber cuál será su calificación final en la materia de Algoritmos. Dicha calificación se compone de los siguientes porcentajes:

55% del promedio de sus tres calificaciones parciales.

30% de la calificación del examen final.

15% de la calificación de un trabajo final.

1. Tres personas deciden invertir su dinero para fundar una empresa. Cada una de ellas invierte una cantidad distinta. Obtener el porcentaje que cada quien invierte con respecto a la cantidad total invertida.
2. Un constructor sabe que necesita 0,5 metros cúbicos de arena por metro cuadrado de revoque a realizar. Hacer un programa donde ingrese las medidas de una pared (largo y alto) expresada en metros y obtenga la cantidad de arena necesaria para revocarla.
3. Elabore un programa que realice la conversión de libras a kilogramos, donde 1 Kg. = 2.2046 libras.

**CAPÍTULO 3: ESTRUCTURAS DE DECISIÓN**

**Introducción**

Los algoritmos que se han elaborado hasta el momento se ejecutan usando únicamente la estructura secuencia, con sus instrucciones elementales: ASIGNACIÓN, LEA y ESCRIBA. Sin embargo, hay situaciones en las cuales es necesario no aplicar exactamente la estructura secuencia, es decir, no ejecutar las instrucciones exactamente en el orden en el cual se han escrito. Para lograr esta funcionalidad se cuenta con las instrucciones correspondientes a la estructura decisión.

**Objetivos**

1. Reconocer la estructura decisión.
2. Identificar cuándo y cómo utilizar decisiones.

Preguntas básicas

1. ¿En qué consiste la estructura decisión?
2. ¿Cuáles son las instrucciones correspondientes a la estructura decisión?
3. ¿Cuál es la forma general de la instrucción SI?

Contenidos

1. Definición de la estructura decisión
2. Instrucción SI, forma general
3. Ejemplo de algoritmo con la instrucción SI

**Definición de la estructura decisión**

La estructura decisión permite instruir al computador para que ejecute ciertas acciones (instrucciones) según alguna condición.

Las instrucciones que pertenecen a la estructura decisión son:

1. La instrucción SI y su componente opcional DE\_LO\_CONTRARIO.
2. La instrucción CASOS.

**Instrucción SI, forma general**

La forma general de la instrucción SI es:

SI condición

Instrucciones que se ejecutan cuando la condición sea verdadera.

DE\_LO\_CONTRARIO

Instrucciones que se ejecutan cuando la condición es falsa.

Fin(SI)

**Ejemplo de algoritmo con la instrucción SI**

Elabore un algoritmo que lea el salario actual de un empleado y que calcule e imprima el nuevo salario de acuerdo a la siguiente condición: si el salario es menor que 1000 pesos, aumentar el 10%; de lo contrario, no hacer aumento.

Análisis:

Datos de entrada:

Salario actual: salact

Datos de salida:

Aumento: au

Nuevo salario: nuesal

Cálculos:

Determinar el aumento según la condición planteada.

Nuestro algoritmo es:

1. Algoritmo aumentoConCondicion(1)

2. Variables: salact, au, nuesal: numericaReal

3. INICIO

4. LEA(salact)

5. SI salact < 1000

6. au = salact \* 0.1

7. DE\_LO\_CONTRARIO

8. au = 0

9. Fin(SI)

10. nuesal = salact + au

11. ESCRIBA(salact, au, nuesal)

12 FIN

13. Fin(aumentoConCondicion)

En la instrucción 2 se definen las variables con las cuales vamos a trabajar.

En la instrucción 4 se lee el salario actual.

En la instrucción 5 se compara el salario leído con el dato de referencia planteado en el enunciado.

Si la condición de la instrucción es verdadera, se ejecuta la instrucción 6; de lo contrario se efectuará la instrucción 8.

En la instrucción 6 se determina el aumento, el cual es el diez por ciento del salario actual, mientras que en la instrucción que en la instrucción 8 se asigna cero al aumento.

La instrucción 9 delimita el alcance de la instrucción SI.

En la instrucción 10 se calcula el nuevo salario, y en la instrucción 11 se escriben los datos leídos y los resultados pedidos.

El anterior algoritmo se puede escribir sin utilizar la componente DE\_LO\_CONTRARIO, la cual como habíamos dicho, es opcional.

Veamos nuestro nuevo algoritmo:

1. Algoritmo aumentoConCondicion(2)

2. Variables: salact, au, nuesal: numericaReal

3. INICIO

4. LEA(salact)

5. au=0

6. SI salact < 1000

7. au = salact \* 0.1

8. Fin(SI)

9. nuesal = salact + au

10. ESCRIBA(salact, au, nuesal)

11 FIN

12. Fin(aumentoConCondicion)

La diferencia de este segundo algoritmo con el primero es que al aumento inicialmente se le asigna el valor de cero en la instrucción 5.

Cuando se compara el salario actual con el valor de referencia (1000), se modificará el aumento sólo si el salario actual es menor que el valor de referencia; de lo contrario el aumento permanece en cero.

**Ejercicios propuestos**

1. Elabore un algoritmo que lea un número entero y que produzca el mensaje acerca de si el entero leído es par o impar.
2. Elabore un algoritmo que lea el nombre de una persona, su estatura y su peso. El algoritmo debe imprimir los datos leídos y un mensaje acerca de si la persona es obesa o no. Una persona se considera obesa si la relación estatura-peso es menor o igual que 2. Su programa debe leer la estatura en metros y el peso en kilogramos; además, la relación estatura-peso se calcula en centímetros sobre kilogramos.
3. Elabore un algoritmo que lea tres datos numéricos enteros y que determine si con esos tres datos se puede construir un triángulo equilátero. Su algoritmo debe imprimir: ‘sí se puede’ o ‘no se puede’.
4. Elabore un algoritmo que lea dos números enteros, efectúe el producto y la suma de ellos, y luego imprima el resultado mayor entre la suma y el producto, con un mensaje apropiado.
5. Elabore un algoritmo que lea tres enteros positivos y que determine si pueden formar triángulo o no. Si pueden formar triángulo debe imprimir qué clase de triángulo es: equilátero, isósceles o escaleno. Tres enteros forman triángulo si cada uno de ellos es menor que la suma de los otros dos.

**Componente DE\_LO\_CONTRARIO**

**Introducción**

En el punto anterior se introdujo el uso de la instrucción condicional SI y se planteó que tiene una parte opcional que se denominó DE\_LO\_CONTRARIO. Cuando se elabora un algoritmo, es muy común que se presenten situaciones en las cuales las condiciones se pueden plantear así: utilizando la instrucción SI con su componente DE\_LO\_CONTRARIO, utilizando únicamente la instrucción SI sin su componente DE\_LO\_CONTRARIO o utilizando necesariamente la instrucción SI con su componente DE\_LO\_CONTRARIO.

Aquí se tratarán ejemplos en los cuales es necesario utilizar la parte opcional DE\_LO\_CONTRARIO de la instrucción SI.

**Objetivo**

Reconocer cuándo es necesario el uso de la opción DE\_LO\_CONTRARIO.

**Preguntas básicas**

1. ¿Cuándo es necesario utilizar la componente DE\_LO\_CONTRARIO en una instrucción SI?
2. ¿Cuándo se puede obviar la componente DE\_LO\_CONTRARIO en una instrucción SI?

**Contenidos**

1. Ejemplo 1
2. Ejemplo 2

Ejemplo 1

Elabore un algoritmo que lea un dato d y una característica, la cual puede ser 1 o 2. Si la característica leída es un 1, el dato leído corresponde al radio de un círculo; si la característica leída es un 2, el dato leído corresponde al lado de un cuadrado. El algoritmo debe calcular e imprimir el área del círculo o del cuadrado, con su correspondiente mensaje: círculo o cuadrado.

Análisis:

Datos de entrada:

Característica: c

Dato: d

Datos de salida:

Área de la figura: area

Cálculos:

Determinar el área de la figura según la característica leída.

Nuestro algoritmo es:

1. Algoritmo areaFiguraConMensaje

2. Variables: d: numericaEntera

3. area: numericaReal

4. INICIO

5. LEA(c, d)

6. SI c == 1

7. area = 3.14 \* d ^ 2

8. ESCRIBA(“área del círculo: “, area)

9. DE\_LO\_CONTRARIO

10. area = d ^ 2

11. ESCRIBA(“área del cuadrado: “, area)

12. Fin(SI)

13. FIN

14. Fin(areaFiguraConMensaje)

En las instrucciones 2 y 3 definimos las variables con las cuales trabajará el algoritmo.

En la instrucción 5 se leen los datos de trabajo.

En la instrucción 6 comparamos la característica leída con 1. Si la característica es 1 se ejecutan las instrucciones 7 y 8; si no, se ejecutan las instrucciones 10 y 11.

Si los datos leídos son 1 y 10, entonces c queda valiendo 1 y d queda valiendo 10. Al ejecutar la instrucción 6 la condición es verdadera y se ejecutan las instrucciones 7 y 8, es decir, la variable area queda valiendo 314.0 e imprime:

área del círculo: 314

Si los datos leídos hubieran sido 2 y 10, entonces c queda valiendo 2 y d queda valiendo 10. Al ejecutar la instrucción 6 la condición es falsa y por lo tanto ejecuta las instrucciones correspondientes a la parte DE\_LO\_CONTRARIO, o sea que la variable area queda valiendo 100 e imprime:

área del cuadrado: 100

Ejemplo 2

Elabore un algoritmo que lea dos datos numéricos enteros y que los imprima ordenados ascendentemente.

Análisis:

Datos de entrada:

Dos datos enteros: a, b

Datos de salida:

Los mismos de entrada

Cálculos:

Identificar cuál es mayor para imprimirlo de primero.

1. Algoritmo dosDatosOrdenados

2. Variables: a, b: numericaEntera

3. INICIO

4. LEA(a, b)

5. SI a < b

6. ESCRIBA (a, b)

7. DE\_LO\_CONTRARIO

8. ESCRIBA (b, a)

9. Fin(SI)

10. FIN

11. Fin(dosDatosOrdenados)

En la instrucción 2 se definen las variables con las cuales vamos a trabajar.

En la instrucción 4 se leen dichos valores.

En la instrucción 5 se comparan los datos leídos. Si el contenido de la variable **a** es menor que el contenido de la **b**, ejecuta la instrucción 6, es decir, escribe primero el contenido de la variable **a** y luego el contenido de la **b**; si no, ejecuta la instrucción 8, es decir, escribe primero el contenido de la variable **b** y luego el contenido de la **a**.

Con base en lo anterior, si los datos entrados son 3 y 6 nuestro algoritmo escribe:

3 6

Si los datos entrados hubieran sido 8 y 2 nuestro algoritmo escribe:

2 8

**Ejercicios propuestos**

1. Elabore un algoritmo que lea un número entero y que determine e imprima el mensaje apropiado si el número leído es impar y menor que 100.
2. Elabore un algoritmo que lea un número entero y que produzca como salida el número leído con el mensaje ‘es positivo’ o ‘es negativo’ según el caso.
3. Elabore un algoritmo que lea el nombre de un estudiante y las tres notas obtenidas en los exámenes de una materia. El algoritmo debe calcular la definitiva y producir un mensaje de felicitación si la materia fue aprobada, o un mensaje de reproche si la materia fue reprobada. La materia se aprueba si obtiene como definitiva una nota mayor o igual que 3.0.
4. Una agencia automotriz ofrece planes de crédito para la adquisición de los diferentes modelos de automóviles. Si el automóvil vale más de $25.000.000 la agencia solicita un 35% de cuota inicial y el resto debe cubrirse en 24 mensualidades sin intereses. Si el automóvil tiene un valor máximo de $25.000.000 se requiere una cuota inicial del 25% y el resto se debe pagar en 18 mensualidades sin intereses. La agencia desea saber, con base en el precio de un automóvil el valor de la cuota inicial y el valor de cada cuota.
5. Escriba un programa que, al recibir como dato el precio de un producto, incremente 11% del mismo si es inferior a $150.000 y que además escriba el nuevo precio del producto.
6. En un museo los visitantes mayores de edad reciben como obsequio un libro; en caso contrario, reciben un lapicero. Dada la edad del visitante, diseñe un algoritmo que determine si el visitante es mayor de edad o menor de edad y que regalo le corresponde.
7. Dado un partido de fútbol jugado entre dos equipos A y B, diseñe un algoritmo que determine el resultado del partido diciendo si *ganó A*, *ganó B* o *hubo empate*.
8. Elaborar un algoritmo que calcule el valor de R de acuerdo a la siguiente relación:

R = (A \* B) / (C \* D) Si X \* Y > 0

R = (A + B) / (C + D) Si X \* Y = 0

R = (A + B) – C + D Si X \* Y < 0

1. Le han contratado para elaborar un programa que le permita a un profesor de Poesía y Literatura Moderna, saber si un estudiante aprobó o no la materia. El estudiante aprueba si su nota es superior a 2.95, en caso contrario reprueba. La forma de calcular la nota definitiva es:
   * Promedio de los cuatro ensayos que se deben elaborar \* 40%
   * Promedio de las cinco poesías que se elaboran a lo largo del curso \* 50%
   * Nota única por la crítica hecha a las obras entregadas \* 10%
2. Una empresa de bienes raíces ofrece casas de interés social, bajo las siguientes condiciones: Si los ingresos del comprador son menores de $8000, la cuota inicial será del 15% del costo de la casa y el resto se distribuirá en pagos mensuales, a pagar en diez años. Si los ingresos del comprador son de $8000 o más, la cuota inicial será del 30% del costo de la casa y el resto se distribuirá en pagos mensuales a pagar en 7 años. La empresa quiere obtener cuanto debe pagar un comprador por concepto de cuota inicial y cuanto por cada pago parcial o cuotas mensuales.
3. En una panadería se hace un descuento del 20% a los clientes cuya compra supere los $10.000. ¿Cuál será la cantidad que debe pagar una persona por su compra?
4. Consulte en la fuente que prefiera la manera de saber la cantidad de cifras que tiene un número cualquiera (el 7 es de una cifra, el 23 de dos cifras, el 231 de tres, etc). Luego, escriba un algoritmo que una vez leída una cantidad numérica, diga la cantidad de cifras que tiene. Máximo hasta cinco cifras, es decir, entre 0 y 99999.
5. Escriba un algoritmo que lea dos números enteros y que verifique si el primero es divisible por el segundo. Ejemplo, al ingresar 36 y 4, debe escribirse “36 es divisible por 4”.
6. Escribir un algoritmo que calcule el total a pagar por la compra de camisas. Si se compran tres camisas o más se aplica un descuento del 20% sobre el total de la compra, y si son menos de tres camisas, un descuento del 10%.
7. Calcular el número de pulsaciones que debe tener una persona por cada 10 segundos de ejercicio aeróbico; la fórmula que se aplica dependiendo del sexo de la persona, así:
   * Número de pulsaciones = (220 - edad) / 10 cuando es mujer
   * Número de pulsaciones = (210 - edad) / 10 cuando es hombre
8. En un almacén se efectúa una promoción en la cual se hace un descuento sobre el valor de la compra total según el color de la bolita que el cliente saque en el momento de pagar en la caja. Si la bolita es de color blanco, no se le hará descuento, si es de color verde, se le hará un descuento del 10%, si es amarilla un 25%, si es azul un 50% y si es roja un 100%. Determinar la cantidad final que el cliente deberá pagar por su compra. Solo hay bolitas de los colores mencionados.
9. En un juego de preguntas a las que se responde “Si” o “No” gana quien responda correctamente las tres preguntas. Si se responde mal a cualquiera de ellas ya no se pregunta la siguiente y termina el juego. Las preguntas son:
   * ¿Bolívar libertó a Colombia?
   * ¿Colón descubrió a América?
   * ¿J Balvin es un artista colombiano?

**INTRODUCCIÓN AL LENGUAJE DE PROGRAMACIÓN JAVA**

*Antes de iniciar este apartado, donde nos dedicaremos a aspectos básicos del lenguaje, es necesario aclarar que este documento no pretende ser un curso de Java, sino solo la codificación de los algoritmos usando: Estructuras de secuencia, Estructuras de decisión y Estructura ciclo.*

Java es un lenguaje de programación de propósitos generales. Podemos usar Java para desarrollar el mismo tipo de aplicaciones que programamos con otros lenguajes como C o Pascal.

La tecnología Java que permite construir este tipo de aplicaciones está basada en el desarrollo de Servlets, pero esto es parte de lo que se conoce como JEE (Java Enterprise Edition). JEE es un conjunto de bibliotecas que permiten desarrollar “aplicaciones empresariales” con Java. Es decir que para programar con JEE primero debemos conocer el lenguaje de programación Java.

**El Entorno Integrado de Desarrollo (IDE)**

Si bien podemos editar nuestro código utilizando cualquier editor de texto y luego compilarlo en línea de comando, lo recomendable es utilizar una herramienta que nos ayude en todo el proceso de programación.

Una IDE (Integrated Development Environment) es una herramienta que permite editar programas, compilarlos, depurarlos, documentarlos, ejecutarlos, etc.

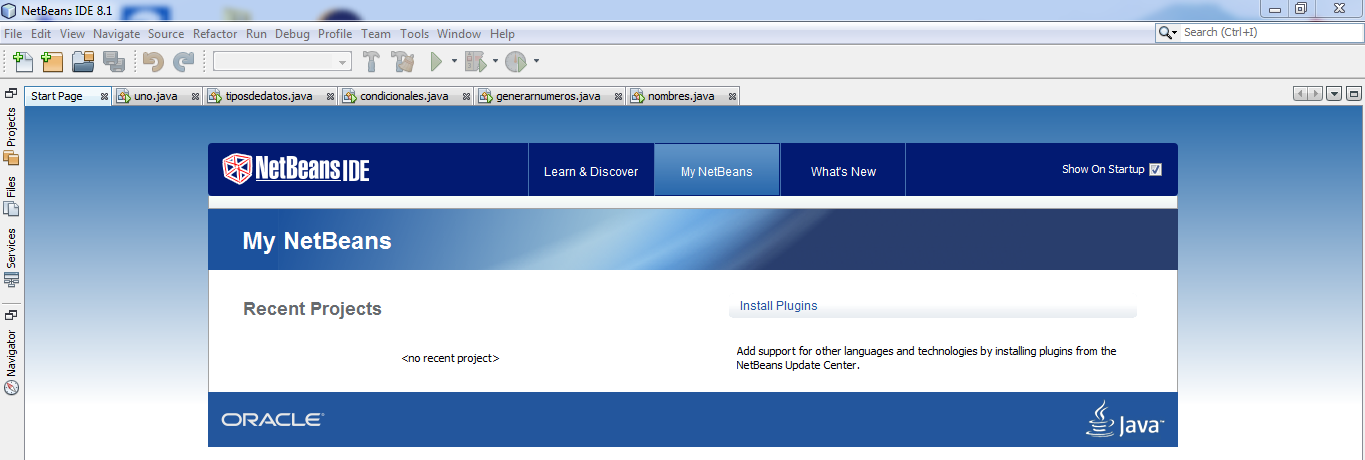
Para trabajar con Java existen en el mercado diferentes IDE. Algunas son de código abierto (open source) como Eclipse y NetBeans y otras son pagas como JBuilder (de Borland), JDeveloper (de Oracle), WebPhere (de IBM), etcétera.

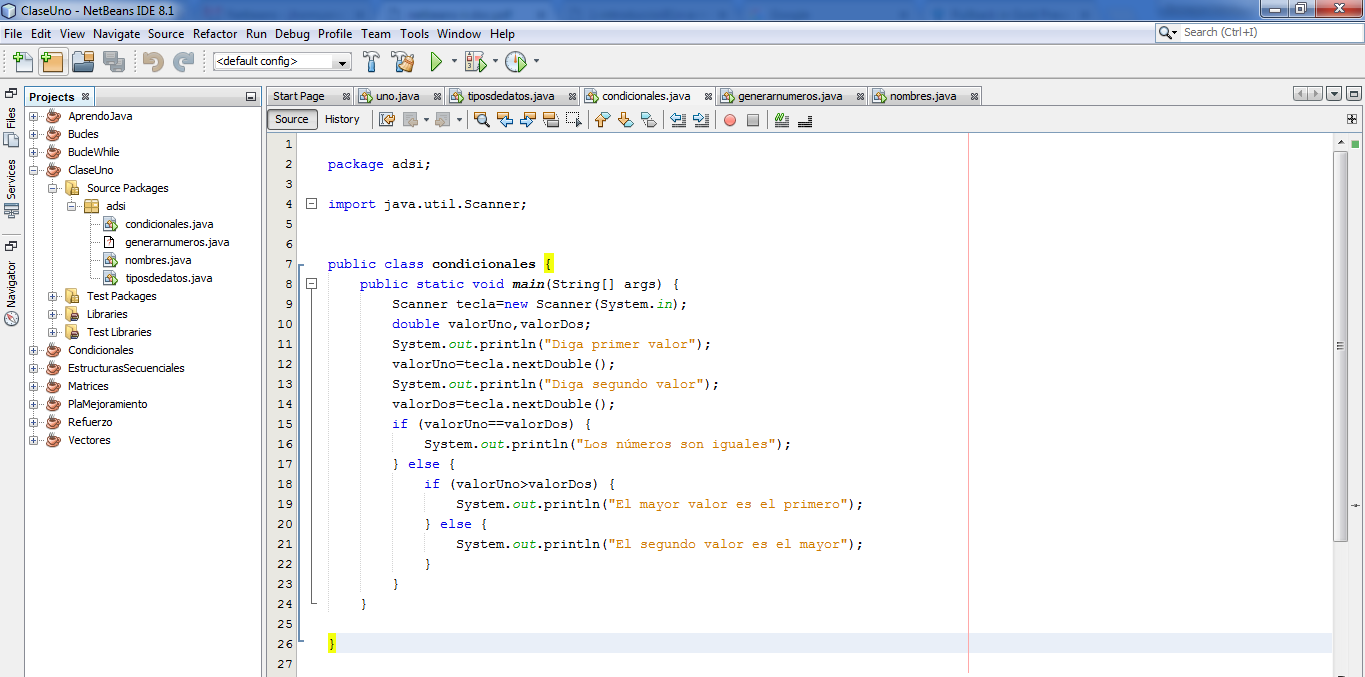
Durante este trimestre, estaremos usando Netbeans como IDE, pues en él podemos realizar todas las tareas asociadas a la programación:

* Editar el código
* Compilarlo
* Ejecutarlo
* Depurarlo

**La ventana estándar**

Cada vez que inicie Netbeans, posiblemente vea algo así:





Código fuente

Paquetes

Archivos abiertos

Proyectos

**Projects**: permite tener a mano todos los proyectos en que uno está trabajando actualmente o en los que ha trabajado en el último tiempo. A través de la expansión o colapso de los nodos en el árbol uno puede ir navegando a través de los elementos que conforman el proyecto. Sugiero hacer clic derecho sobre el proyecto y revisar la ventana de propiedades.

**Files**: permite navegar a través de las carpetas del proyecto. A veces algunos elementos pueden no aparecer en el árbol del proyecto en “Proyects”, pero si en “Files”. Por ejemplo, los archivos de salida al ejecutar o debugear un proyecto.

**Services**: maneja conexiones a algunos servicios. Los más comunes son los de Bases de datos. Son una ayuda sumamente útil si uno realiza proyectos que interactúan con servidores de base de datos.

**Navigator**: despliega los miembros de una clase. Se sugiere revisar su funcionamiento ya que puede ser de suma utilidad. Permite desplegar o esconder los miembros heredados de una clase, desplegar o esconder los atributos de una clase, ir hasta la definición de un miembro (haciendo doble clic sobre este), encontrar ocurrencias del elemento (find usages), etc.

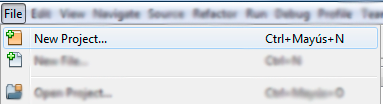
La mayoría de los entornos de desarrollo integrados (IDEs) para Java, hacen uso de un concepto de proyecto. **Un proyecto** en un IDE cualquiera, es un "contenedor" global donde podemos incluir o crear todos los archivos que de una u otra forma vayan a ser utilizados en nuestro programa, clases, interfaces, archivos de texto, imágenes, paquetes, etc.

Un proyecto en un entorno de desarrollo nos permite dar un orden y una clasificación a nuestro trabajo y a nuestros programas, evitando así que estemos desarrollando algo con los archivos de diferentes programas mezclados en un mismo lugar. Los proyectos nos permiten mantener ordenado nuestro espacio de trabajo permitiéndonos ser más ágiles y evitar errores.

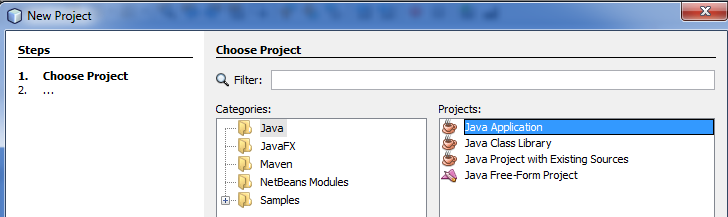
Vamos a ver rápidamente lo fácil que es crear un proyecto en Netbeans. Veremos algunas imágenes y una guía para crear un proyecto en este entorno de desarrollo. Comencemos:

Una vez haya iniciado correctamente Netbeans, podremos crear el nuevo proyecto.

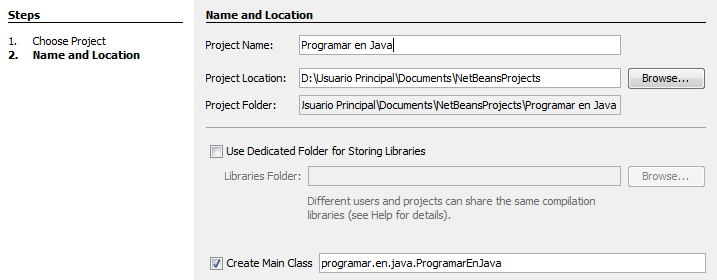
Para crear el proyecto, debemos seguir unos pasos bastante simples. En nuestra ventana principal en la parte superior tenemos una serie de opciones, debemos dar clic en: File » New Project..., tal como muestra la imagen.



Hecho esto, se nos mostrará una nueva ventana donde podremos seleccionar el tipo de proyecto a crear. Tenemos varias opciones, de éstas seleccionaremos Java » Java Application y damos en Next>



Luego se abrirá una ventana pidiéndonos alguna información de nuestro proyecto. Para este ejemplo, como nombre al proyecto le pondremos **Programar en Java**. La ubicación la podemos cambiar dando clic en examinar, sin embargo, es recomendable dejarla en la ubicación por defecto por comodidad y facilidad. Luego, deshabilitamos la opción Create Main Class, y finalmente damos clic en Finish. Así nuestra ventana quedaría como la imagen.



Deshabilitar

**Ordenando nuestro proyecto Java. Uso de los paquetes en Java (packages)**

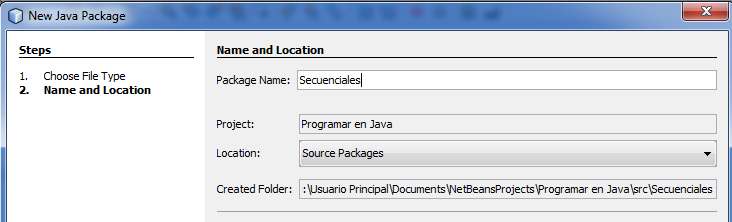
Los paquetes en Java (packages) son la forma en la que Java nos permite agrupar de alguna manera lógica los componentes de nuestra aplicación que estén relacionados entre sí.

Los paquetes permiten poner en su interior casi cualquier cosa como: clases, interfaces, archivos de texto, entre otros. De este modo, los paquetes en Java ayudan a darle una buena organización a la aplicación ya que permiten modularizar o categorizar las diferentes estructuras que componen nuestro software.

Los paquetes en Java, adicionalmente al orden que nos permite darle a la aplicación, también nos brindan un nivel adicional de seguridad para nuestras clases, métodos o interfaces, pues como veremos más adelante podremos especificar si una clase o interfaz en particular es accesible por todos los componentes del software (sin importar el paquete) o si en realidad es solo accesible por las clases que estén en el mismo paquete que ésta. Veremos con más detalle este tema del acceso más adelante.

Ahora veamos cómo crear paquetes en Java

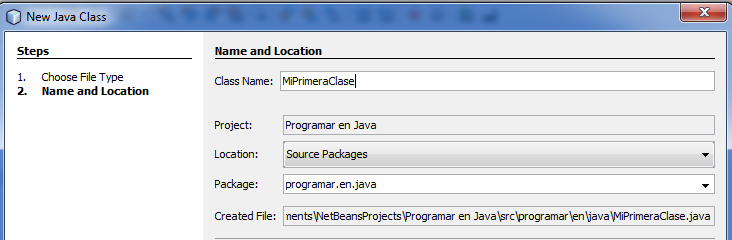
Damos clic con el botón sobre la carpeta Source Packages, que está dentro del proyecto previamente creado, como se muestra en la siguiente imagen. Observe que aparece el nombre del proyecto, indicando que el paquete será creado dentro del mismo. Terminamos con Finish.



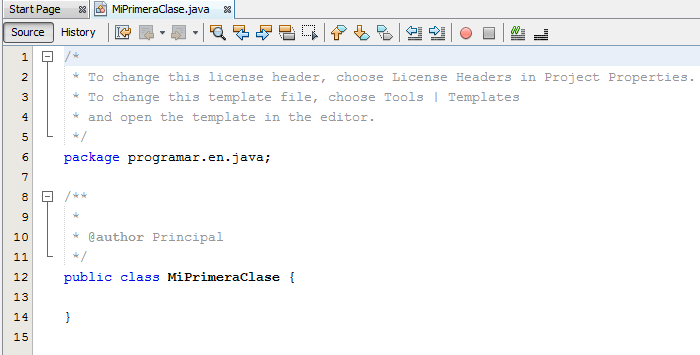
Hecho esto, podremos añadir elementos a nuestro proyecto: clases, interfaces, etc. Vamos a ver cómo crear una clase al interior de nuestro proyecto:

Hacemos clic derecho sobre el proyecto, luego en **New** y finalmente en **Java Class**...

Aparecerá una ventana pidiéndonos el nombre para la clase, le pondremos **MiPrimeraClase**. Luego, en Package, seleccionaremos el paquete previamente creado con el nombre de **programar.en.java**. Finalizamos con Finish.



Como resultado del proceso anterior, tendremos lo siguiente:

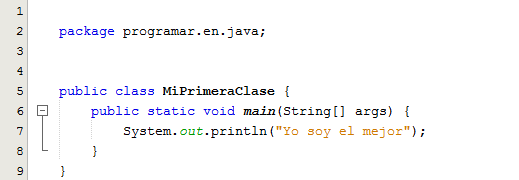


**Ejemplo simple de un programa en Java.**

Vamos a escribir nuestro primer programa en el lenguaje Java, y así aprenderemos lo que trataremos en el siguiente punto (**Resolución de problemas con Java**).

*Escriba un programa en Java que imprima el mensaje* ***Yo soy el mejor****.*

La codificación es la siguiente:



**Resolución de problemas con Java**

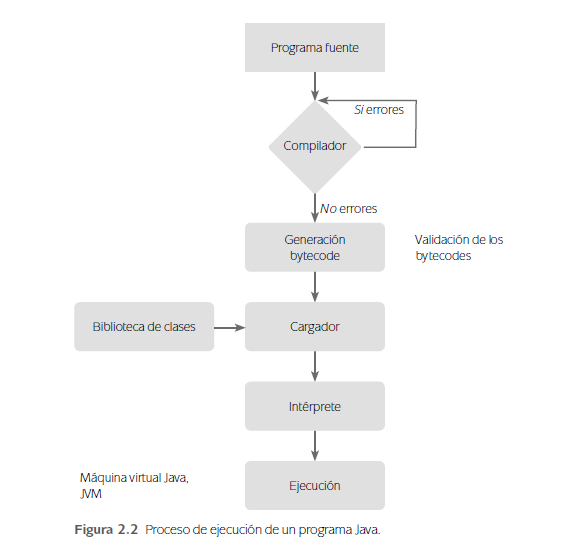
El proceso para resolver problemas con una computadora implica la escritura de un programa y su posterior ejecución; así, la programación es un proceso de resolución de problemas. Existen diferentes técnicas para resolverlos, y aunque el proceso de diseñar y construir programas es esencialmente creativo, se pueden considerar diferentes etapas en el proceso de programación. Las fases de resolución de un problema y sus características más destacadas son:

* **Análisis**. El problema se examina considerando la especificación de los requisitos dados por el cliente, respecto al programa.
* **Diseño** del algoritmo. Una vez que se analiza el problema, se diseña una solución que conduzca a un algoritmo (método) que lo resuelva.
* **Codificación (implementación)**. La solución se escribe en la sintaxis de algún lenguaje de alto nivel, en este caso Java, y se obtiene un programa fuente que a continuación se compilará.
* **Compilación y ejecución**. El programa se agrupa, en el caso de Java se interpreta, y ejecuta.
* **Verificación y depuración**. El programa se comprueba rigurosamente y se eliminan todos los errores que aparezcan; a dichas erratas se les denomina bugs.
* **Mantenimiento**. El programa se actualiza y se modifica cada vez que sea necesario para que se cumplan todas las necesidades de los usuarios; en esta fase se utilizan y mejoran los algoritmos realizando los cambios si los requisitos así lo exigen.
* **Documentación**. Escritura de las diferentes fases del ciclo de vida del software, esencialmente el análisis, diseño y codificación; junto con manuales de usuario y de referencia, así como normas para el mantenimiento.

Al ejecutar el programa se visualizará en pantalla:

**Yo soy el mejor**

Esto se debe a que la instrucción o sentencia **println** envía la frase contenida entre comillas a la pantalla. El proceso de ejecución del programa anterior comprende las siguientes etapas, ilustradas en la figura 2.2.



**El método main**

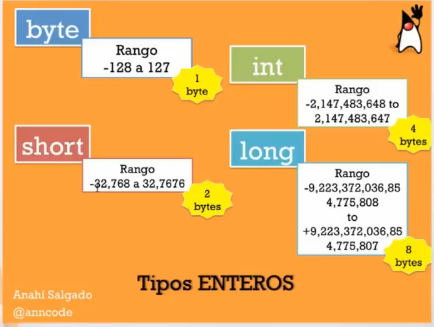


El método **main** es el punto de inicio de toda aplicación en Java. Los paréntesis después del identificador main indican que éste es un bloque de construcción del programa, el cual se le llama método. Las declaraciones de clases en Java generalmente contienen uno o más métodos. Es una aplicación en Java, sólo uno de esos métodos debe llamarse main, de no ser así, la Máquina Virtual de Java (JVM) no ejecutará la aplicación.

Mientras que la palabra public específica "quién" puede acceder al método main, la palabra static específica "cómo" acceder al método main. Con un método no estático, se debe realizar trabajo extra antes de acceder a este. Por otro lado, a un método de tipo estático se puede acceder sin realizar ningún trabajo extra. Ya que main es el punto de inicio para todos los programas de Java, debe ser inmediatamente accesible y, por tanto, requiere de la palabra static.

La palabra void indica que un método no devuelve nada. Puesto que el método main no devuelve nada, entonces se utiliza void en el encabezado del método main. Las palabras String[] args entre paréntesis son una parte requerida del encabezado del método **main**.

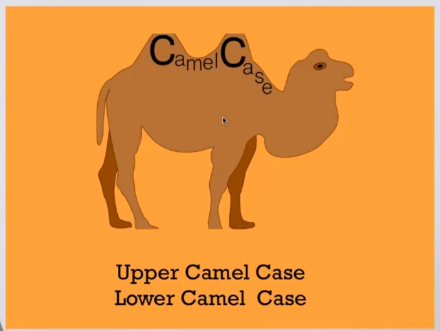
**TIPOS DE DATOS, VARIABLES Y CONSTANTES**

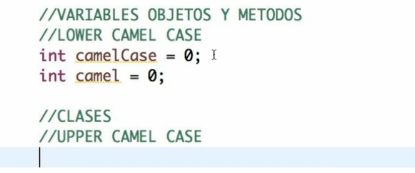


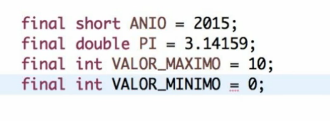












**Tipos de datos en Java (Tipos primitivos)**

Se conocen como tipos de datos primitivos a los tipos de datos que forman parte del núcleo del lenguaje Java. Existen ocho tipos primitivos que se muestran a continuación:

**Enteros**

int: 4 bytes de espacio para almacenamiento. Desde -2.147.483.648 hasta 2.147,483.647

short: 2 bytes de espacio para almacenamiento. Desde -32.768 hasta 32.767

long: 8 bytes de espacio para almacenamiento. Una barbaridad. Sufijo L.

byte: 1 byte de espacio para almacenamiento. Desde -128 hasta 127.

**Coma flotante (decimales)**

**float**: 4 bytes para almacenamiento . Aproximadamente 6 a 7 cifras decimales significativas. Sufijo F.

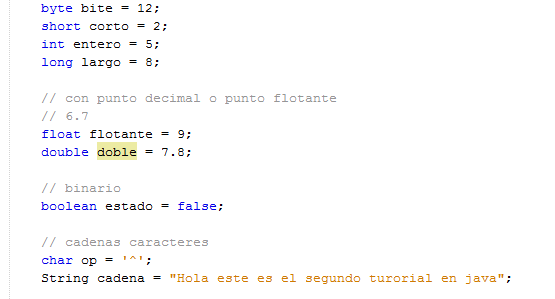
**double**: 8 bytes de espacio para almacenamiento. Aproximadamente 15 cifras decimales significativas.

**Char**: Para representar caracteres.

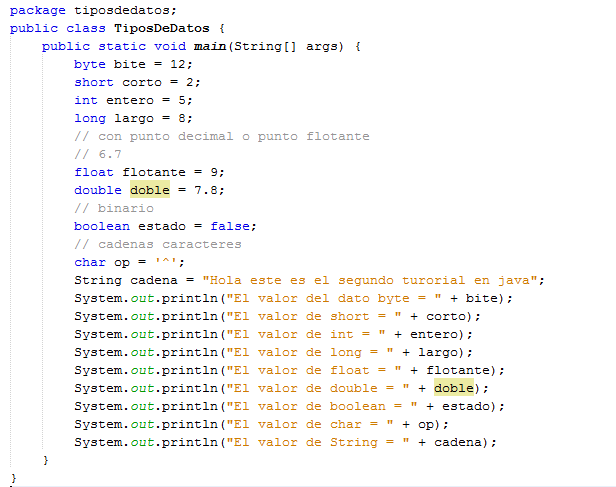
**Boolean**: Para valores lógicos. Valores True o False.

Para el manejo de cadenas de caracteres el lenguaje Java dispone del tipo referencia **String**.

Ahora vamos a ver algunos ejemplos de cómo se definen las variables en Java.



A continuación veamos y analicemos un programa un poco más completo:



**Operadores**

Operador de asignación simple

= Operador de asignación simple

Operadores aritméticos

+ Operador de suma (también se usa para la concatenación de cadenas)

- Operador de resta

\* Operador de multiplicación

/ Operador de división

% Operador de resto o Módulo

Operadores unarios

+ Operador más unario; indica positivo

- Operador menos unario; niega una expresión

++ Operador de aumento; aumenta un valor en 1

- - Operador de disminución; disminuye un valor en 1

! Operador de complemento lógico; invierte el valor de un booleano

**Operadores lógicos**

Operadores de igualdad y relacionales

= = Igual que

!= Distinto de

> Mayor que

< Menor que

>= Mayor o igual que

<= Menor o igual que

Operadores condicionales

&& AND condicional

|| OR condicional

**Entrada con la clase Scanner**

En la versión 5.0, Java incluyó una clase para simplificar la entrada de datos por el teclado llamada Scanner, que se conecta a System.in; para leer la entrada a la consola se debe construir primero un objeto de Scanner pasando el objeto System.in al constructor Scanner.

*Scanner entrada = new Scanner(System.in);*

Una vez creado el objeto Scanner, se pueden utilizar diferentes métodos de su clase para leer la entrada: nextInt o nextDouble leen enteros o de coma flotante.

System.out.print("Introduzca cantidad: ");

int cantidad;

cantidad = entrada.nextInt();

System.out.print("Introduzca precio: ");

double precio = entrada.nextDouble();

Cuando se llama a uno de los métodos anteriores, el programa espera hasta que el usuario teclee un número y pulsa Enter.

El método nextLine lee una línea de entrada:

System.out.print("¿ Cuál es su nombre?");

String nombre;

nombre = entrada.nextLine();

El método next se emplea cuando se desea leer una palabra sin espacios:

String apellido = entrada.next();

La clase Scanner se define en el paquete java.util y siempre que se utiliza una clase no definida en el paquete básico java.lang se necesita utilizar una directiva import.

La primera línea cerca del principio del archivo indica a Java dónde encontrar la definición de la clase Scanner:

*import java.util.Scanner*

Esta línea significa que la clase Scanner está en el paquete java.util; util es la abreviatura de utility (utilidad o utilería), la cual siempre se utiliza en código Java. Como se ha mencionado, un paquete es una biblioteca de clases y la sentencia import hace disponible la clase dentro del programa.

import java.util.Scanner;

/\*\*

Este programa muestra la entrada por consola

y ha sido creado en el año 2017

\*/

public class EntradaTest {

public static void main)String [] args) {

Scanner entrada = new Scanner(System.in);

// obtener la primera entrada

System.out.print("¿ Cual es su nombre?");

String nombre = entrada.nextLine();

// leer la segunda entrada

System.out.print("¿ Cual es su edad?");

int edad = entrada.nextInt();

// visualizar salida

System.out.println("Buenos días " + nombre + "; años " + edad);

}

}

**Sintaxis de entrada de teclado utilizando Scanner**

•Hacer disponible la clase Scanner para utilizarla en su código; incluir la siguiente línea al comienzo del archivo que contiene su programa:

import java.util.Scanner;

•Antes de introducir algo por medio del teclado, se debe crear un objeto de la clase Scanner.

Scanner nombreObjeto = new Scanner(System.in);

nombreObjeto es cualquier identificador Java.

Scanner teclado = new Scanner(System.in);

•Los métodos nextInt, nextDouble y next leen respectivamente un valor de tipo int, un valor de tipo double y una palabra.

**Sintaxis**

variable\_int = nombreObjeto.nextInt();

variable\_double = nombreObjeto.nextDouble();

variable\_cadena = nombreObjeto.next();

variable\_cadena = nombreObjeto.nextLine();

Ejemplo

int edad;

edad = teclado.nextInt();

double precio;

precio = teclado.nextDouble();

String rio;

rio = teclado.next();

**¡Comencemos a programar!**

Vamos a retomar algunos ejemplos planteados al inicio de este documento para realizar su respectiva codificación en Java.

1 Algoritmo Uno

2 Variables: nom: alfanumérica

3 Tel: numericaEntera

4 INICIO

5 ESCRIBA (“Teclee nombre y teléfono”)

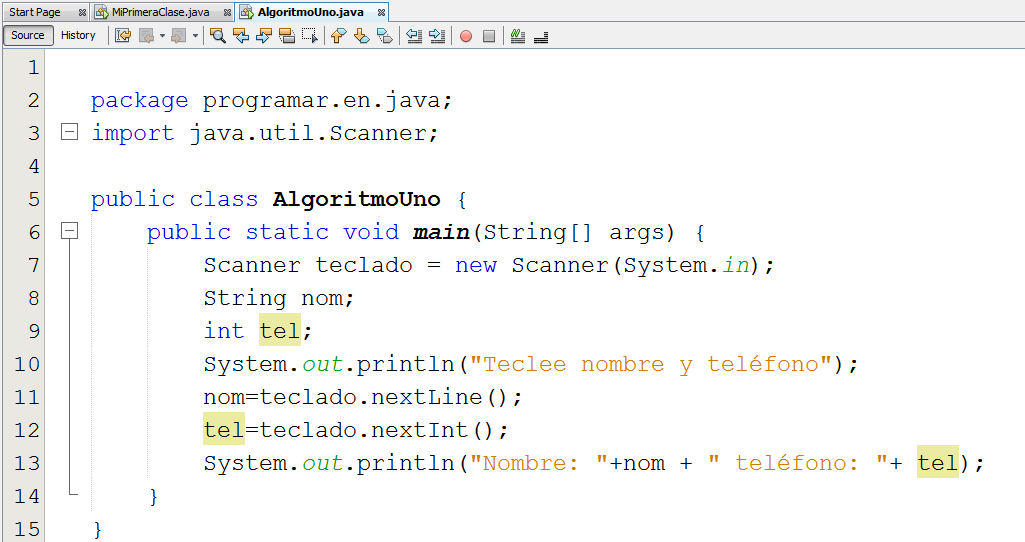
6 LEA (nom, tel)

7 ESCRIBA (“nombre: “, nom, “teléfono: “, tel)

8 FIN

9 Fin (Uno)

Observemos el código en Java:



Ahora le corresponde a usted codificar los siguientes enunciados, a los que previamente ha escrito el pseudocódigo en clases anteriores.

**Parte 1**

1. Elaborar un algoritmo que lea un dato correspondiente al lado de un cuadrado y que calcule e imprima el área de dicho cuadrado.
2. Elaborar un algoritmo que lea el salario actual de un empleado y el porcentaje de aumento y que calcule e imprima el salario actual, el aumento y el nuevo salario.
3. Elabore un algoritmo que lea el nombre de una persona y que imprima el mensaje “Hola” seguido del nombre de la persona leída.
4. Elabore un algoritmo que lea dos datos enteros correspondientes a los catetos de un triángulo y que calcule e imprima el valor de la hipotenusa de dicho triángulo.
5. Elabore un algoritmo que lea el código de un artículo, el precio unitario del artículo y la cantidad vendida. Su algoritmo debe calcular e imprimir el total de la venta, el IVA y el total a pagar, sabiendo que el impuesto es diecinueve por ciento.
6. Elabore un algoritmo que lea dos números enteros y que produzca como salida la suma, resta, multiplicación, división y módulo del primero por el segundo.
7. Elabore un algoritmo que lea una temperatura en grados Fahrenheit y la convierta y la imprima en grados centígrados. Los grados Fahrenheit se convierten a grados centígrados restándoles 32 y multiplicando por cinco novenos.
8. Elabore un algoritmo que lea un dato, el cual es el valor de un ángulo en grados, y que lo convierta e imprima en radianes.

**Parte 2**

1. A un empleado le retienen el 12% de su salario básico. Calcular el salario neto, sabiendo que le entregan un bono del 2.3% del salario básico. Se debe leer el salario básico del empleado.
2. Elaborar un algoritmo que luego de leer la edad de una persona, muestre la cantidad de meses que ha vivido.
3. Se dice que en el comercio el precio de una pelota de béisbol es el 15% del precio de un balón de fútbol. Escriba un algoritmo que lea el precio de un balón de fútbol y determine el precio de la pelota de béisbol.
4. Una persona recibe una cantidad cualquiera en dólares. Realice un algoritmo que determine el valor equivalente en pesos, sabiendo que el dólar tiene un valor de $2.886.
5. Calcular el salario neto de un trabajador sabiendo que se le debe hacer una retención del 3% para pensión y 5% para salud. El salario se calcula a partir del valor de cada hora y el número de horas trabajadas por el empleado.
6. Escriba un pseudocódigo que luego de leer los siguientes datos de un trabajador (Nombre y Sueldo mensual), genere un comprobante de pago con los siguientes detalles:

\*\*\*\*\*COMPROBANTE DE PAGO\*\*\*\*\*

Nombre: xxxxxxxxxxxxxx

Sueldo: $xxxxx

DETALLES DEL PAGO

Aportes a Salud 4%: $xxxxxx

Aportes a Pensión 4%: $xxxxxx

Total a pagar: $xxxxxxxxx

**Parte 3**

1. Elabore un algoritmo que lea el nombre de un estudiante y las tres notas obtenidas en los exámenes de una materia. El algoritmo debe calcular la definitiva y producir un mensaje de felicitación si la materia fue aprobada, o un mensaje de reproche si la materia fue reprobada. La materia se aprueba si obtiene como definitiva una nota mayor o igual que 3.0.
2. Una agencia automotriz ofrece planes de crédito para la adquisición de los diferentes modelos de automóviles. Si el automóvil vale más de $25.000.000 la agencia solicita un 35% de cuota inicial y el resto debe cubrirse en 24 mensualidades sin intereses. Si el automóvil tiene un valor máximo de $25.000.000 se requiere una cuota inicial del 25% y el resto se debe pagar en 18 mensualidades sin intereses. La agencia desea saber, con base en el precio de un automóvil el valor de la cuota inicial y el valor de cada cuota.
3. Escriba un programa que, al recibir como dato el precio de un producto, incremente 11% del mismo si es inferior a $150.000 y que además escriba el nuevo precio del producto.
4. En un museo los visitantes mayores de edad reciben como obsequio un libro; en caso contrario, reciben un lapicero. Dada la edad del visitante, diseñe un algoritmo que determine si el visitante es mayor de edad o menor de edad y que regalo le corresponde.
5. Dado un partido de fútbol jugado entre dos equipos A y B, diseñe un algoritmo que determine el resultado del partido diciendo si *ganó A*, *ganó B* o *hubo empate*.
6. Escriba un algoritmo que lea dos números enteros y que verifique si el primero es divisible por el segundo. Ejemplo, al ingresar 36 y 4, debe escribirse “36 es divisible por 4”.

**Decisiones anidadas**

Introducción

Anteriormente se ha tratado la instrucción SI, con su componente opcional DE\_LO\_CONTRARIO, de una manera simple. En la práctica se presentan hechos en los cuales es necesario controlar situaciones dentro de situaciones ya controladas, es decir, comprobar condiciones dentro de condiciones, o comprobar varias condiciones a la vez. Estos acontecimientos implican el uso de la instrucción SI de una forma más compleja, la cual se tratará en este apartado.

Objetivos

1. Elaborar algoritmos más complejos utilizando la instrucción SI con su componente opcional DE\_LO\_CONTRARIO.

Preguntas básicas

1. Qué es mejor: ¿condición compuesta o varias instrucciones SI anidadas?
2. ¿Cómo afecta la ejecución de un algoritmo el uso, o no, de la opción DE\_LO\_CONTRARIO?

Ejemplo de ilustración

Elabore un algoritmo que lea tres datos numéricos enteros y que los imprima ordenados ascendentemente.

Análisis:

Datos de entrada:

Tres datos numéricos enteros: a, b, c

Datos de salida:

Los mismos tres datos de entrada ordenados ascendentemente.

Cálculos:

Determinar el menor de los tres datos para imprimirlo de primero, luego determinar el menor de los otros dos para imprimirlo de segundo y luego imprimir el tercer dato.

Una primera forma de escribir este algoritmo es siendo exhaustivos en la comparación de los datos. Realmente, las diferentes situaciones que se pueden presentar para escribir los tres datos son:

La primera, cuando a es menor que b y b es menor que c.

La segunda, cuando a es menor que c y c es menor que b.

La tercera, cuando b es menor que a y a es menor que c, y así sucesivamente.

A cada situación le corresponde una relación de orden diferente.

1. a, b, c
2. a, c, b
3. b, a, c
4. b, c, a
5. c, a, b
6. c, b, a

Un algoritmo para efectuar esta tarea es:

1. Algoritmo ordenaTresDatos(1)

2 Variables: a, b, c: numericaEntera

3. INICIO

4. LEA(a, b, c)

5. SI a <= b && b <= c

6. ESCRIBA(a, b, c)

7. Fin(SI)

8. SI a <= c && c <= b

9. ESCRIBA(a, c, b)

10. Fin(SI)

11. SI b <= a && a <= c

12. ESCRIBA(b, a, c)

13. Fin(SI)

14. SI b <= c && c <= a

15. ESCRIBA(b, c, a)

16. Fin(SI)

17. SI c <= a && a <= b

18. ESCRIBA(c, a, b)

19. Fin(SI)

20. SI c <= b && b <= a

21. ESCRIBA(c, b, a)

22. Fin(SI)

23. FIN

24. Fin(ordenaTresDatos)

Las instrucciones 5 a 7 consideran la primera situación; las instrucciones 8 a 10 consideran la segunda; las instrucciones 11 a 13 consideran la tercera; las instrucciones 14 a 16 consideran la cuarta; las instrucciones 17 a 19 consideran la quinta; y las instrucciones 20 a 22 consideran la sexta.

Este algoritmo tiene el inconveniente de que cuando una situación sea verdadera, continúa preguntando por las demás situaciones, lo cual genera ineficiencia. Para evitar esta ineficiencia utilizamos la parte opcional DE\_LO\_CONTRARIO.

Veamos cómo queda nuestro algoritmo:

1. Algoritmo ordenaTresDatos(2)

2 Variables: a, b, c: numericaEntera

3. INICIO

4. LEA(a, b, c)

5. SI a <= b && b <= c

6. ESCRIBA(a, b, c)

7. DE\_LO\_CONTRARIO

8. SI a <= c && c <= b

9. ESCRIBA(a, c, b)

10. DE\_LO\_CONTRARIO

11. SI b <= a && a <= c

12. ESCRIBA(b, a, c)

13. DE\_LO\_CONTRARIO

14. SI b <= c && c <= a

15. ESCRIBA(b, c, a)

16. DE\_LO\_CONTRARIO

17. SI c <= a && a <= b

18. ESCRIBA(c, a, b)

19. DE\_LO\_CONTRARIO

20. ESCRIBA(c, b, a)

21. Fin(SI)

22. Fin(SI)

23. Fin(SI)

24. Fin(SI)

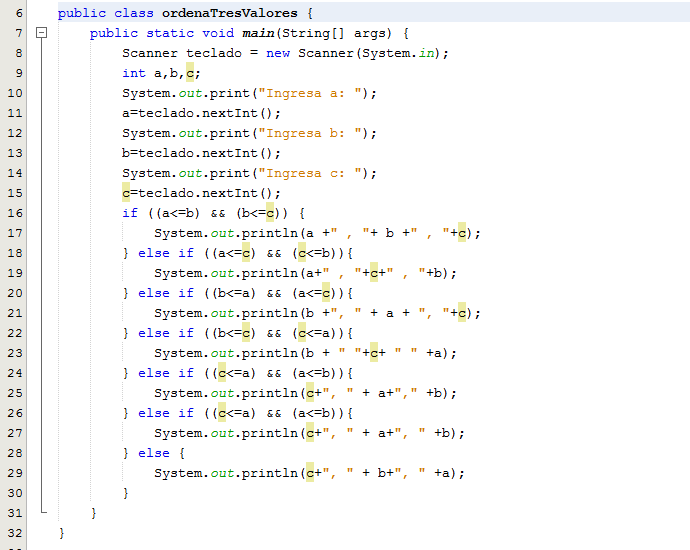
25. Fin(SI)

26. FIN

27. Fin(ordenaTresDatos)

De esta manera, cuando encuentre que una condición (situación) es verdadera, procede a imprimir los datos en forma ordenada y no sigue preguntando por las demás condiciones.

Ahora analicemos la codificación en Java



Una tercera forma en que podemos elaborar el algoritmo es la siguiente:

Comparamos **a** con **b**.

Pueden suceder dos cosas: una, que **a** sea menor que **b**, y dos, que **b** sea menor que **a**.

1. Si a es menor que b, implica que habrá que escribir el dato a antes que el dato b; por lo tanto, las posibles formas de escribir los tres datos son:
2. a, b, c
3. a, c, b
4. c, a, b

Si b es menor que c, imprimimos la primera posibilidad: a, b, c; de lo contrario, debemos comparar a con c. Si a es menor que c, imprimimos la segunda posibilidad: a, c, b; de lo contrario, imprimimos la tercera posibilidad: c, a, b.

1. Si b es menor que a implica que habrá que escribir el dato b antes que el dato a; por tanto las posibles formas de escribir los tres datos son:
2. b, a, c
3. b, c, a
4. c, b, a

Si a es menor que c, imprimimos la primera posibilidad: b, a, c; de lo contrario, debemos comparar b con c.

Si b es menor que c, imprimimos la segunda posibilidad: b, c, a; de lo contrario, imprimimos la tercera posibilidad: c, b, a.

Con base en el anterior análisis nuestro algoritmo queda:

1. Algoritmo ordenaTresDatos(3)

2. Variables: a, b, c: numericaEntera

3. INICIO

4. LEA(a, b, c)

5. SI a <= b

6. SI b <= c

7. ESCRIBA(a, b ,c)

8. DE\_LO\_CONTRARIO // c es menor que b

9. SI a <= c

10. ESCRIBA(a, c, b)

11. DE\_LO\_CONTRARIO // c es menor que a

12. ESCRIBA(c, a, b)

13. Fin(SI)

14. Fin(SI)

15. DE\_LO\_CONTRARIO // b es menor que a

16. SI a <= c

17. ESCRIBA(b, a, c)

18. DE\_LO\_CONTRARIO // c es menor que a

19. SI b <= c

20. ESCRIBA(b, c, a)

21. DE\_LO\_CONTRARIO / c es mayor que b

22. ESCRIBA (c, b, a)

23. Fin(SI)

24. Fin(SI)

25. Fin(SI)

26. FIN

27. Fin(ordenaTresDatos)

Veamos cómo se comportaría este último algoritmo con algunos datos de entrada:

a b c

1º 3 1 6

2º 5 6 2

3º 7 6 5

4º 1 2 3

Al ejecutar la instrucción de lectura de los datos de la primera línea, los datos almacenados en las variables **a**, **b** y **c** son 3, 1 y 6, respectivamente.

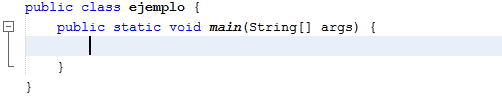
El resultado de ejecutar la instrucción 5 (comparar **a** con **b**) es falso; por lo tanto la ejecución continúa en la instrucción 16, en la cual compara **a** con **c**, obteniendo un resultado verdadero y, como consecuencia, ejecuta la instrucción 17, en la cual escribe **b**, **a**, **c**, lo cual es 1, 3, 6.

Si se leyeran los datos de la segunda línea, los datos almacenados en las variables **a**, **b** y **c** son 5, 6 y 2.

El resultado de ejecutar la instrucción 5 es verdad; por lo tanto, continúa ejecutando la instrucción 6 (compara **b** con **c**), la cual produce un resultado falso, lo que implica que la siguiente instrucción que se ejecuta es la instrucción 9 (compara **a** con **c**), la cual produce un resultado falso y por consiguiente ejecutará la instrucción 12: escribe **c**, **a**, **b**; 2, 5, 6.

Por favor, analizar el comportamiento del algoritmo con los datos de las líneas tercera y cuarta.

Escriba la codificación del anterior algoritmo en lenguaje Java, y compárelo con el anterior ejemplo.



Resumen

Hemos tratado cómo se pueden elaborar algoritmos más complejos utilizando la instrucción SI en forma anidada, es decir, instrucciones de decisión dentro de instrucciones de decisión.

**Ejercicios propuestos**

Para cada uno de los enunciados se recomienda realizar: Análisis, Pseudocódigo, Prueba de escritorio y Codificación en Java.

1. Escribir un algoritmo que pida por teclado tres notas de un alumno, calcule el promedio e imprima alguno de estos mensajes:

Si el promedio es >=7 mostrar "Promocionado".

Si el promedio es >=4 y <7 mostrar "Regular".

Si el promedio es <4 mostrar "Reprobado".

1. En una llantería se ha establecido una promoción de las llantas marca “Ponchadas”, dicha promoción consiste en lo siguiente:

Si se compran menos de cinco llantas el precio es de $30.000 cada una, de $25.000 si se compran de cinco a 10 y de $20.000 si se compran más de 10.

Obtener la cantidad de dinero que una persona tiene que pagar por cada una de las llantas que compra y la que tiene que pagar por el total de la compra.

1. Construya un algoritmo que capture por teclado la estatura de una persona y, dependiendo del valor ingresado, imprima los siguientes resultados:

* Si la estatura es igual o menor a 1,50 m imprima "Su estatura es baja"
* Si la estatura es mayor que 1,50 m pero menor o igual que 1,90 m imprima "Su estatura es media"
* Si la estatura es mayor a 1,90 m imprima "Su estatura es alta"

1. En una tienda se ofrecen descuentos a los clientes en época de navidad, de acuerdo con el monto total de su compra. El criterio para establecer el descuento se muestra abajo. Escriba un algoritmo, tal que, al recibir como dato el monto de la compra del cliente, obtenga el precio real que debe pagar luego de aplicar el descuento correspondiente.

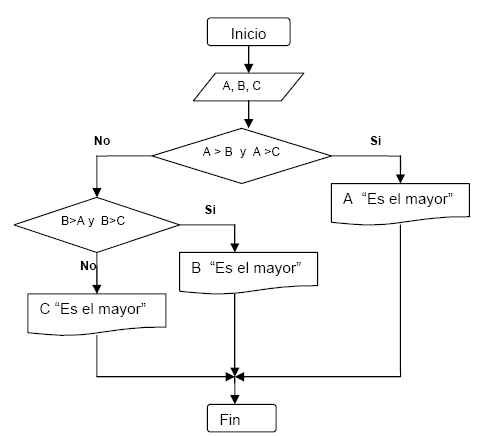
Compra <$80.000 Descuento de 0%

Compra entre $80.000 y $150.000 Descuento 10%

Compras mayores a $150.000 y menor o igual a $500.000 Descuento 15%

Compras mayores a $500.000 Descuento 20%

1. Escriba el correspondiente algoritmo para el siguiente diagrama de flujo:



1. Escribir un algoritmo que pida por teclado tres notas de un estudiante, calcule el promedio e imprima alguno de estos mensajes:

Si el promedio es >=7 mostrar "Promocionado".

Si el promedio es >=4 y <7 mostrar "Regular".

Si el promedio es <4 mostrar "Reprobado".

1. Una frutería ofrece las manzanas con descuento según la siguiente tabla:

**NUM. DE KILOS COMPRADOS % DESCUENTO**

0 – 2 0%

2.01 – 5 10%

5.01 – 10 15%

10.01 en adelante 20%

Determinar cuál será el descuento para una persona que compre manzanas es esa frutería.

1. Una empresa desea calcular los nuevos salarios de sus empleados de la siguiente forma: quienes ganan hasta $ 500.000 tendrán un incremento del 10%, quienes devengan más de $ 500.000 y hasta 800.000 recibirán un aumento del 8% y los demás del 5%. ¿Cuál será el valor del aumento? ¿Cuál será el valor del nuevo sueldo?
2. Consulte cómo saber en qué cuadrante del plano cartesiano se ubican las coordenadas representadas por los valores que toman X y YE. Luego escribir un algoritmo que pida ingresar la coordenada de un punto en el plano, es decir dos valores enteros x e y (distintos a cero). Posteriormente imprimir en pantalla en que cuadrante se ubica dicho punto. (1º Cuadrante si x > 0 Y y > 0 , 2º Cuadrante: x < 0 Y y > 0, lo mismo para el tercer y cuarto cuadrante.)

Como apoyo o referencia, observe los siguientes enlaces:

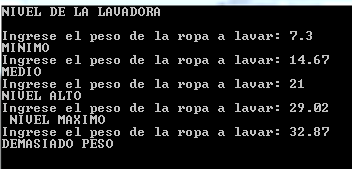
<https://www.youtube.com/watch?v=jlKv4Vugy8c>

<https://www.youtube.com/watch?v=G_Oq4ma_cXY>

<http://www.profesorenlinea.cl/geometria/Plano_Cartesiano.html>

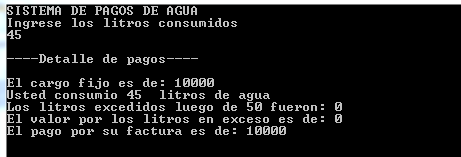
1. Implemente un algoritmo que tome el peso en kilos de una cantidad de ropa a lavar en una lavadora industrial y que le devuelva el nivel dependiendo del peso. Se sabe que con más de 30 kilos la lavadora no funcionará ya que es DEMASIADO PESO. Si la ropa pesa 22 o más kilos, el nivel será de MÁXIMO; si pesa 15 o más el nivel será de ALTO; si pesa 8 o más será un nivel MEDIO o de lo contrario el nivel será MÍNIMO.

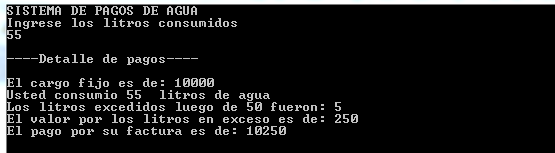
Se debe solicitar al menos cinco veces el peso de la ropa que se quiere lavar y la salida debe ser parecida a la siguiente:

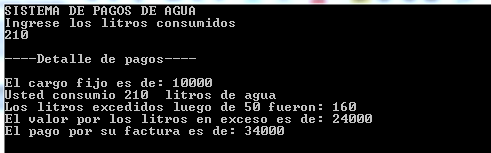


1. Implemente un algoritmo que permita calcular el pago que tiene que realizar una familia por el consumo mensual del agua potable, sabiendo que existe un pago fijo de $10.000 y que los 50 primeros litros de agua no tienen costo, entre 51 y 200 litros se cobra $50 por litro, y por más de 200 litros a $150 por litro.

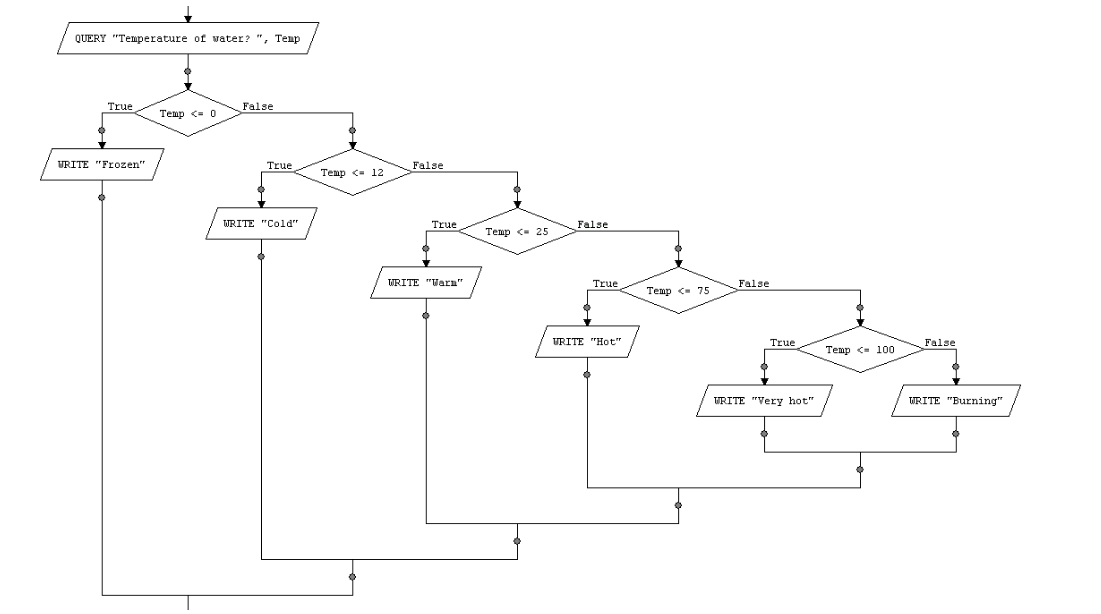
Los resultados deben ser similares a los siguientes:







1. Consulte qué es un Diagrama de flujo y analice y entienda el uso de cada uno de sus componentes. Luego interprete el siguiente diagrama y escriba el respectivo pseudocódigo y su codificación en Java.



1. Escriba un algoritmo que lea dos números enteros (**Num1 y Num2**) y un operador aritmético (+, -, \*, /), al cual almacenaremos en la variable **Op**. Luego debe imprimir el resultado de la operación Num1 Op Num2. Para facilitar su análisis, vea el siguiente ejemplo:

*Ingrese el primer número: 5*

*Ingrese el segundo número: 2*

*Ingrese el operador: \**

*El resultado es: 10*

1. Escriba un algoritmo que lea un número real y que imprima un mensaje de acuerdo con la siguiente tabla:

|  |  |
| --- | --- |
| Contenido de alcohol en g | Mensaje |
| 40 < g | Licor extra fuerte |
| 20 < g <= 40 | Licor fuerte |
| 15 < g <= 20 | Licor |
| 12 < g <= 15 | Vino fuerte |
| 10.5 < g <= 12 | Vino normal |
| g <= 10.5 | Vino suave |

1. La mamá de Juancho vende hamburguesas por la 33, pero al final del día no sabe si tuvo o no ganancias, pues no cuenta con un sistema de costos que le permita saber este dato. Una hamburguesa la vende en $5.000, incluyendo la gaseosa. Para producir una hamburguesa se requiere la siguiente inversión:

|  |  |  |
| --- | --- | --- |
| **Ingrediente** | **Cantidad/Unidad** | **Costo/Unidad** |
| Pan | 1 | $800 |
| Ensalada | 1 | $1200 |
| Salsas | 1 | $300 |
| Carne | 1 | $850 |
| Queso | 1 | $250 |
| Huevo | 1 | $100 |
| Tocineta | 1 | $800 |
| Gaseosa | 1 | $500 |

Ella desea saber si se está ganando por hamburguesa al menos un 20%, pues de lo contrario, debe aumentar el precio de venta.

1. Una familia cuenta con ingresos mensuales de $2.300.000, pero debe distribuirlos en los siguientes gastos:

|  |  |  |
| --- | --- | --- |
| **Concepto** | **Valor** | **Porcentaje** |
| Arriendo | 550.000 |  |
| Servicios públicos | 230.000 |  |
| Mercado | 450.000 |  |
| Colegio | 180.000 |  |
| Cuota carro | 350.000 |  |
| Transporte escolar | 100.000 |  |
| Matrícula | ¿? |  |
| TOTAL |  |  |
| SALDO |  |  |

Luego de leer estos datos, se requiere mostrar lo siguiente:

El total de gastos de la familia: $xxxxx

El porcentaje que representa cada gasto con respecto al total es de: x% (No incluir Matrícula).

El porcentaje del total de gastos con respecto al total de ingresos es de: x%

Desean conocer si pueden pagar la matrícula, sabiendo que cuesta $500.000.

**Selección múltiple**

**Introducción**

Ya hemos visto que la estructura decisión se puede construir con la instrucción SI y su componente opcional DE\_LO\_CONTRARIO. Sin embargo, hay situaciones en las cuales su utilización genera algoritmos poco legibles y quizás más complejos. Con el fin de evitar estos inconvenientes se ha creado la instrucción CASOS, la cual permite construir algoritmos más legibles y menos complejos. En este módulo se trata dicha instrucción.

**Objetivos**

1. Reconocer y aplicar la instrucción CASOS.

Preguntas básicas

1. ¿En qué consiste la instrucción CASOS?
2. ¿Cuándo se aplica la instrucción CASOS?
3. ¿En qué consiste la opción SALTE?

Contenidos

* Forma general
* Ejemplo de uso

**Forma general**

La forma general de la instrucción CASOS es:

CASOS

:condición 1: instrucciones que se ejecutan cuando la condición 1 es verdadera.

SALTE

:condición 2: instrucciones que se ejecutan cuando la condición 2 es verdadera.

SALTE

:condición n: instrucciones que se ejecutan cuando la condición n es verdadera.

SALTE

.

.

.

:OTRO\_CASO: instrucciones que se ejecutan cuando ninguna de las condiciones anteriores es verdadera.

Fin(CASOS)

La componente SALTE presente al final de cada condición es opcional, y veremos su funcionalidad más adelante, con los ejemplos.

**Ejemplo de uso**

Elabore un algoritmo que lea el nombre de una persona y su estado civil. El estado civil está codificado con un dígito con los siguientes significados:

1: Soltero

2: Casado

3: Separado

4: Viudo

5: Unión libre

El algoritmo debe imprimir el nombre leído y la descripción correspondiente al estado civil leído.

Análisis:

Datos de entrada:

Nombre de la persona: nom

Estado civil: ec

Datos de salida:

Nombre de la persona: nom

Estado civil: ec

Cálculos:

Comparar el estado civil leído según el código establecido para determinar si es 1, 2, 3 u otro valor, e imprimir la descripción correspondiente.

Nuestro algoritmo quedará así:

1. Algoritmo estadoCivil(1)

2. Variables nom: alfanumérica

3. ec: numericaEntera

4. INICIO

5. LEA(nom, ec)

6. SI ec == 1

7. ESCRIBA(nom, “ soltero”)

8. DE\_LO\_CONTRARIO

9. SI ec == 2

10. ESCRIBA(nom, “ casado”)

11. DE\_LO\_CONTRARIO

12. SI ec == 3

13. ESCRIBA(nom, “ separado”)

14. DE\_LO\_CONTRARIO

15. SI ec == 4

16. ESCRIBA(nom, “ viudo”)

17. DE\_LO\_CONTRARIO

18. SI ec == 5

19. ESCRIBA(nom, “ unión libre”)

20. DE\_LO\_CONTRARIO

21 ESCRIBA(ec, “ estado civil inválido”)

22. Fin(SI)

23. Fin(SI)

24. Fin(SI)

25. Fin(SI)

26. Fin(SI)

27. FIN

28. FIN(estadoCivil)

Si en vez de haber tenido en estado civil hubiéramos tenido más de 5, implicaría que tendríamos que hacer más instrucciones SI, pero que le quitan legibilidad al algoritmo. En este caso es más conveniente utilizar la instrucción CASOS.

Veamos nuestro algoritmo:

1. Algoritmo estadoCivil(2)

2. Variables: nom: alfanumérica

3. ec: numericaEntera

4. INICIO

5. LEA(nom, ec)

6. CASOS

7. :ec == 1: ESCRIBA(nom, “ soltero”)

8. :ec == 2: ESCRIBA(nom, “ casado”)

9. :ec == 3: ESCRIBA(nom, “ separado”)

10. :ec == 4: ESCRIBA(nom, “ viudo”)

11. :ec == 5: ESCRIBA(nom, “ unión libre”)

12. :OTRO\_CASO: ESCRIBA(ec, “ estado civil inválido”)

13. Fin(CASOS)

14. FIN

15. Fin(estadoCivil)

Dentro de la instrucción CASOS cuando escribimos :ec == 1: la máquina compara ec con 1; si ec es igual a 1 ejecuta las instrucciones correspondientes a ese caso. Como no hemos utilizado la instrucción SALTE, al evaluar la primera condición y resultar verdadera, seguiría preguntando por la segunda y así hasta el final, haciendo preguntas inoficiosas.

Un algoritmo equivalente al anterior, utilizando la instrucción SI, es el siguiente:

1. Algoritmo estadoCivil(3)

2. Variables: nom: alfanumérica

3. ec: numericaEntera

4. INICIO

5. LEA(nom, ec)

6. SI ec == 1

7. ESCRIBA(nom, “ soltero”)

8. Fin(SI)

9. SI ec == 2

10. ESCRIBA(nom, “ casado”)

11. Fin(SI)

12. SI ec == 3

13. ESCRIBA(nom, “ separado”)

14. Fin(SI)

15. SI ec == 4

16. ESCRIBA(nom, “ viudo”)

17. Fin(SI)

18. SI ec == 5

19. ESCRIBA(nom, “ unión libre”)

20. Fin(SI)

21. SI ec <> 1 && ec<>! 2 && ec <> 3 && ec <> 4 && ec <> 5

22. ESCRIBA(ec, “ estado civil inválido”)

23. Fin(SI)

24. FIN

25. Fin(estadoCivil)

Este algoritmo tiene el “inconveniente” de que si el estado civil es 1, al hacer la primera comparación escribe el nombre y la descripción “soltero” y sigue preguntando inoficiosamente si el estado civil es 2, o 3, o 4, o 5, y más aún, en la instrucción 21 pregunta si el estado civil es diferente de 1.

Si utilizamos la instrucción CASOS y queremos evitar las preguntas inoficiosas, usamos la componente opcional SALTE. Cuando el grupo de instrucciones correspondiente a una condición termina con la instrucción SALTE, se está instruyendo a la máquina para que no siga averiguando por las demás condiciones y que continúe la ejecución en la instrucción siguiente al fin de los CASOS.

Nuestro algoritmo estadoCivil, utilizando la instrucción CASOS con la componente SALTE,

queda así:

1. Algoritmo estadoCivil(4)

2. Variables: nom: alfanumérica

3. ec: numericaEntera

4. INICIO

5. LEA(nom, ec)

6. CASOS

7. :ec == 1: ESCRIBA(nom, “ soltero”)

8. SALTE

9. :ec == 2: ESCRIBA(nom, “ casado”)

10. SALTE

11. :ec == 3: ESCRIBA(nom, “ separado”)

12. SALTE

13. :ec == 4: ESCRIBA(nom, “ viudo”)

14. SALTE

15. :ec == 5: ESCRIBA(nom, “ unión libre”)

16. SALTE

17. :OTRO\_CASO: ESCRIBA(ec, “ estado civil inválido”)

18. Fin(CASOS)

19. FIN

20. Fin(estadoCivil)

Un punto importante que se debe considerar en este sitio es: ¿cuándo utilizar la instrucción SI, y cuándo utilizar la instrucción CASOS?

La respuesta es muy sencilla:

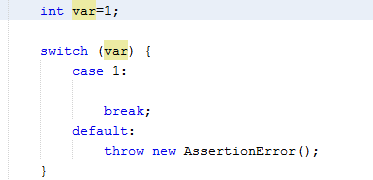
* Cuando el resultado de una comparación sólo da dos alternativas se utiliza la instrucción SI.
* Cuando el resultado de una comparación da más de dos alternativas se utiliza la instrucción CASOS.

Es conveniente también hacer notar que no siempre es recomendable utilizar la componente SALTE. Por ejemplo, se ejecutan el grupo de instrucciones que pertenecen a una condición, y puede que sea necesario que sigamos preguntando por las siguientes condiciones.

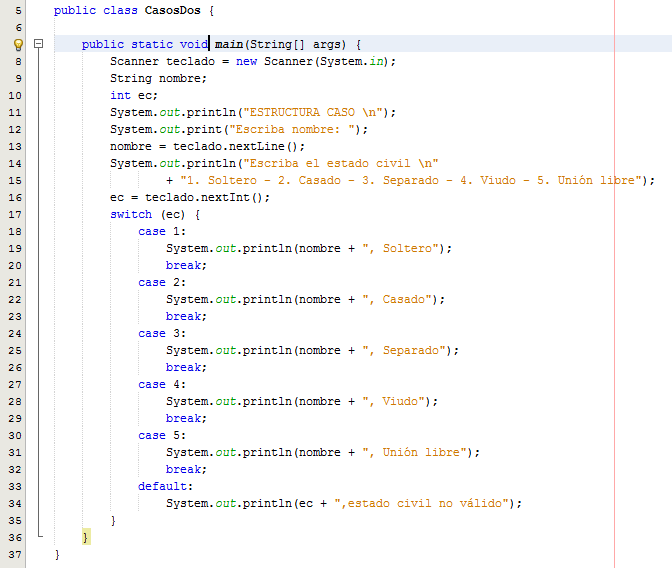
**Resumen**

Hemos presentado la instrucción de decisión CASOS, que hace más fácil evitar muchas instrucciones SI y logra que el algoritmo sea más sencillo y legible.

Veamos ahora la sintaxis de esta instrucción en el lenguaje Java:



Veamos la codificación del algoritmo:



**Ejemplo completo**

El siguiente ejemplo incluye todas las instrucciones que hemos visto hasta el momento.

Vamos a construir un algoritmo, en el cual se lea el nombre de una persona, el estado civil de la persona, la edad de la persona, el salario actual de la persona. Con base en esos datos vamos a definir cuál es el nuevo salario de esa persona y vamos a mostrar un mensaje con el nuevo salario y la descripción del estado civil. Para calcular el nuevo salario de la persona, vamos a tener en cuenta el estado civil y la edad. Veamos:

* Estado civil: Soltero y mayor de 30 años, el aumento es del 12%
* Estado civil: Soltero y menor o igual a 30 años, el aumento es del 10%
* Estado civil: Casado y mayor de 25 años, el aumento es del 15%
* Estado civil: Casado y menor o igual a 25 años, el aumento es del 8%
* Estado civil: Separado y mayor de 40 años, el aumento es del 20%
* Estado civil: Separado y menor o igual a 40 años, el aumento es del 16%

Procedamos:

Análisis

Datos de entrada: nombre, estado civil, edad, salario actual

Datos de salida: nombre, estado civil, descripción del estado civil, edad, % de aumento, aumento, nuevo salario, salario actual.

Cálculos: determinar % de aumento según las políticas de la empresa:

Soltero > 30 años 12%

<= 30 años 10%

Casados > 25 años 15%

<= 25 años 8%

Separados > 40 años 20%

<= 40 años 16%

Algoritmo **aumento\_con\_condicion**

Variables: nom, ec, edad, sa, desc, pau, au, ns

INICIO

LEA (nom, ec, edad, sa)

CASOS

:ec=1 desc=”Soltero”

SI edad > 30

pau=0.12

DE\_LO\_CONTRARIO

pau=0.10

Fin(SI)

SALTE

:ec=2 desc=”Casado”

SI edad > 25

pau=0.15

DE\_LO\_CONTRARIO

pau=0.08

Fin(SI)

SALTE

:ec=3 desc=”Separado”

SI edad > 40

pau=0.20

DE\_LO\_CONTRARIO

pau=0.16

Fin(SI)

SALTE

:otro\_caso: desc=”Estado civil no válido”

pau=0.0

Fin(casos)

au=sa \* pau

ns = sa + au

ESCRIBA (nom, ec, desc, edad, sa, pau, au, ns)

FIN

Fin (aumento\_con\_condicion)

Haga una prueba de escritorio al algoritmo anterior para cada uno de los siguientes datos:

|  |  |  |  |
| --- | --- | --- | --- |
| nom | ec | ed | sa |
| Juan | 1 | 28 | 2500 |
| Luis | 2 | 31 | 800 |
| Ana | 3 | 18 | 1600 |
| Luz | 4 | 45 | 3000 |
| Clara | 5 | 33 | 2500 |
| Inés | 3 | 25 | 1000 |
| Lulú | 1 | 39 | 700 |

**Ejercicios propuestos**

Elabore un algoritmo que lea los siguientes datos de una muchacha: edad, estatura, peso y estado civil. El algoritmo debe clasificar cada chica de acuerdo con los siguientes criterios: si es menor de 15 años y mide menos de 160 cm es una Mafalda; si está entre 18 y 25 años, mide más de 175 cm y pesa menos de 65 kg es una Barbie; si es mayor de 50 años, mide más de 168 cm, pesa más de 80 kg y es casada es una Helga; si es casada, mayor de 25 años y menor de 30 años y pesa menos de 70 kg es una reina; y si es soltera, mide más de 180 cm, pesa entre 65 y 75 kg y tiene menos de 21 años es una Princesa.

La empresa de productos de belleza ‘El Acné’ otorga descuento a sus clientes según la siguiente clasificación: si es mayorista, tiene una antigüedad de más de dos años y el valor de la compra es mayor de 2.00.000 de pesos, le da un descuento del 25%; si es mayorista, tiene una antigüedad menor o igual que dos años y el valor de la compra está entre 1.500.000 y 2.000.000 de pesos, le da un descuento de 20%; si es minorista, tiene una antigüedad superior a cinco años y el valor de la compra es superior a 2.000.000 de pesos, le da un descuento del 10%. En cualquier otro caso, la compañía no da ningún descuento. Elabore un algoritmo que lea la clase de cliente, la antigüedad y el valor de la compra y determine el valor a pagar por la compra.

Los códigos de clasificación del cliente son:

1. Mayorista
2. Minorista
3. Ocasional

**CAPÍTULO 2: ESTRUCTURAS CÍCLICAS**

Los algoritmos elaborados hasta este punto han tratado temas de programación como entradas, salidas, asignaciones, expresiones, secuencias y decisiones. Sin embargo, hay muchas situaciones en las cuales se requiere que algunas operaciones o secuencia de instrucciones se repitan una y otra vez utilizando diferentes datos.

En este capítulo se estudian las diferentes instrucciones que se utilizan para instruir al computador con el fin de que ejecute más de una vez cierto conjunto de instrucciones de acuerdo con alguna condición establecida.

Ejecutar la misma actividad varias veces, con diferentes datos, clientes, etc., es una situación común en la vida diaria.

**Estructura ciclo e instrucción MIENTRAS**

Introducción

En los ejercicios desarrollados anteriormente leíamos el salario actual de un empleado y determinábamos el aumento con base en un porcentaje leído o en una condición específica (que gane menos de mil pesos, por ejemplo).

Realmente, cuando se desea actualizar salarios, no es para un solo empleado, sino para todos los empleados de la empresa. Con el algoritmo que habíamos desarrollado, si deseamos actualizar el salario de todos los empleados debemos ejecutarlo tantas veces como empleados tenga la empresa, lo cual es una tarea muy dispendiosa. Para evitar esto podemos escribir un algoritmo con el cual procesemos todos los empleados utilizando una instrucción de ciclo.

Objetivos

* Reconocer la estructura ciclo y sus instrucciones.
* Reconocer la instrucción MIENTRAS.
* Identificar el concepto de contador.

Preguntas básicas

* ¿Qué es un ciclo?
* ¿Cuáles son las instrucciones que pertenecen a la estructura ciclo?
* ¿Cómo funciona la instrucción MIENTRAS?
* ¿En qué consiste un contador?

Contenidos

* Definición de estructura ciclo
* Instrucción MIENTRAS
* Entrada de datos usando un ciclo
* Ejemplo de uso
* Concepto de contador

Definición de estructura ciclo

La estructura ciclo está conformada por instrucciones que permiten al programador instruir la máquina para que ejecute varias veces un grupo de instrucciones según alguna condición.

Las instrucciones que pertenecen a esta estructura son:

MIENTRAS, PARA y HAGA.

Instrucción MIENTRAS

La forma general de la instrucción MIENTRAS es:

MIENTRAS condición

Conjunto de instrucciones que se ejecutan mientras que la condición sea verdadera

Fin(MIENTRAS)

La condición puede ser una expresión relacional o una expresión lógica. En la condición, por lo general, se evalúa el valor de una variable. Esta variable, que se debe modificar en cada iteración, recibe el nombre de variable controladora del ciclo.

Entrada de datos usando un ciclo

Para elaborar un algoritmo con ciclos con el fin de determinar el aumento y el nuevo salario de todos los empleados de una empresa, según los criterios del algoritmo del módulo anterior, debemos primero definir cómo entrar los datos al algoritmo. Una forma es a través del teclado y otra forma es leyendo los datos de cada empleado desde un archivo previamente almacenado en el computador.

Vamos a elaborar nuestro algoritmo leyendo los datos desde un archivo. Como habíamos visto anteriormente, un archivo es una colección de registros, en la cual en cada registro se almacenan los datos correspondientes a un empleado. Cuando se procesa un archivo, el computador debe identificar el nombre del archivo y cuándo terminó de procesarlo. Para identificar esta segunda situación, cada archivo tiene al final un registro especial que indica que ya no hay más registros. Este registro especial contiene lo que se conoce como marca de fin de archivo. Llamemos a esta marca EOF, de las iniciales de las palabras en inglés End of File (fin de archivo).

Por ejemplo, un archivo con cinco registros llamado ‘**nomina’**, en el cual cada registro contiene el nombre y el salario actual de un empleado, tiene la siguiente conformación:

Archivo ‘**nómina’**

|  |  |  |
| --- | --- | --- |
| n.º de registro | Nombre | Salario actual |
| 1 | Sara | 2000 |
| 2 | Pedro | 3000 |
| 3 | María | 1500 |
| 4 | Juan | 500 |
| 5 | Elena | 2800 |
| 6 | EOF |  |

Cuando se está procesando un archivo y se ejecuta la instrucción de lectura LEA, la máquina lee los datos de un registro y se posiciona para leer los datos del siguiente registro. Por consiguiente, para procesar el anterior archivo debemos ejecutar la instrucción LEA cinco veces.

Además, para procesar un archivo necesitamos ponerlo a disposición del programa que lo va a procesar. Esto se logra con la instrucción ABRA(nombre del archivo). También incluiremos una pequeña modificación en la instrucción de lectura para señalar cuál es el archivo que se va a leer, y una instrucción que cierre el archivo cuando se termina de procesar.

Nuestra instrucción de lectura queda:

LEA(nombre del archivo: lista de variables, separadas por comas)

Ejemplo de uso

Considerando estas características, con respecto al manejo de archivos, nuestro algoritmo queda:

1. Algoritmo aumentoEnArchivo(1)

2. Variables: sa, au, ns: numericaReal

3. nom: alfanumérica

4. INICIO

5. ABRA(nomina)

6. MIENTRAS no sea EOF(nomina)

7. LEA(nomina: nom, sa)

8. au = 0

9. SI sa < 1000

10. au = sa \* 0.1

11. Fin(SI)

12. ns = sa + au

13. ESCRIBA(nom, sa, au, ns)

14. Fin(MIENTRAS)

15 CIERRE(nomina)

16. FIN

17. Fin(aumentoEnArchivo)

Al ejecutar la instrucción 5, el dispositivo de lectura se posiciona para leer el primer registro del archivo llamado ‘**nomina’**.

La instrucción 6 pregunta si el registro disponible para ser leído tiene la marca de fin de archivo; de ser así, la ejecución continuará en la instrucción 15.

Si el registro disponible para ser leído no es la marca de fin de archivo, la ejecución continúa con la instrucción 7.

La instrucción 7 lee nombre y salario y los almacena en las variables llamadas nom y sa: en la variable nom queda almacenado Sara y en la variable sa queda almacenado 2000.

Además de cargar las variables nom y sa con los datos leídos, la instrucción de lectura deja posicionado el dispositivo de lectura para que lea el siguiente registro.

En la instrucción 8 se le asigna 0 al aumento.

La instrucción 9 compara el salario leído con 1000. Si el salario leído es menor que 1000 ejecuta la instrucción 10, la cual calcula el aumento como el diez por ciento del salario leído (au = sa \* 0.1). Si el salario leído no es menor que 1000, el aumento se queda en cero.

En la instrucción 12 se calcula el nuevo salario, y en la 13 se escribe el nombre del empleado, el salario actual, el aumento y el nuevo salario.

La instrucción 14, que es el fin de la instrucción MIENTRAS, hace que el algoritmo retorne a la instrucción 6 en la cual va a controlar que el registro para leer no tenga la marca de fin de archivo.

Al ejecutar por segunda vez la instrucción LEA, las variables nom y sa quedan con los datos Pedro y 3000, respectivamente.

Nuevamente se le asigna 0 al valor del aumento, ya que si no hiciéramos esto en la variable au quedará el aumento que se le hizo al anterior empleado.

Al procesar el archivo ‘**nomina’** en su totalidad, los resultados obtenidos serán:

|  |  |  |  |
| --- | --- | --- | --- |
| nom | sa | au | ns |
| Sara | 2000 | 0 | 2000 |
| Pedro | 3000 | 0 | 3000 |
| María | 1500 | 0 | 1500 |
| Juan | 500 | 50 | 550 |
| Elena | 2800 | 0 | 2800 |

Concepto de contador

En muchas ocasiones es importante conocer, al final del proceso, cuántos registros se procesaron. Para ello utilizamos una variable adicional, la cual tendrá un valor inicial de 0, y cada vez que se lea un registro la incrementamos en 1. Una variable con esta característica se denomina CONTADOR.

Escribamos nuestro algoritmo aumentoEnArchivo contando los registros procesados e imprimiendo dicho total al final del proceso del archivo:

1. Algoritmo aumentoEnArchivo(2)

2. Variables: sa, au, ns: numericaReal

3. nom: alfanumérica

4. i: numericaEntera

5. INICIO

6. ABRA(nomina)

7. i = 0

8. MIENTRAS no sea EOF(nomina)

9. LEA(nomina: nom, sa)

10. i = i + 1

11. au = 0

12. SI sa < 1000

13. au = sa \* 0.1

14. Fin(SI)

15. ns = sa + au

16. ESCRIBA(sa, au, ns)

17. Fin(MIENTRAS)

18. CIERRE(nomina)

19. ESCRIBA(“Total de empleados: ”, i)

20. FIN

21. Fin(aumentoEnArchivo)

En este segundo algoritmo hemos introducido cuatro nuevas instrucciones: la instrucción 4, en la cual definimos la variable i, que hará las veces de contador; la instrucción 7, en la cual se inicializa el contador en 0; la instrucción 10, en la cual se incrementa el contador en 1, cada vez que se lee un registro; y la instrucción 19, en la cual se imprime el contenido del contador con su correspondiente mensaje.

Al ejecutar este segundo algoritmo con el mismo archivo ‘**nómina**’ el resultado que se obtiene es:

|  |  |  |  |
| --- | --- | --- | --- |
| nom | sa | au | ns |
| Sara | 2000 | 0 | 2000 |
| Pedro | 3000 | 0 | 3000 |
| María | 1500 | 0 | 1500 |
| Juan | 500 | 50 | 550 |
| Elena | 2800 | 0 | 2800 |

Total de empleados: 5

**Ejercicios propuestos**

1. Elabore un algoritmo que procese un archivo llamado ‘**edades’**, en el cual cada registro contiene el nombre de una persona y su edad en años. El algoritmo debe imprimir el nombre de la persona y su clasificación de acuerdo a los siguientes criterios: la persona es ‘niño’, si la edad está entre 0 y 10 años; ‘adolescente’, si la edad está entre 11 y 18 años; ‘adulto’, si la edad está entre 19 y 35 años; ‘maduro’, si la edad está entre 36 y 60 años; y ‘anciano’, si la edad es mayor de 60 años. Además, al final debe imprimir cuántas personas hay en cada clasificación y el total de personas.
2. Elabore un algoritmo que procese un archivo llamado ‘**promedios’**, en el cual cada registro contiene el nombre de un estudiante y su promedio acumulado de la carrera. El algoritmo debe imprimir el nombre del estudiante y alguno de los siguientes mensajes: ‘pésimo’, ‘malo’, ‘regular’, ‘bueno’ o ‘excelente’. El estudiante se considera pésimo si el promedio acumulado es menor o igual que 1; malo, si el promedio es mayor que 1 y menor que 3; regular, si el promedio es mayor o igual que 3 y menor que 4; bueno, si el promedio es mayor o igual que 4 y menor que 4.5; y excelente, si el promedio es mayor o igual que 4.5. Al final el algoritmo debe imprimir el número de estudiantes en cada clasificación y el total de estudiantes.
3. Se tiene una competencia ciclística en la cual los competidores son clasificados por categorías dependiendo del sexo y la edad que tengan. Las categorías definidas son: categoría A, mujeres menores de 14 años; B, mujeres entre 14 y 20 años; C, mujeres entre 21 y 35 años; D, mujeres mayores de 35 años; E, hombres menores de 15 años; F, hombres entre 15 y 25 años; G, hombres entre 26 y 40 años; H, hombres mayores de 40 años. Para ello se ha grabado un archivo, llamado ‘**inscritos’**, en el cual cada registro contiene los siguientes datos: nombre, sexo y edad. El sexo está codificado así: 0 significa que es mujer, 1 significa que es hombre. Elabore un algoritmo que procese dicho archivo y que imprima cuántas personas hay inscritas en cada categoría, en cual categoría se inscribieron más personas y el total de participantes inscritos.
4. Se tiene un archivo llamado ‘**definitivas’**, en el cual cada registro contiene el nombre de un estudiante y las tres notas correspondientes a las notas definitivas de las tres material que cursó. Elabore un algoritmo que calcule e imprima el promedio de cada estudiante y que al final imprima el nombre del estudiante que obtuvo el mejor promedio y el nombre del estudiante que obtuvo el peor promedio.

Contadores y acumuladores

Cuando se trabaja un ciclo es necesario, en muchas ocasiones, contabilizar el total de ciertos datos que se procesan dentro del ciclo, para luego usarlos. Esta contabilización se efectúa usando variables que, en general, se denominan contadores y acumuladores.

Objetivos

* Identificar los conceptos de contador y acumulador.
* Aplicar y usar los conceptos de contador y acumulador.

Preguntas básicas

* ¿Qué es un acumulador?
* ¿Cómo se procesa un acumulador?
* ¿En cuál parte del procesamiento se usa el resultado de un acumulador?

Contenidos

* Concepto de acumulador
* Ejemplo de algoritmo con acumula
* Aplicación y uso de acumuladores

***Cuando se trata de actividades que se ejecutan cíclicamente, es necesario en muchas ocasiones manejar datos acerca de las actividades que se están desarrollando. Por ejemplo, al tendero le interesará conocer cuántos clientes ha atendido y cuál es el total de ventas. La variable que le dice cuántos clientes ha atendido será un contador, y la que le dice cuál es el total de ventas es un acumulador***.

**Concepto de acumulador**

En uno de los algoritmos anteriores se presentó el concepto de contador, que es una variable que se utiliza, como su nombre lo dice, para contar las ocurrencias de algún evento. Es también necesario, en muchas ocasiones, conocer totales de ciertos datos, como total de salarios anteriores, total de aumentos y total de nuevos salarios.

Para contabilizar estos totales debemos manejar otras tres variables adicionales: una para llevar el total de salarios anteriores, otra para llevar el total de aumentos y otra para llevar el total de nuevos salarios. Llamemos a estas variables tsa, tau y tns, respectivamente.

Inicialmente estas variables deben tener un valor de cero. Cada vez que se lea un registro debemos incrementar el total de salarios anteriores con el salario leído:

tsa = tsa + sa

Cada vez que se calcule un nuevo aumento debemos incrementar el total de aumentos:

tau = tau + au

Cada vez que se calcule un nuevo salario debemos incrementar el total de nuevos salarios:

tns = tns + ns

Una variable con esta característica se denomina ACUMULADOR.

En nuestro ejemplo las variables tsa, tau y tns son entonces acumuladores.

Ejemplo de algoritmo con acumuladores

Nuestro algoritmo , aumentoEnArchivo, quedará así:

1. Algoritmo aumentoEnArchivo(3)

2. Variables: sa, au, ns, tsa, tau, tns: numericaReal

3. nom: alfanumérica

4. i: numericaEntera

5. INICIO

6. tsa = 0

7. tau = 0

8. tns = 0

9. i = 0

10. ABRA(nomina)

11. MIENTRAS no sea EOF(nomina)

12. LEA(nomina: nom, sa)

13. tsa = tsa + sa

14. i = i + 1

15. au = 0

16. SI sa < 1000

17. au = sa \* 0.1

18. tau = tau + au

19. Fin(SI)

20. ns = sa + au

21. tns = tns + ns

22. ESCRIBA(sa, au, ns)

23. Fin(MIENTRAS)

24. CIERRE(nomina)

25. ESCRIBA(“total empleados: “, i)

26. ESCRIBA(“total salario anterior: “, tsa)

27. ESCRIBA(“total aumentos: “, tau)

28. ESCRIBA(“total nuevos salarios: “, tns)

29. FIN

30. Fin(aumentoEnArchivo)

En las instrucciones 6 a 8 se inicializan los acumuladores en cero.

En la instrucción 13 se acumula el salario actual. Esta instrucción se ejecuta en este sitio, puesto que todo empleado que se lee tiene un salario actual y el acumulador se debe actualizar tan pronto se tenga el dato al cual hay que llevarle un total.

La instrucción 18 acumula el total de aumentos. Esta instrucción se pone en este sitio, ya que ahí es donde se determina el aumento de cada empleado. Realmente esta instrucción se puede colocar fuera de la instrucción SI, pero es mejor colocarla adentro, ya que si lo hacemos después del fin del SI, en algunas ocasiones el resultado de efectuar esta instrucción es sumarle cero al acumulador tau, lo cual no afecta el contenido de tau pero implica ejecutar una operación.

En la instrucción 21 se acumula el total del nuevo salario, es decir, en la instrucción inmediatamente siguiente a la instrucción en la cual se calculó el nuevo salario.

Al ejecutar este algoritmo con los datos de nuestro archivo ‘nomina’ los resultados que obtenemos son:

|  |  |  |  |
| --- | --- | --- | --- |
| nom | sa | au | ns |
| Sara | 2000 | 0 | 2000 |
| Pedro | 3000 | 0 | 3000 |
| María | 1500 | 0 | 1500 |
| Juan | 500 | 50 | 550 |
| Elena | 2800 | 0 | 2800 |

Total de empleados: 5

Total del salario anterior: 9800

Total de aumentos: 50

Total de nuevos salarios: 9850

**Aplicación y uso de acumuladores**

Otros datos de interés que se suelen calcular son los promedios de algunos datos. En nuestro ejercicio, por ejemplo, es interesante conocer los promedios de salario anterior, de aumento y de nuevo salario. En general, para determinar un promedio basta con dividir un acumulador entre un contador. Así, en nuestro ejercicio el promedio de salario anterior se obtiene dividiendo el total de salario anterior entre el contador de empleados, el promedio de aumento se obtiene dividiendo el total de aumento entre el contador de empleados y el promedio del nuevo salario se obtiene dividiendo el total del nuevo salario entre el contador de empleados.

Llamemos a estas variables para promedios psa, pau y pns.

Nuestro algoritmo aumentoEnArchivo queda:

1. Algoritmo aumentoEnArchivo(4)

2. Variables: sa, au, ns, tsa, tau, tns, psa, pau, pns: numericaReal

3. nom: alfanumérica

4. i: numericaEntera

5. INICIO

6. tsa = 0

7. tau = 0

8. tns = 0

9. i = 0

10. ABRA(nomina)

11. MIENTRAS no sea EOF(nomina)

12. LEA(nomina: nom, sa)

13. tsa = tsa + sa

14. i = i + 1

15. au = 0

16. SI sa < 1000

17. au = sa \* 0.1

18. tau = tau + au

19. Fin(SI)

20. ns = sa + au

21. tns = tns + ns

22. ESCRIBA(sa, au, ns)

23. Fin(MIENTRAS)

24. CIERRE(nomina)

25. psa = tsa / i

26. pau = tau / i

27. pns = tns / i

28. ESCRIBA(“total empleados: “, i)

29. ESCRIBA(“total salario anterior: “, tsa, “ promedio: “, psa)

30. ESCRIBA(“total aumentos: “, tau, “ promedio: “, pau)

31. ESCRIBA(“total nuevos salarios: “, tns, “ promedio: “, pns)

32. FIN

33. Fin(aumentoEnArchivo)

Los promedios se calculan en las instrucciones 25, 26 y 27.

Nota: los promedios se calculan después de que se ha terminado de ejecutar el ciclo MIENTRAS.

Es un error común en los principiantes calcular promedios dentro del ciclo. Los promedios se calculan cuando se tiene el total de los contadores y el total de los acumuladores, y estos datos sólo se tienen cuando se ha terminado el ciclo.

Al ejecutar este último algoritmo con nuestro archivo ‘**nomina’** los resultados que obtenemos son los siguientes:

|  |  |  |  |
| --- | --- | --- | --- |
| nom | sa | au | ns |
| Sara | 2000 | 0 | 2000 |
| Pedro | 3000 | 0 | 3000 |
| María | 1500 | 0 | 1500 |
| Juan | 500 | 50 | 550 |
| Elena | 2800 | 0 | 2800 |

Total de empleados: 5

Total del salario anterior: 9800 promedio: 1960

Total de aumentos: 50 promedio: 10

Total de nuevos salarios: 9850 promedio: 1970

Ejercicios propuestos

1. Elabore un algoritmo que procese un archivo llamado ‘**edades’**, en el cual cada registro contiene el nombre de una persona y su edad en años. El algoritmo debe imprimir el nombre de la persona y su clasificación de acuerdo a los siguientes criterios: la persona es ‘niño’, si la edad está entre 0 y 10 años; ‘adolescente’, si la edad está entre 11 y 18 años; ‘adulto’, si la edad está entre 19 y 35 años; ‘maduro’, si la edad está entre 36 y 60 años; y ‘anciano’, si la edad es mayor de 60 años. Además, al final debe imprimir cuántas personas hay en cada clasificación, el total de personas en el archivo, el promedio de edad en cada categoría y el promedio de edad de todas las personas.
2. Elabore un algoritmo que procese un archivo llamado ‘**promedios’**, en el cual cada registro contiene el nombre de un estudiante y su promedio acumulado de la carrera. El algoritmo debe imprimir el nombre del estudiante y alguno de los siguientes mensajes: ‘pésimo’, ‘malo’, ‘regular’, ‘bueno’ o ‘excelente’. El estudiante se considera pésimo si el promedio acumulado es menor o igual que 1; malo, si el promedio es mayor que 1 y menor que 3; regular, si el promedio es mayor o igual que 3 y menor que 4; bueno, si el promedio es mayor o igual que 4 y menor que 4.5; y excelente, si el promedio es mayor o igual que 4.5. Al final el algoritmo debe imprimir el número de estudiantes en cada clasificación, el promedio de notas de los estudiantes que quedaron en cada mensaje y el promedio total de notas.
3. Se tiene una competencia ciclística en la cual los competidores son clasificados por categorías dependiendo del sexo y la edad que tengan. Las categorías definidas son: categoría A, mujeres menores de 14 años; B, mujeres entre 14 y 20 años; C, mujeres entre 21 y 35 años; D, mujeres mayores de 35 años; E, hombres menores de 15 años; F, hombres entre 15 y 25 años; G, hombres entre 26 y 40 años; H, hombres mayores de 40 años. Para ello se ha grabado un archivo, llamado ‘**inscritos’**, en el cual cada registro contiene los siguientes datos: nombre, sexo y edad. El sexo está codificado así: 0 significa que es mujer, 1 significa que es hombre. Elabore un algoritmo que procese dicho archivo y que imprima cuántas personas hay inscritas en cada categoría, en cual categoría se inscribieron más personas, el total de participantes inscritos y el promedio de edad de cada categoría.
4. Se tiene un archivo llamado ‘**definitivas’**, en el cual cada registro contiene el nombre de un estudiante y las tres notas correspondientes a las notas definitivas de las tres material que cursó. Elabore un algoritmo que calcule e imprima el promedio de cada estudiante y que al final imprima el nombre del estudiante que obtuvo el mejor promedio, el nombre del estudiante que obtuvo el peor promedio y el promedio de cada una de las materias que cursaron los estudiantes.