Laura Guzmán Rojas

Jorge Alvarado Zamora

Lógica para computación

Código 3071

Guía de estudio

UNED
UNIVERSIDAD ESTATAL A DISTANCIA
Institución Benemérita de la Educación y la Cultura

Producción académica
y asesoría metodológica
Kay Guillén Díaz

Revisión filológica
María Benavides González

Diagramación
Kay Guillén Díaz

Encargado de cátedra
Roberto Morales Hernández

Esta guía de estudio ha sido confeccionada en la Uned, en el año 2011, para ser utilizada en la asignatura Lógica para computación, código 3071, que se imparte en el programa de Diplomado en Ingeniería Informática.

Universidad Estatal a Distancia
Vicerrectoría Académica
Escuela de Ciencias Exactas y Naturales

Contenido

Sobre el curso

Esta guía ha sido preparada con el fin de acompañarle, junto con el libro de texto, en el estudio de la lógica para computación. Pretende que los estudiantes, quienes inician la carrera de Informática, desarrollen el proceso lógico mental. Esta habilidad permitirá abordar las herramientas necesarias para incursionar en el mundo de la programación de computadoras.

Es aconsejable **leer estas primeras secciones con atención**, de manera que logre comenzar este camino de estudio conociendo las herramientas disponibles y lo que se espera de usted.

Objetivo general del curso

El propósito general de este curso es introducirlo en el desarrollo del proceso lógico mental y el uso de algoritmos que le permitan resolver problemas. Estos pueden ser tanto específicos como de propósito general y para su resolución se requiere una comprensión e implementación de la lógica, la cual se consigue con el estudio de técnicas actuales de desarrollo de *software* y la consideración de criterios de calidad apropiados para la solución.

Objetivos específicos del curso

En este curso se espera que usted logre:

1. Identificar la composición básica de una computadora y los diferentes componentes que la conforman.
2. Explicar los elementos de la lógica y su aplicación en las labores de programación de computadoras.
3. Elaborar algoritmos que resuelvan casos reales de programación por medio de diferentes técnicas.

4. Desarrollar diagramas de flujo con estándares a nivel internacional y con programación visual.
5. Utilizar estructuras de datos básicas, como arreglos (vectores y matrices) y estructuras de control (selección y repetición).

Material de apoyo del curso

Durante el estudio de este curso se cuenta con el siguiente apoyo bibliográfico:

Ramírez y Felipe. (2007). Introducción a la Programación. Algoritmos y su implementación en Visual Basic.NET, C#, Java y C++. México: Alfaomega Grupo Editor, S. A. de C.V.

Cátedra de Desarrollo de Sistemas– ECEN. *Orientaciones para el curso Programación Intermedia*. Costa Rica: EUNED.

Material adicional disponible en la plataforma de aprendizaje virtual.

Desglose general temas de la guía con respecto a la unidad didáctica

La presente guía consta de cinco unidades, las cuales abordan los temas del curso y corresponden a las cuatro tutorías que se brindan. Para consultar las fechas las fechas respectivas, refiérase a las *Orientaciones del curso.*

El siguiente cuadro le indica cuáles capítulos del libro corresponden a cada unidad por estudiar:

Cuadro 1. Desglose de lecturas por tema y por unidad para cada tutoría

Tema	Capítulo	Sección	Tutoría
Tema I. Definiciones básicas de computación			
	Capítulo 1. Introducción a la programación		
		Conceptos generales	1
		Lenguajes de programación	1
		Proceso de desarrollo	1
		Lógica para programación	1
		Algoritmos	1

Tema	Capítulo	Sección	Tutoría
Tema II. Introducción a la lógica de computación			
	Capítulo 2. Algoritmos y resolución de problemas		
		Elementos para solucionar problemas de pseudocódigo	1
Tema III. Técnicas de diseño de algoritmos			
	Capítulo 3. Estructuras de control		
		Selección	2
		Repetición DO … WHILE	2
		Repetición FOR	2
		Repetición WHILE	2
Tema IV. Desarrollo de diagrama de flujos			
	Capítulo 5. Diagramas de flujos		
		Diagramas de flujo	3
Tema V. Arreglos			
	Capítulo 4. Arreglos y métodos		
		Arreglos	4
		Métodos	4

Cómo estudiar este curso

Recomendaciones antes de iniciar

Probablemente usted tiene una manera de estudiar probada a lo largo de los años y sabe la calidad de resultados que obtiene. De todas maneras, nunca está de más contar con algunas recomendaciones al respecto:

- prepárese antes de comenzar a estudiar: lugar iluminado y calmo. Tenga a mano papel, lápiz, materiales y, llegado el momento, una computadora;
- intente descansar un poco antes de estudiar, pues así le será más sencillo concentrarse;
- planifique de antemano el tiempo que dedicará al estudio por semana;
- lea detalladamente la orientación del curso;

- si su estilo de aprendizaje es visual, puede subrayar las ideas principales, elaborar gráficas como mapas mentales o conceptuales, esquemas, ilustraciones comentadas, etc.;

- si es auditivo, puede ser beneficioso leer el texto en voz alta y explicar oralmente y con sus propias palabras los conceptos que estudia;

- es recomendable tener a mano una hoja o un documento electrónico para anotar todas las consultas que surjan en su estudio;

- recuerde que usted tiene la posibilidad de aclarar sus dudas en las tutorías presenciales y la plataforma virtual del curso, por medio de un correo electrónico y demás servicios del PADD, o bien llamar a la cátedra durante las horas de atención a estudiantes.

Estas recomendaciones pretenden evitar, en la medida de lo posible, frustración en el proceso, ya que, desde un inicio, debe saber a qué se enfrenta para que tome las medidas necesarias.

Conocimientos previos

Este curso forma parte del plan de estudios del Diplomado en Ingeniería Informática. Recomendamos que usted haya aprobado los cursos mencionados en el cuadro 2, pues constituyen las bases teóricas y prácticas para facilitar la comprensión de esta asignatura:

Cuadro 2. Requisitos académicos del curso

Código	Nombre
0210	Introducción a la computación

Si usted no ha aprobado los cursos mencionados, este le demandará mayor esfuerzo y más horas de estudio, así como la asimilación de gran cantidad de conceptos en muy poco tiempo.

Competencias recomendadas para lograr un mayor éxito

Debe analizar las situaciones desde diferentes enfoques para encontrar soluciones a los problemas o ejercicios planteados; también, tener paciencia porque no es común que se logre la resolución de problemas de lógica en el primer intento.

Recuerde que para conseguirlo, se empieza desde lo más básico y se documentan todos los pasos por seguir. Sea ordenado en la solución y estructure las partes de manera que cualquier persona comprenda la respuesta del problema, esto le permitirá mantener un registro adecuado y entender mejor el resultado.

En la segunda parte del curso debe contar con una computadora para instalar algún software para creación de diagramas de flujo. La cátedra recomienda el aplicativo Raptor, disponible en la plataforma, o bien descargarlo de manera gratuita desde el enlace <http://raptor.martincarlisle.com/>. Puede utilizar cualquier otro software si lo maneja previamente, crear sus diagrama en Word de Microsoft® o en forma manual. Sin embargo, el aplicativo Raptor y otros, le dan facilidades para pruebas de su solución, las cuales simplifican contribuyen al aprendizaje.

La principal recomendación para asimilar y aplicar adecuadamente los conocimientos de este curso es: **¡práctica y más práctica!**

Metodología utilizada

Por ser este un curso de índole teórico-práctica, es indispensable que cumpla con las lecturas asignadas, así como con las tareas y ejercicios propuestos, ya que lo prepararán para enfrentar trabajos individuales y grupales. También, se espera despertar en usted la inquietud por investigar y aplicar lo aprendido para que pueda desenvolverse de forma

adecuada dentro de una empresa u organización, generando productos con un acabado altamente profesional y de calidad.

Descripción de esta guía didáctica

Esta guía pretende ser un instrumento para enfrentar el curso de manera sencilla. Le brindará ayuda sobre cómo leer el libro y lograr el éxito mientras aprende a solucionar problemas de lógica para la computación y, posteriormente, programarlos.

Como bien lo indica el nombre, este documento es una guía, así que no se provee toda la información requerida para el curso, más bien, constituye un primer paso para enfrentar el amplio mundo de la programación. Recuerde: este es un método que lo irá llevando por los distintos materiales para finalizar con éxito el aprendizaje de las bases de programación.

La guía se divide en cinco unidades, que corresponden a las cuatro tutorías, de la siguiente manera:

Cuadro 3. Desglose de temas por tutoría

Tema	Tutoría
Definiciones básicas de computación	1
Algoritmo y resolución de problemas	1
Estructuras de control	2
Diagramas de flujo	3
Arreglo y métodos	4

En ellas se presentan paulatinamente todos los conceptos necesarios, para que, en conjunto y aplicados en la práctica, se resuelvan problemas de lógica para computación, incrementando, también, su nivel de complejidad.

Cada tema busca resaltar conceptos claves a los que el estudiante debe prestar especial atención. Al final de cada tema encontrará dos secciones, una de *Ejercicios de*

autoevaluación y otra de *Resolución de los ejercicios de autoevaluación*. El propósito es que el estudiante trate de responder por sí mismo los ejercicios propuestos y solo entonces, no antes, revise la solución proporcionada y autoevalúe su nivel de aprendizaje y reconozca cuáles aspectos debe reforzar.

Se le recomienda iniciar con la lectura de esta guía y abordar el libro conforme se le indique.

TEMA 1
Introducción a la programación

Sumario

- Conceptos generales
- Lenguajes de programación
- Proceso de desarrollo
- Lógica para programación
- Algoritmos

Objetivo general

Introducir conceptos básicos de manejo de computadoras y temas relevantes para la creación de algoritmos efectivos y eficientes, que cumplen con lo necesario para plantear soluciones de calidad.

Objetivos específicos

1. Identificar la composición básica de una computadora y conceptos básicos relacionados con esta área.
2. Definir qué son los lenguajes de programación y sus elementos.
3. Distinguir las diferentes clasificaciones de los lenguajes de programación.
4. Detallar el proceso de convertir el código fuente a código máquina.
5. Identificar las partes del proceso de desarrollo.
6. Distinguir la finalidad de los diferentes tipos de enunciados.
7. Comprender los diferentes principios de elaboración de enunciados.
8. Reconocer el concepto de algoritmo.
9. Ejemplificar, por medio de algoritmos, situaciones del diario vivir.

Guía de lectura

Para el estudio de este tema, usted debe realizar una lectura analítica del capítulo 1, "Introducción a la programación". Al finalizar el estudio de este, verifique que los objetivos se cumplieron y los contenidos han sido cubiertos. El siguiente cuadro le ayudará a organizar sus lecturas.

Cuadro 4. Guía de lectura del capítulo 1

Sección	Páginas
Conceptos generales	1-9
Lenguajes de programación	10-23
Proceso de desarrollo	24-27
Lógica para programación	28-35
Algoritmos	36-40

Usted puede leer las páginas indicadas en la tabla 2 para cada sección y, luego, revisar el apartado referente a cada tema, donde se recalcan y complementan conceptos y se aclaran generalidades importantes.

Desarrollo

En esta primera unidad, usted debe lograr introducirse en temas relevantes al desarrollo de programas de computación, además de conocer y comprender conceptos que aplicará más adelante cuando empiece a plantear sus soluciones de acuerdo con lo estudiado.

Conceptos generales

Cuando realice la lectura de los conceptos generales presentados en el capítulo, se espera que sea un tema de conocimiento, pero es necesario se refuerce y enriquezca.

Para abordar con detalle el tema, estudiar las páginas de la 1 a la 9 del libro.

Lenguajes de programación

Usted debe poner especial atención a las características de los lenguajes de programación y a las de un buen programa, pues son conceptos que más adelante debe aplicar.

Para aproximarse con detalle el tema, estudiar las páginas de la 10 a la 23 del libro.

Proceso de desarrollo

En el texto se mencionan seis pasos involucrados en el proceso de desarrollo, es importante que usted comprenda la diferencia entre cada uno de ellos, pues en todo momento deberá aplicarlos.

Para un acercamiento detallado al tema, estudiar las páginas de la 24 a la 27 del libro.

A. Lógica de programación

Uno de los principales propósitos de esta sección consiste en conocer y aprender la teoría general de la lógica, su aplicación a las labores de programación de computadoras y cómo automatizar procesos. Así mismo, es necesario que pueda caracterizar qué son enunciados con principios lógicos en un proceso informatizado, ya que con este conocimiento aprenderá a elaborarlos, y su relación con los datos y los procesos. También, es fundamental conocer/comprender los principios relacionados con su desarrollo, pues debe aplicarlos a lo largo del curso al plantear las soluciones a los distintos ejercicios.

Para abordar en detalle el tema, estudiar las páginas de la 28 a la 35 del libro.

B. Algoritmos

La aproximación detallada a este tema se hará más adelante y esperamos logre comprender el concepto y aplicarlo a la hora de plantear las soluciones de sus ejercicios. Por ahora, usted debe conocer el término y sus elementos.

Estudiar las páginas 36 a la 40 del libro de texto para conocer más del tema.

TEMA 2
Algoritmos y resolución de problemas

Sumario

- Estructura de datos
- Operaciones primitivas elementales
- Secuenciación

Objetivo general

Profundizar en el tema de desarrollo de resolución de problemas utilizando algoritmos.

Objetivos específicos

1. Conocer, comprender y aplicar el concepto de algoritmo en la resolución de problemas.
2. Identificar la naturaleza de los valores que representan datos para la computadora.
3. Definir, con sus propias palabras, las propiedades de los datos y el concepto de expresiones y variables.
4. Reconocer los tipos de datos soportados por los lenguajes más actuales.
5. Representar, de forma abstracta, los datos y sus dominios.
6. Identificar las categorías existentes de operadores.
7. Enunciar cuáles son los operadores aritméticos, de asignación comparativa y lógica, para la construcción de procesos computarizados.
8. Discernir en qué consisten las reglas de precedencia implícitas, posicionales y explícitas.
9. Construir expresiones complejas.

Guía para la lectura

Para el estudio de este tema, usted debe realizar una lectura analítica del capítulo 2, "Algoritmos y Resolución de Problemas". Al finalizar, verifique que los objetivos se cumplieron y los contenidos han sido cubiertos. El siguiente cuadro le ayudará a organizar sus lecturas.

Cuadro 5. Guía de lectura del capítulo 2

Sección	Páginas
Estructuras de datos	47-50
Operaciones primitivas elementales	51-66
Secuenciación	71-76

Usted puede leer las páginas indicadas en el recuadro anterior y, posteriormente, revisar la parte de conceptos relevantes, o bien puede ir leyendo el apartado de la guía y complementarlo con las secciones del libro.

Desarrollo

El principal tema por abordar en este punto es la elaboración de algoritmos, ya se introdujo el concepto en la unidad anterior, pero vamos a reafirmarlo, detallarlo más y estudiarlo en la práctica.

A. Algoritmos

Debemos recordar que los algoritmos sirven para la solución de distintos problemas computacionales; sin embargo, iniciaremos con el concepto básico de algoritmo y cómo nos permite, incluso, solucionar temas de la vida diaria. Es probable, además, que sin saber, ya haya aplicado este concepto. Un ejemplo típico son los instructivos (manuales de usuario), los cuales indican paso a paso cómo utilizar el aparato en cuestión. También podemos identificarlo en las instrucciones que recibe un trabajador por parte de su superior.

Es por ello que definimos algoritmo como un método para resolver un problema mediante una serie de pasos definidos, precisos y finitos. En este caso, *preciso* implica el orden de realización de cada uno de los pasos; *definido*, si se sigue dos veces, se obtiene el mismo resultado y *finito* tiene un número determinado de pasos, implica un inicio y un fin.

B. Medios de expresión de algoritmos

Podemos expresar los algoritmos de distintas maneras: lenguaje natural, pseudocódigo o miniespecificación, diagramas de flujo y lenguajes de programación.

Estas formas de representarlo, nos llevan en algunos casos a más detalle o a diversos modos de expresar la solución. Por un lado, el lenguaje natural es una forma de descripción de alto nivel, que establece el problema y explica de manera general, omitiendo detalles, los pasos para resolver el problema planteado.

En el caso del pseudocódigo o miniespecificación, es una descripción más formal y detallada de la solución, en la cual debemos aplicar ciertos estándares o estructuras definidas para plantear mi respuesta. Igualmente, el diagrama es una descripción formal, pero más visual y permite explicar el resultado de forma más adecuada a otra persona.

Finalmente, tenemos como resultado la implementación de una posible solución al problema, planteada en un lenguaje de programación y basada en lo que previamente se analizó en la descripción de alto nivel y formal.

Vamos a practicar la definición de algoritmos en lenguaje natural:

PROBLEMA: cambiar la llanta estallada de un automóvil, se tiene una gata hidráulica en buen estado, una llanta de reemplazo y una llave inglesa.

ALGORITMO:

INICIO

Paso 1. Aflojar los tornillos de la llanta estallada con la llave inglesa.

Paso 2. Ubicar la gata hidráulica en su sitio.

Paso 3. Levantar la gata hasta que la llanta estallada pueda girar libremente.

Paso 4. Quitar los tornillos y la llanta estallada.

Paso 5. Poner la llanta de repuesto y los tornillos.

Paso 6. Bajar la gata hasta que se pueda liberar.

Paso 7. Sacar la gata de su sitio.

Paso 8. Apretar los tornillos con la llave inglesa.

FIN

Ahora, una vez que practicamos mucho la resolución de problemas y dominamos el planteamiento de solución en lenguaje natural, podemos empezar a plantearlos en pseudocódigo o miniespecificación. Para ello, debemos conocer algunos de los conceptos básicos, como los siguientes: estructuras de datos, operaciones primitivas elementales y estructuras de control. Este último punto lo estaremos desarrollando en forma pormenorizada en la siguiente unidad.

C. Estructuras de datos

Para solucionar los algoritmos se utilizan estructuras de datos conectadas entre sí de diversas formas y definidas como conjuntos de variables, quizá de tipos distintos. Un ejemplo de una estructura de datos es el arreglo, este se estudiará más adelante.

Como ya se mencionó, las estructuras de datos se componen de variables, las cuales tienen el tipo de dato dentro de sus componentes. Podemos asignar valores a las variables con datos de entrada o de lectura. Una vez conseguidos, podemos generar

nuevas variables y aplicar distintas operaciones sobre ellas, entre las cuales están: operaciones aritméticas, operaciones lógicas, operadores comparativos y de asignación. Estas siempre deben respetar las reglas de precedencia. Como resultado de todo este proceso, se obtendrán datos de salida.

Para abordar detalladamente el tema, estudiar las páginas de la 47 a la 50 del libro.

D. Variables

Revisamos la siguiente definición: *"*Una variable es un nombre o identificador asociado con un dato. Ejemplos: apellido, teléfono, foto. Para cada variable, el programa reserva un espacio suficiente para guardar el dato en la memoria RAM. Para recuperar el dato asociado con una variable, la CPU tiene que buscar en el espacio reservado. 'Variable' indica que se puede cambiar el dato asociado con un identificador. No obstante, la ubicación en la memoria nunca cambia*"* (Jonsson, 2007-2008, film. 8).

Las variables, además, constan de tres partes: tipo de dato, nombre y contenido. Con ellas se pueden almacenar los valores que el usuario ingrese. También se utilizan para presentar los datos necesarios en una salida.

Para una aproximación pormenorizada al tema, estudiar las páginas de la 52 a la 53 del libro.

A las variables podemos aplicarles distintos operadores, dependiendo de lo que se necesita para la solución. Estos pueden ser aritméticos, lógicos, comparativos y de asignación, siempre respetando las reglas de precedencia establecidas.

Para abordar con detalle el tema, estudiar las páginas de la 54 a la 58 del libro.

E. Estructuras de control

Una de sus funciones es que “permiten modificar el flujo de ejecución de las instrucciones de un algoritmo o programa” (Universidad de Antioquía, 2009, p. 3).

Las estructuras de control se verán a lo largo del curso, aplicadas, en términos generales, a la solución de problemas computacionales; sin embargo, en cada lenguaje existen con una sintaxis particular, porque todos los lenguajes de programación poseen estructuras de control y sus diferencias radican en la forma como se escriben, es decir, en su sintaxis.

Existen tres tipos de estructura de control: de secuenciación, de selección y de repetición. Ya se verán con detalle en la siguiente unidad.

Estudiar la página 58 del libro para una aproximación más minuciosa al tema.

F. Operadores y precedencia

A continuación se muestra un resumen de todos los operadores y su jerarquía en la precedencia:

Cuadro 6. Jerarquía de operadores

Operadores	Jerarquía
( )	(mayor)
^	
*, /, mod	
=, <>, <, >, <=, >=	
No	
Y	
O	(menor)

Fuente: Cairó, 2010, p. 19.

Operadores aritméticos

Ejemplo:

30 ÷ 3 × (24 + 68 – (15 × 33 + 54) × (45 - 4 × 5) - ((45 + 30 ÷ 2) ÷ 3))

30 ÷ 3 × (24 + 68 – (495 + 54) × (45 - 20) - ((45 + 15) ÷ 3))

30 ÷ 3 × (24 + 68 – (495 + 54) × (45 - 20) - (60 ÷ 3))

30 ÷ 3 × (24 + 68 – (549) × (25) - (20))

30 ÷ 3 × (24 + 68 – 13725 - (20))

30 ÷ 3 × (- 13653)

10 × - 13653

- 136530

Como se observa en el ejemplo anterior, siempre se deben resolver las operaciones aritméticas de los paréntesis internos, pero también otras operaciones que estén entre paréntesis, siempre respetando la precedencia de cada uno de los operadores. Es decir, existe la posibilidad de resolver varias operaciones en una misma línea, respetando siempre la precedencia de paréntesis y operadores.

Operadores de asignación y comparativos

Ejemplo:

A = 5,4

Y = 8

(A > 20) < 25 – 12 × 25 < (7 + 30 ÷ 2 – 45 × (Y ^ 3)) > (90 ÷ 3) = (45 ÷ 3 × 2)

(A > 20) < 25 – 12 × 25 < (7 + 30 ÷ 2 – 45 × (512)) > (30) = (15 × 2)

(A > 20) < 25 – 12 × 25 < (7 + 15 – 45 × (512)) > (30) = (30)

A > 20) < 25 – 12 × 25 < (7 + 15 – 23040) > 30 = 30

(A > 20) < 25 – 12 × 25 < (– 23018) > 30 = 30

(FALSO) < 25 – 12 × 25 < (– 23018) > VERDADERO

FALSO) < 25 – 12 × 25 < (– 23018) > VERDADERO

(FALSO) < 25 – 300 < – 23018 > VERDADERO

(FALSO) < – 275 < – 23018 > VERDADERO

(FALSO) < FALSO > VERDADERO

FALSO > VERDADERO

FALSO

Recuerde que cuando se llega a expresiones donde FALSO se compara con VERDADERO, este siempre será mayor que FALSO.

Realice el mismo ejercicio, pero donde A = 30 y Y = 2,2

Operadores lógicos

Para realizar operaciones en las cuales existen operadores lógicos, es necesario tener clara la siguiente tabla de verdad:

Cuadro 7. Tabla de verdad de operadores lógicos

P	Q	¬P	¬Q	P v Q	P ^ Q
VERDADERO	VERDADERO	FALSO	FALSO	VERDADERO	VERDADERO
VERDADERO	FALSO	FALSO	VERDADERO	VERDADERO	FALSO
FALSO	VERDADERO	VERDADERO	FALSO	VERDADERO	FALSO
FALSO	FALSO	VERDADERO	VERDADERO	FALSO	FALSO

Fuente: Cairó, 2010, p. 19.

Ejemplo:

(45 >= 20 – 5 × 2) ^ ¬ ((35 – 45 ÷ 5 × 3 > 45) v (50 ÷ 15 × 3 <> 10)) v (17 – 2 <= 16)

(45 >= 20 – 10) ^ ¬ ((35 – 9 × 3 > 45) v (3.33 × 3 <> 10)) v (15 <= 16)

(45 >= 10) ^ ¬ ((35 – 27 > 45) v (9.99 <> 10)) v (15 <= 16)

(45 >= 10) ^ ¬ ((8 > 45) v (9.99 <> 10)) v (15 <= 16)

VERDADERO ^ ¬ (FALSO v VERDADERO) v VERDADERO

VERDADERO ^ ¬ (VERDADERO) v VERDADERO

VERDADERO ^ FALSO v VERDADERO

FALSO v VERDADERO

VERDADERO

Para abordar con detalle el tema, estudiar las páginas de la 59 a la 66 del libro.

G. Ejemplo de resolución de un algoritmo

A continuación, se le presenta un problema y su respectiva solución del algoritmo:

Problema:

Se requiere un sistema que calcule la pensión mensual de una persona con las siguientes reglas:

a. Los cálculos se realizan a partir de la siguiente información: la edad, el sexo y el estado civil de una persona.
b. Los hombres de 65 años o más reciben 120 000 colones por semana.
c. Los casados reciben 15 000 colones adicionales por semana.
d. Las mujeres de 60 años o más reciben 110 000 colones por semana sin cuota extra por estar casadas.
e. Además, cualquier hombre o mujer de 70 años o más recibe 20% más por semana del total que le corresponde.

f. A los hombres y mujeres demasiado jóvenes para una pensión se les presenta un mensaje indicándolo.

ALGORITMO:

INICIO
Paso 1: Imprimir mensaje que indica “Cálculo de pensión mensual”
Paso 2: Imprimir mensaje que indica “Digite su edad”
Paso 3: Leer Edad
Paso 4: Imprimir mensaje que indica “Digite su sexo”
Paso 5: Leer Sexo
Paso 6: Imprimir mensaje que indica “Digite su estado civil”
Paso 7: Leer EstadoCivil
Paso 8: Si edad en menor a 65 años y Sexo es igual a Masculino
entonces imprimir mensaje que indica “Demasiado joven para pensión” e ir a Paso 16.
Paso 9: Si edad en menor a 60 años y Sexo es igual a Femenino
entonces imprimir mensaje que indica “Demasiado joven para pensión” e ir a Paso 16.
Paso 10: Si Sexo es igual a Masculino y la edad es mayor o igual a 65
entonces el Salario Semanal es igual a 120 000
Sino
ir a Paso 12
Paso 11: Si Estado Civil es igual a casado
entonces Salario Semanal es igual al Salario Semanal + 15 000 e ir a Paso 13
Sino
Ir a paso 13
Paso 12: Si Sexo es igual a Femenino y la edad es mayor o igual a 60
entonces el Salario Semanal es igual a 110 000
Paso 13: Si Edad es mayor o igual a 70
entonces el Salario Semanal es igual a SalarioSemanal * 1.20
Paso 14: Calcular Salario Mensual igual a Salario Semanal * 4.33
Paso 15: Imprimir el Salario Mensual
Paso 16: Imprimir el mensaje “Fin del programa, gracias por utilizarlo…”
FIN

En el ejercicio anterior, la lectura siempre es de manera secuencial, esto significa que se lee de arriba hacia abajo. Pero cuando es requerido, se puede utilizar el salto de un paso a otro, según las condiciones.

H. Actividades adicionales

Resuelva las siguientes operaciones:

1. ¬ (1 > 1 ∧ 3 < 5 ∧ (6-4 × 4÷2 = 2) ∨ ((2 > 24) ∨ 3^2 > 10))

2. Determine el valor final de cada una de las siguientes expresiones, asuma los siguientes valores para las siguientes variables:

p = Verdadero q = Verdadero r = Verdadero s = Falso

a. $(\neg p \wedge r) \vee \neg((s \vee p \vee \neg s) \vee (q \wedge p))$

b. $\neg(r \wedge (25 > 20 \vee (q \wedge r) \vee s) \wedge (r \vee \neg((2^2 \times 6) < 134^0 \times 8)))$

Resuelva los siguientes algoritmos de forma detallada:

A. En la empresa ABC, S. A. requieren de un sistema para calcular el salario semanal de un empleado. Para ello, deben ingresar el monto por hora que gana y el número de horas trabajadas por semana. Si es mayor a 40, entonces se calcula la diferencia de horas para pagarle un 40% adicional por hora extra. Al final, el sistema debe mostrar el detalle del cálculo del salario semanal y el monto total por pagar.

   Por ejemplo:

   Total de horas semanales = 46

   Precio por hora = 5000

Resultado:

Horas normales = 40 × 5000 = 200 000

Horas extra = 6 × (5000 × 1,4) = 42 000

Total salario semanal = 242 000

B. En la empresa El Árbol, S.A. requieren calcular la planilla, para un empleado, de manera mensual. Este sistema debe ser capaz de calcular los rebajos de ley, estos son el 9,17% de la CCSS y el impuesto de renta. El impuesto de renta se calcula de la siguiente manera: si el salario es mayor a ¢500 000, se le rebaja un 10% sobre la diferencia de ¢500 000. Si el monto del salario es mayor a ¢700 000, se le rebaja el 15% sobre la diferencia de ¢700 000. Ha de considerarse que si el salario sobrepasa los ¢700 000, se le debe cobrar sobre ¢500 000 y hasta los ¢700 000 el 10% y sobre los ¢700 000 el 15%. El sistema recibe el monto bruto del salario mensual, calcular el monto de la CCSS y del impuesto de renta, rebajarlos del monto bruto para calcular el salario neto del empleado. Al final, el sistema tiene que mostrar todo el detalle de los rebajos y el cálculo final y total de salario.

Por ejemplo:

Salario mensual: 950 000

Monto CCSS= 87 115

Impuesto renta 10% = 200 000 * 0,1 = 20 000

Impuesto renta 15% = 250 000 * 0,15 = 37 500

Total de impuesto renta = 57 500

Total de rebajos = 144 615

Salario neto = 805 385

TEMA 3
Estructuras de Control

Sumario

- Selección
- Repetición DO… WHILE
- Repetición FOR
- Repetición WHILE

Objetivo general

Elaborar algoritmos utilizando adecuadamente cada una de las diferentes estructuras de control.

Objetivos específicos

1. Comprender, de manera significativa, qué son las estructuras de control y cuál es su función al aplicarlas en el diseño de algoritmos.
2. Elaborar algoritmos utilizando estructuras de:
    a. control de selección simple (IF-THEN).
    b. control de selección doble (IF-THEN-ELSE).
    c. control de selección múltiple (IF-THEN-ELSE).
3. Utilizar, adecuadamente, según corresponda, las expresiones lógicas simples (>, <, >=, <=, <>, =) y complejas (AND, OR, NOT y XOR) en las estructuras de control de selección.
4. Elaborar algoritmos utilizando estructuras de control de repetición (DO… WHILE, FOR y WHILE).

Guía para la lectura

Para el estudio de este tema, usted debe realizar una lectura analítica del capítulo 3, “Estructuras de control”. Al finalizar el estudio de este, verifique que los objetivos se cumplieron y los contenidos han sido cubiertos. El siguiente cuadro le ayudará a organizar sus lecturas.

Cuadro 8. Guía de lectura del capítulo 3

Sección	Páginas
Selección	79-98
Repetición DO… WHILE	111-118
Repetición FOR	134-138
Repetición WHILE	159-166

Usted puede leer las páginas indicadas en el cuadro 8 para cada sección y, luego, revisar el apartado referente a cada tema, en el cual se recalcan y complementan conceptos y se aclaran generalidades importantes.

Desarrollo

En este punto ya debe ser capaz de plantear algoritmos utilizando distintas secuencias de instrucciones y, quizá, haya utilizado condicionales para solucionar los problemas planteados; sin embargo, existen diferentes tipos: simples, dobles y múltiples, que podrá conocer y aplicar.

En ocasiones, además, se deben repetir sentencias una cantidad definida de veces o hasta cumplir una condición en particular. Para ello, se utilizan otras estructuras de control que no son condicionales, sino de repetición.

Debemos tener en cuenta que una estructura de control:

- tiene un único punto de entrada y un único punto de salida,
- se compone de sentencias o también de otras estructuras de control.

A. Selección

Nos permite decidir, a partir del resultado de evaluar una expresión booleana (resultado final igual a verdadero o falso), si ejecutar o no un bloque determinado u optar entre dos o más posibles.

Simple (IF-THEN)

Ejemplo de algoritmo:

INICIO

Paso 1: Establecer Edad en 0

Paso 2: Mostrar mensaje “Digite edad:”

Paso 3: Leer Edad

Paso 4: Si Edad es mayor o igual a 18 años, entonces

Mostrar “Usted es mayor de edad”

FIN

Doble (IF-THEN-ELSE)

Ejemplo de algoritmo:

INICIO

Paso 1: Establecer Edad en 0

Paso 2: Mostrar mensaje “Digite edad:”

Paso 3: Leer Edad

Paso 4: Si Edad es mayor o igual a 18 años, entonces

Mostrar “Usted es mayor de edad.”

Sino

Mostrar “Usted es menor de edad.”

FIN

Múltiple (SWITCH)

Ejemplo de algoritmo:

```
INICIO
    Paso 1: Establecer Edad en 0
    Paso 2: Mostrar mensaje “Digite edad:”
    Paso 3: Leer Edad
    Paso 4: Si Edad es mayor o igual a 18 años, entonces
                Si Edad es mayor o igual a 60 años, entonces
                    Mostrar “Usted es adulto mayor.”
                Sino
                    Mostrar “Usted es adulto.”
            Sino
                Si Edad mayor o igual a 13 años, entonces
                    Mostrar “Usted es adolescente.”
                Sino
                    Mostrar “Usted es un infante.”
FIN
```

Para abordar detalladamente el tema, estudiar las páginas de la 79 a la 98 del libro.

B. Repetición DO… WHILE y FOR

Nos permite repetir la ejecución de un bloque hasta tanto una condición booleana se vuelva falsa o se cumpla una cantidad definida de repeticiones.

Todos los ciclos de repetición tienen tres componentes necesarios:

- inicialización,
- condición de parada, y
- modificación de la condición de parada (aumento o decremento).

Repetición DO… WHILE

Ejemplo de algoritmo:

INICIO

Paso 1: Mostrar mensaje “Se imprimirán los números del 0 al 10.”

Paso 2: Establecer Contador en 0

Paso 3: Repetir

Paso 4: Imprimir “Número:” + Contador

Paso 5: Establecer Contador = Contador + 1

Paso 6: Mientras Contador no sea mayor que 10

Paso 7: Mostrar "Finaliza el programa"

FIN

Otra forma similar a la DO… WHILE es la repetición WHILE

Ejemplo de algoritmo:

INICIO

Paso 1: Mostrar mensaje “Se imprimirán los números del 0 al 10.”

Paso 2: Establecer Contador en 0, Respuesta en S

Paso 3: Mientras Contador no sea mayor que 10 O Respuesta diferente a S

Paso 4: Imprimir “Número: ” + Contador

Paso 5: Establecer Contador = Contador + 1

Paso 6: Imprimir “Desea continuar (S-N)”

Paso 7: Leer Respuesta

Paso 8: Fin del mientras

Paso 9: Mostrar "Finaliza el programa"

FIN

Repetición FOR

Ejemplo de algoritmo:

INICIO

Paso 1: Mostrar mensaje “Se imprimirán los números del 0 al 10.”

Paso 2: Establecer Contador en 0

Paso 3: Mientras Contador no sea mayor que 10

Paso 4: Imprimir “Número: ” + Contador

Paso 5: Establecer Contador = Contador + 1

Paso 6: Fin del mientras

Paso 7: Mostrar "Finaliza el programa"

FIN

La diferencia entre el WHILE y el DO… WHILE radica en que, en el primero, si la condición de parada no se cumple desde el inicio del ciclo, este nunca se ejecuta; en cambio, en el DO… WHILE, si la condición no se cumple desde el inicio, al menos el ciclo se ejecutará una vez. Para validarlo, cambie el paso 2 de ambos ejemplos, establezca el valor en 10 y analice el resultado.

Por otra parte, en el FOR, la condición de parada se da por la cantidad de repeticiones del ciclo, mientras que con el WHILE o DO… WHILE, puede ser dada por la cantidad de veces que se repetirá el ciclo o por cualquier condición que se cumpla antes de que termine.

Ejemplo de algoritmo de un ciclo con varias instrucciones dentro de él:

INICIO

Paso 1: Establecer Edad en 0
Paso 2: Repetir
Paso 3: Mostrar mensaje “Digite edad: (0 para salir)”
Paso 4: Leer Edad
Paso 5: Si Edad es mayor o igual a 18 años, entonces

Si Edad es mayor o igual a 60 años, entonces
Mostrar “Usted es adulto mayor.”
Sino
Mostrar “Usted es adulto.”
Sino
Si Edad mayor o igual a 13 años, entonces
Mostrar “Usted es adolescente.”
Sino
Mostrar “Usted es un infante.”
Paso 6: Mientras Edad sea diferente de 0
Paso 7: Mostrar "Finaliza el programa"
FIN

Para una aproximación detallada al tema, estudiar las páginas de la 111 a la 149 del libro.

C. Actividades adicionales

1. Cada uno de los algoritmos que se muestran como ejemplo, páselos a Raptor, verifíquelos y analícelos.

2. A los algoritmos de ciclos ya implementados en Raptor, cámbieles el valor de inicialización con valores más altos y más bajos, ejecútelos, analice y verifique sus resultados.

TEMA 4
Desarrollo de diagramas de flujo

Sumario

- Diagramas de flujo

Objetivo general

Aplicar el concepto de diagrama de flujo a la resolución de problemas.

Objetivos específicos

1. Comprender, de manera significativa, qué son los diagramas de flujo y cuál es su funcionalidad e importancia en el diseño de algoritmos.

2. Utilizar los símbolos específicos para el desarrollo de diagramas de flujo, de acuerdo con el estándar ANSI/ISO 5897-1985 o los definidos en la herramienta Raptor, de forma independiente.

3. Elaborar diagramas de flujo utilizando estructuras de control de selección
    a. simple IF-THEN,
    b. doble IF-THEN-ELSE, y
    c. múltiple IF-THEN-ELSE.
4. Elaborar diagramas de flujo utilizando estructuras de control de repetición
    a. DO… WHILE,
    b. FOR, y
    c. WHILE.
5. Manejar los elementos de la interfaz de Raptor.
6. Usar variables, procesos y estructuras con Raptor.
7. Utilizar puntos de interrupción en Raptor.

Guía para la lectura

Para el estudio de este tema, usted debe realizar una lectura analítica del capítulo 4, “Diagramas de flujo”. Al terminar, verifique el cumplimiento de los objetivos y que los contenidos han sido cubiertos. El siguiente cuadro le ayudará a organizar sus lecturas.

Cuadro 9. Guía de lectura del capítulo 4

Sección	Páginas
Diagramas de flujo	176-189

Usted puede leer las páginas indicadas en el cuadro 9 para cada sección; luego, revisar el apartado referente a cada tema, en el cual se recalcan y complementan conceptos y se aclaran generalidades importantes.

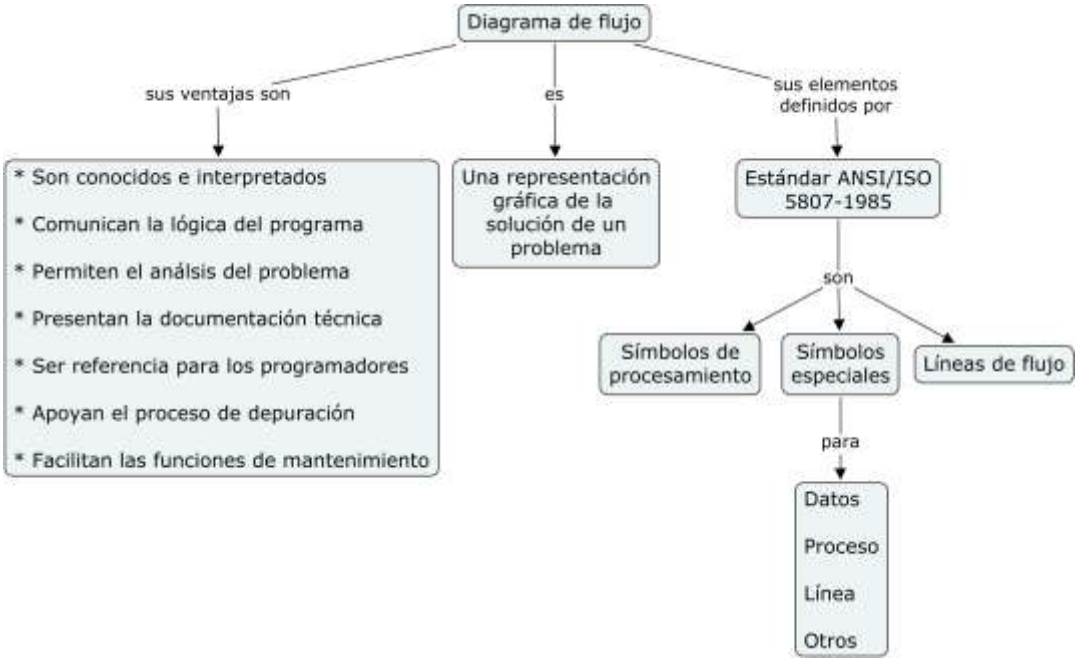
Desarrollo

Los diagramas de flujo, por excelencia, se utilizan en la programación de computadoras en economía y procesos industriales. Para su elaboración, se utilizan símbolos gráficos son universales, los cuales permiten representar los algoritmos. El énfasis del tema aquí está en su aplicación en la Informática.

El flujo que lleva un diagrama de este tipo es secuencial, conectado por flechas para visualizar la entrada, el proceso y la salida de datos. Esta distribución permite observar como un todo la solución del problema planteado, sea el algoritmo elaborado por el observador o por otra persona. El paso siguiente es traducir el diagrama de flujo al lenguaje de programación escogido (C++, C#, Java, Visual .Net, entre otros).

Una de las ventajas de los diagramas de flujo es que utilizan símbolos universales, en otras palabras, su significado a nivel internacional está normado por el estándar ANSI/ISO 5807-1985. Esta universalidad permite a los programadores documentar sus respectivos

sistemas de cómputo. De tal manera, casi cualquier programador puede comprender la solución propuesta para un determinado proceso informático.



Fuente: Morales, 2008, p. 82.
Esquema 1. Diagrama de flujo.

Para trabajar adecuadamente los diagramas de flujo, es necesario conocer y reconocer los símbolos ya establecidos.

Estudiar las páginas de la176 a la 183 del libro para un abordaje pormenorizado del tema.

A. Consideraciones iniciales

En los primeros pasos creando diagramas de flujo, debemos identificar cómo expresar la preparación de datos, la utilización de bucles, cómo mostrar y pedir datos y el manejo general de condicionales

Para estudiar con mayor detalle el tema, estudiar las páginas de la 184 a la 189 del libro.

B. Estructura de control de selección simple, doble y múltiple

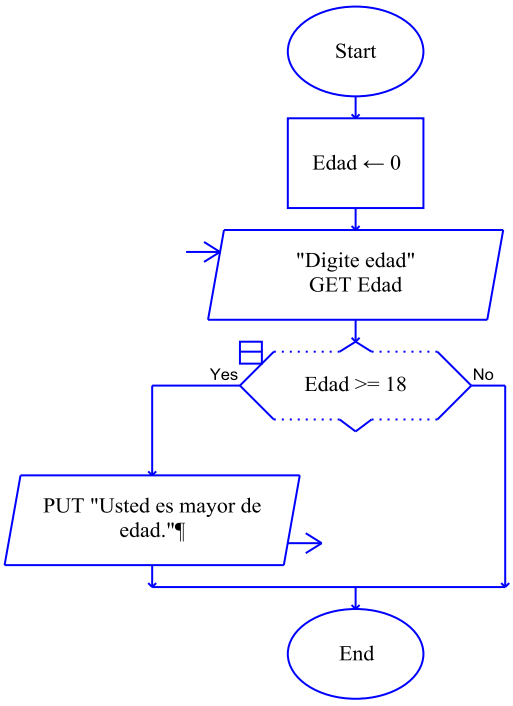
A continuación, se muestran ejemplos de diagramas de flujo correspondientes a las estructuras de control.

Simple (IF-THEN)

Ejemplo de algoritmo:

INICIO
Paso 1: Establecer Edad en 0
Paso 2: Mostrar mensaje “Digite edad:”
Paso 3: Leer Edad
Paso 4: Si Edad es mayor o igual a 18 años, entonces
Mostrar “Usted es mayor de edad”
FIN

Diagrama de flujo de datos correspondiente:

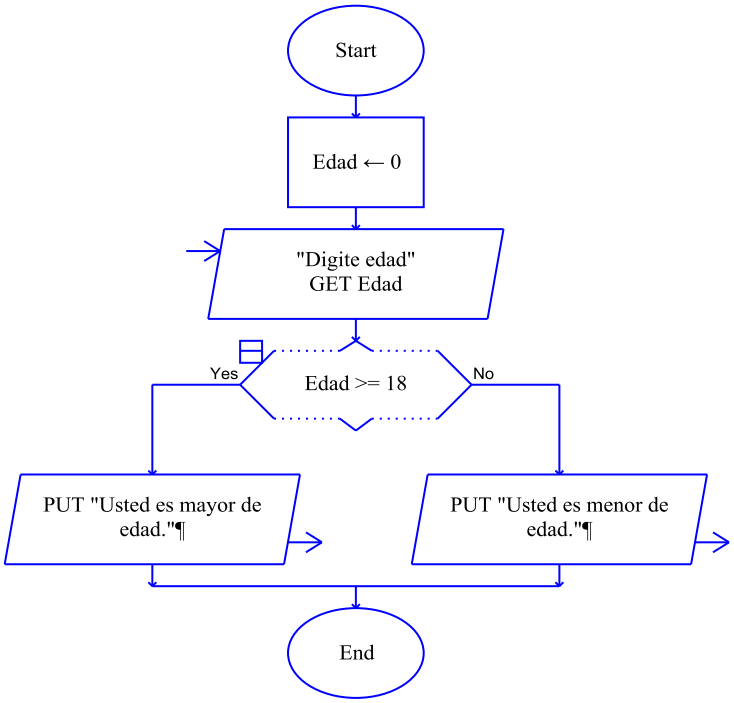


Doble (IF-THEN-ELSE)

Ejemplo de algoritmo:

INICIO
Paso 1: Establecer Edad en 0
Paso 2: Mostrar mensaje “Digite edad:”
Paso 3: Leer Edad
Paso 4: Si Edad es mayor o igual a 18 años, entonces
Mostrar “Usted es mayor de edad.”
Sino
Mostrar “Usted es menor de edad.”
FIN

Diagrama de flujo de datos correspondiente:

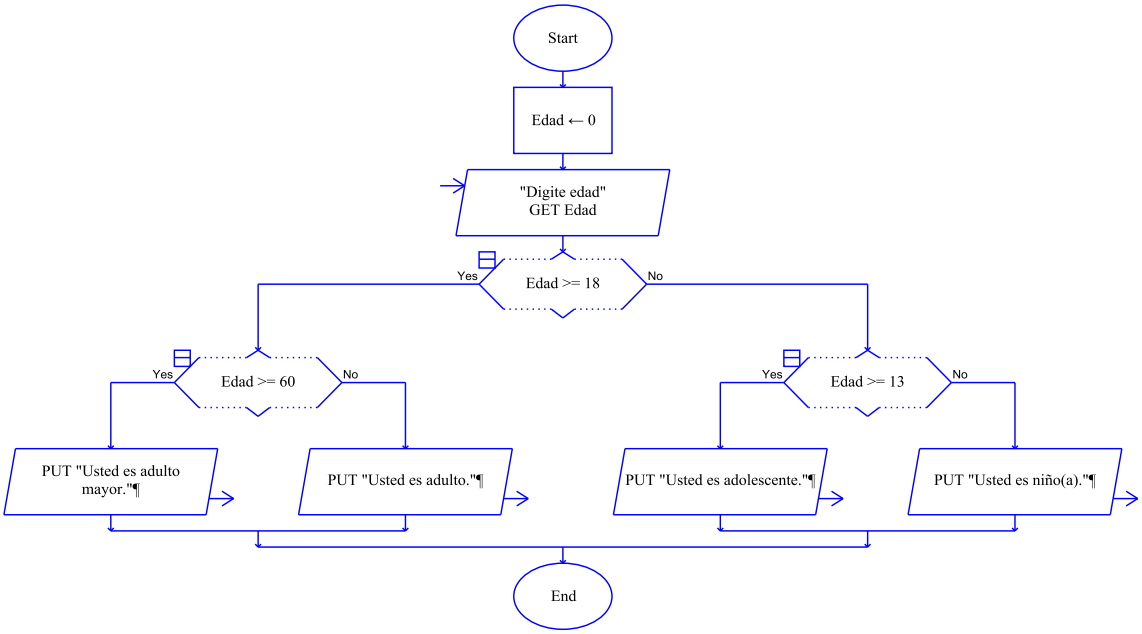


Múltiple (SWITCH)

Ejemplo de algoritmo:

INICIO
Paso 1: Establecer Edad en 0
Paso 2: Mostrar mensaje “Digite edad:”
Paso 3: Leer Edad
Paso 4: Si Edad es mayor o igual a 18 años, entonces
Si Edad es mayor o igual a 60 años, entonces
Mostrar “Usted es adulto mayor.”
Sino
Mostrar “Usted es adulto.”
Sino
Si Edad mayor o igual a 13 años, entonces
Mostrar “Usted es adolescente.”
Sino
Mostrar “Usted es niño(a).”
FIN

Diagrama de flujo de datos correspondiente:



C. Estructura de control de repetición: DO… WHILE, WHILE Y FOR

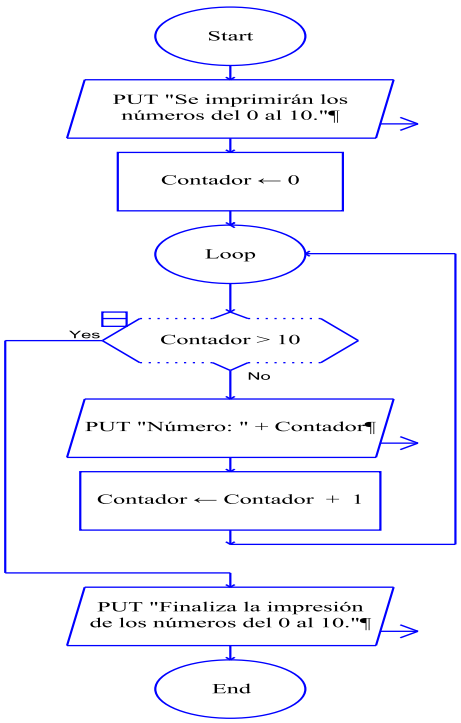
A continuación, se muestran ejemplos de diagramas de flujo correspondientes a los respectivas estructuras de repetición.

FOR

Ejemplo de algoritmo:

INICIO
Paso 1: Mostrar mensaje “Se imprimirán los números del 0 al 10.”
Paso 2: Establecer Contador en 0
Paso 3: Mientras Contador no sea mayor que 10
Paso 4: Imprimir “Número:” + Contador
Paso 5: Establecer Contador = Contador + 1
Paso 6: Fin del mientras
Paso 7: Mostrar "Finaliza el programa"
FIN

Diagrama de flujo de datos correspondiente:

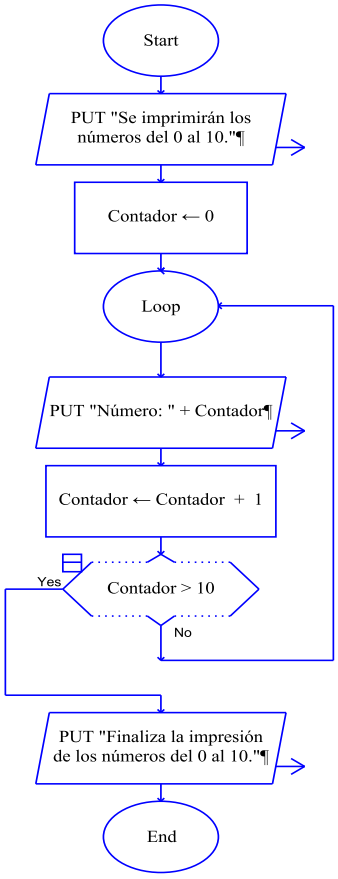


DO… WHILE

Ejemplo de algoritmo:

INICIO
Paso 1: Mostrar mensaje “Se imprimirán los números del 0 al 10.”
Paso 2: Establecer Contador en 0
Paso 3: Repetir
Paso 4: Imprimir “Número:” + Contador
Paso 5: Establecer Contador = Contador + 1
Paso 6: Mientras Contador no sea mayor que 10
Paso 7: Mostrar "Finaliza el programa"
FIN

Diagrama de flujo de datos correspondiente:



WHILE

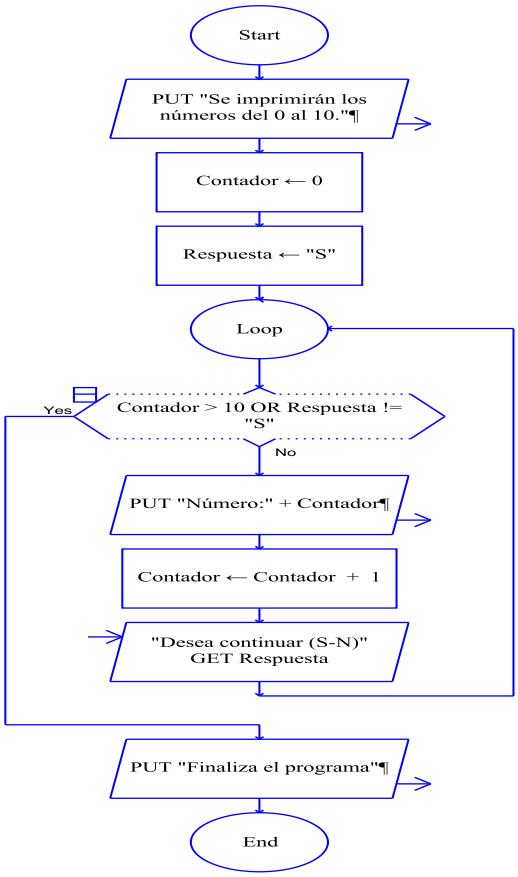
Ejemplo de algoritmo:

INICIO

Paso 1: Mostrar mensaje “Se imprimirán los números del 0 al 10.”
Paso 2: Establecer Contador en 0, Respuesta en S
Paso 3: Mientras Contador no sea mayor que 10 O Respuesta diferente a S
Paso 4: Imprimir “Número:” + Contador
Paso 5: Establecer Contador = Contador + 1
Paso 6: Imprimir “Desea continuar (S-N)”
Paso 7: Leer Respuesta
Paso 8: Fin del mientras
Paso 9: Mostrar "Finaliza el programa"

FIN

Diagrama de flujo de datos correspondiente:



La diferencia entre el WHILE y el DO…WHILE radica en que, en el WHILE, si la condición de parada no se cumple desde el inicio del ciclo, este nunca se ejecuta; por el contrario, en el segundo, si la condición no se cumple desde el inicio, al menos el ciclo se ejecutará una vez. Para validarlo, cambie el paso 2 de ambos ejemplos, establezca el valor en 10 y analice el resultado.

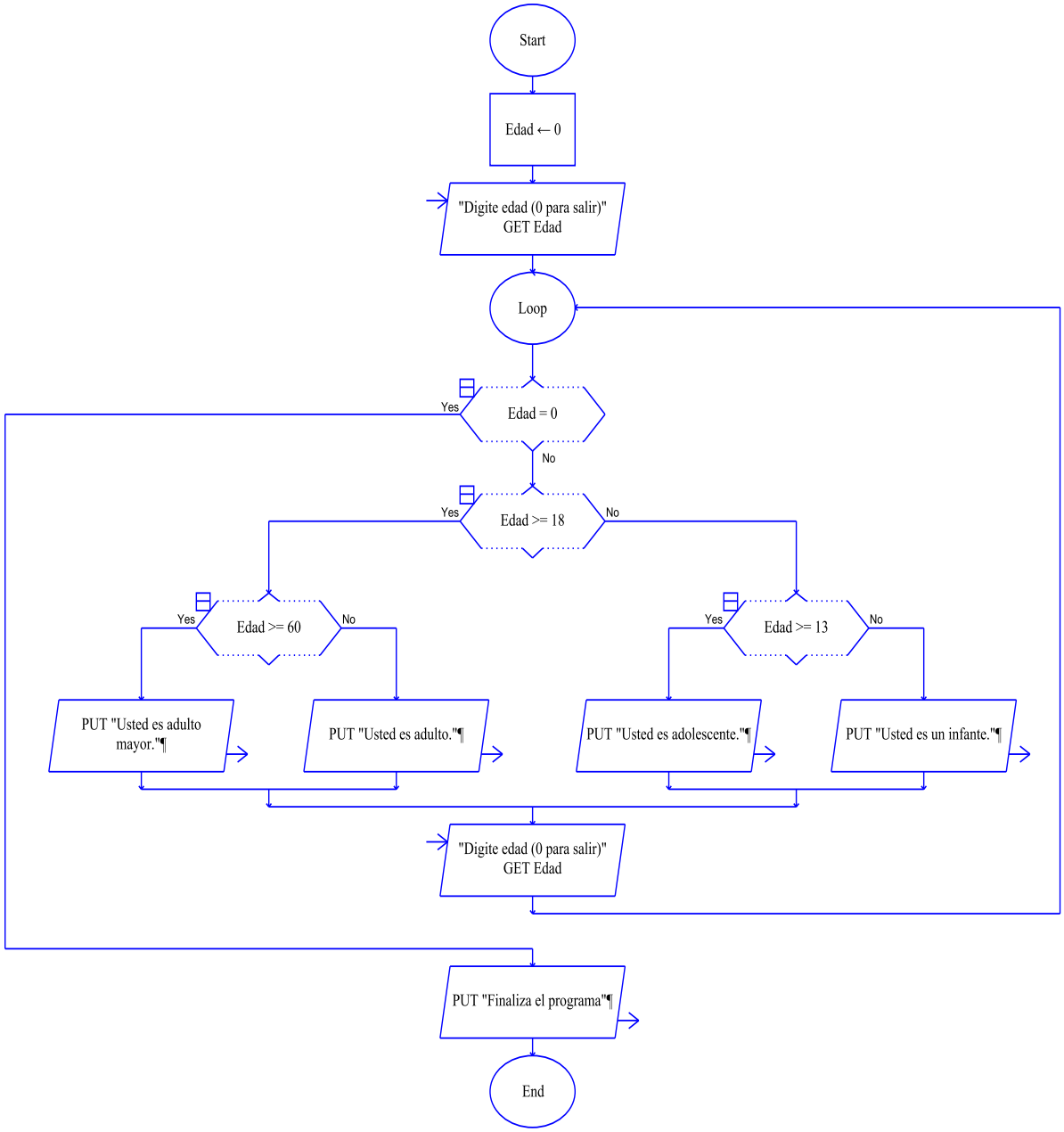
Por otra parte, en el FOR, la cantidad de veces que se repetirá el ciclo da la condición de parada; en cambio, con el WHILE o DO…WHILE, esta puede ser dada por la cantidad de repeticiones del ciclo o por cualquier condición que lo haga terminar antes antes de completar el total de ciclos iniciales.

Ejemplo de un ciclo con varias instrucciones dentro de él:

Algoritmo:

```
INICIO
        Paso 1: Establecer Edad en 0
        Paso 2: Repetir
        Paso 3: Mostrar mensaje “Digite edad: (0 para salir)”
        Paso 4: Leer Edad
        Paso 5: Si Edad es mayor o igual a 18 años, entonces
                        Si Edad es mayor o igual a 60 años, entonces
                                Mostrar “Usted es adulto mayor.”
                        Sino
                                Mostrar “Usted es adulto.”
                Sino
                        Si Edad mayor o igual a 13 años, entonces
                                Mostrar “Usted es adolescente.”
                        Sino
                                Mostrar “Usted es un infante.”
        Paso 6: Mientras Edad sea diferente de 0
        Paso 7: Mostrar "Finaliza el programa"
FIN
```

Diagrama de flujo de datos correspondiente:



D. Actividades adicionales

1. Seleccione tres de los diferentes algoritmos que ha realizado a través de toda la guía y haga lo siguiente:

    a. dibuje los diagramas a mano,
    b. páselos a Raptor,
    c. valídelos y hágales las correcciones del caso,
    d. ejecútelos y pruébelos, y
    e. cambie varios valores y analice los resultados.

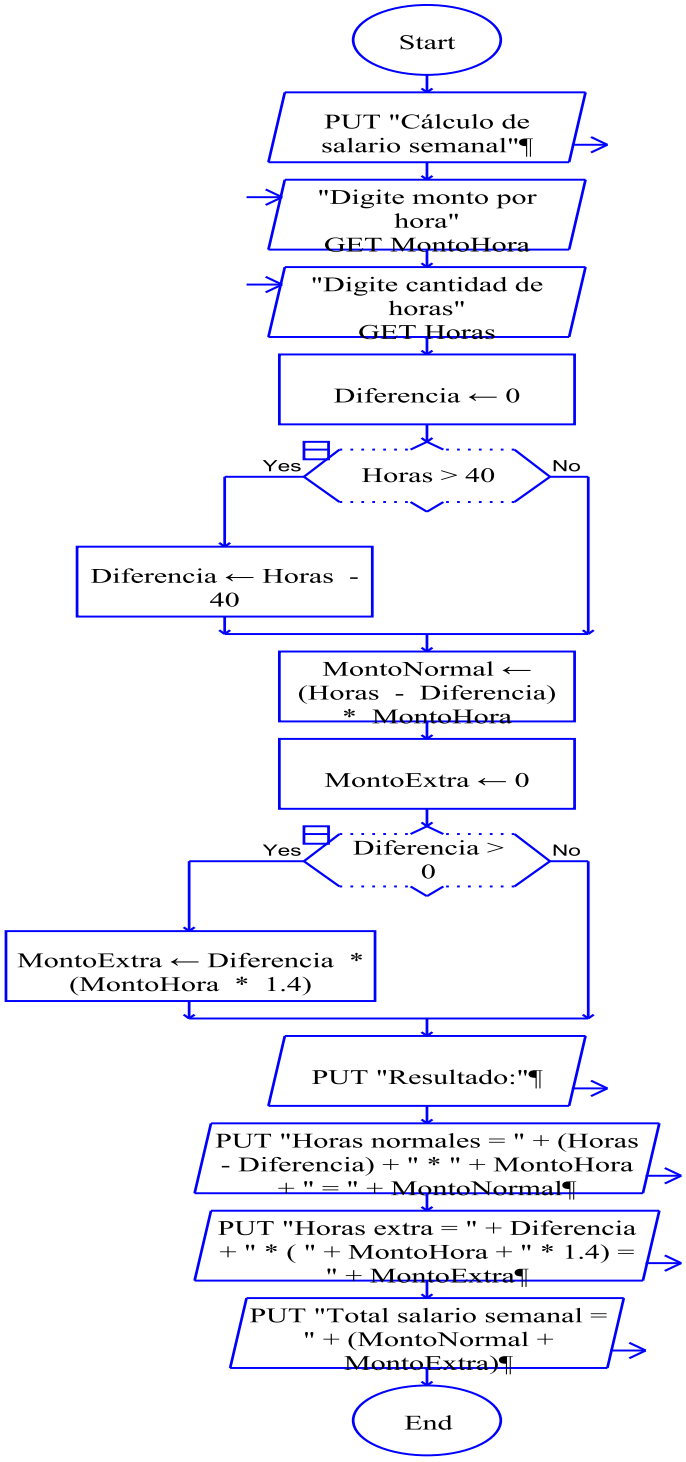
E. Guía de resolución a las actividades adicionales

A continuación se le presenta un ejemplo de un algoritmo en Raptor y su respectivo resultado:

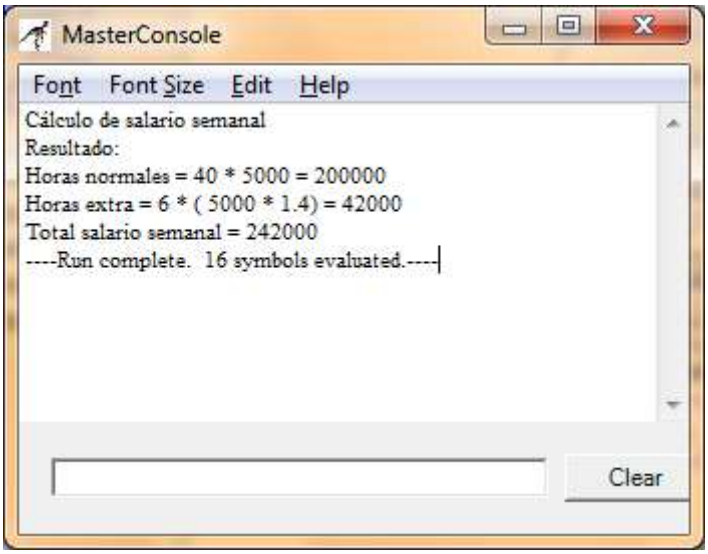
Algoritmo:

INICIO
Paso 1: Imprimir mensaje que indica “Cálculo de salario semanal”
Paso 2: Imprimir mensaje que indica “Digite monto por Hora”
Paso 3: Leer MontoHora
Paso 4: Imprimir mensaje que indica “Digite cantidad de horas”
Paso 5: Leer Horas
Paso 6: Diferencia = 0
Paso 7: Si horas es mayor que 40
entonces calcular Diferencia = Horas – 40
Paso 8: Calcular MontoNormal = (Horas – Diferencia) × MontoHora
Paso 9: MontoExtra = 0
Paso 10: Si Diferencia es mayor que 0
entonces Calcular MontoExtra = Diferencia × (MontoHora×1,4)
Paso 11: Imprimir mensaje que indica “Resultado:”
Paso 12: Imprimir mensaje que indica “Horas normales=” + (Horas – Diferencia) + “*” + MontoHora + “=” + MontoNormal
Paso 13: Imprimir mensaje que indica “Horas extra =” + Diferencia + “× (” + MontoHora + “×1,4) =” + MontoExtra
Paso 14: Imprimir mensaje que indica “Total salario semanal =” + (MontoNormal + MontoExtra)
FIN

Diagrama:


Start
PUT "Cálculo de salario semanal"¶
"Digite monto por hora"
GET MontoHora
"Digite cantidad de horas"
GET Horas
Diferencia ← 0
Yes
Horas > 40
No
Diferencia ← Horas - 40
MontoNormal ← (Horas - Diferencia) * MontoHora
MontoExtra ← 0
Yes
Diferencia > 0
No
MontoExtra ← Diferencia * (MontoHora * 1.4)
PUT "Resultado:"¶
PUT "Horas normales = " + (Horas - Diferencia) + " * " + MontoHora + " = " + MontoNormal¶
PUT "Horas extra = " + Diferencia + " * (" + MontoHora + " * 1.4) = " + MontoExtra¶
PUT "Total salario semanal = " + (MontoNormal + MontoExtra)¶
End

Resultado en Raptor:



TEMA 5
Arreglos y métodos

Sumario

- Arreglos unidimensionales
- Arreglos bidimensionales
- Métodos que no regresan valor
- Métodos que regresan valor

Objetivo general

Aplicar concepto de arreglos y métodos en las distintas resoluciones de problemas.

Objetivos específicos

1. Entender en qué consiste la estructura de datos denominada arreglo y sus dos posibles tipos: unidimensionales (de una dimensión) y bidimensionales (de dos dimensiones).
2. Utilizar las estructuras de datos, unidimensional y bidimensional, en la resolución de problemas.
3. Comprender la estructura general de un método y la importancia de su utilización en el planteamiento de resolución de problemas.
4. Definir métodos que pueden ser utilizados en la resolución de más de un problema.
5. Utilizar métodos que no regresan valor y que si regresan valor.

Guía para la lectura

Para el estudio de este tema, usted debe realizar una lectura analítica del capítulo 5, "Arreglos y métodos". Al finalizar el estudio, verifique que los objetivos se cumplieron y los contenidos han sido cubiertos. El siguiente cuadro le ayudará a organizar sus lecturas.

Cuadro 10. Guía de lectura del capítulo 5

Sección	Páginas
Arreglos	190-212
Métodos	213-222

Usted puede leer las páginas indicadas en el cuadro 6 para cada sección y, luego, revisar el apartado referente a cada tema, en el cual se recalcan y complementan conceptos y se aclaran generalidades importantes.

Desarrollo

A. Definición de arreglo

Un arreglo es un área reservada en memoria, el cual cuenta con múltiples espacios (celdas) para almacenar valores.

Recuerde que una variable es un espacio reservado en memoria, en este solo puede almacenarse un único valor en un momento dado, pero un arreglo tiene varios espacios reservados y cada uno de ellos guarda un valor, por ende un arreglo puede almacenar diversos valores al mismo tiempo.

B. Tipos de arreglo

Los arreglos pueden ser unidimensionales (vectores) o multidimensionales (matrices). Los vectores se visualizan como una hilera de celdas y las matrices como una cuadrícula.

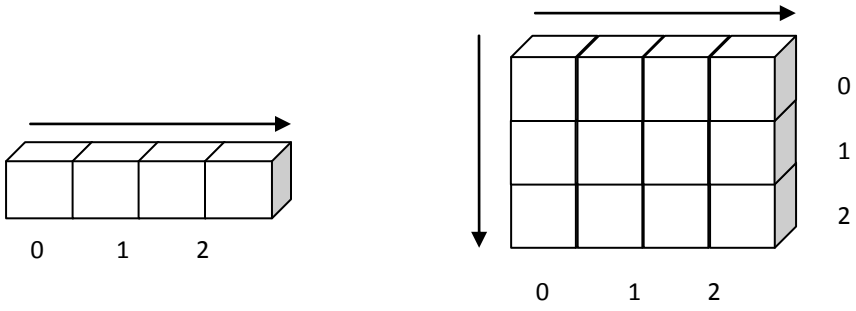


Figura 1. Tipos de arreglos

C. Características

Los arreglos presentan las siguientes características:

- **Poseen un nombre:** este se rige por los mismos principios de las variables.
- **Tienen longitud:** se refiere a la cantidad de celdas que los componen. Dicha longitud se debe especificar cuando se declara el arreglo.
- **Tienen tipo:** al declararlos, se debe especificar el tipo de arreglo, si es entero, doble o cadena.
- **Almacenan valores del mismo tipo:** si un arreglo se declara de tipo entero, entonces solo podrá almacenar valores de este tipo, lo mismo ocurre al declararlo doble o cadena.
- **Se relacionan con un índice:** al arreglo se le asocia una variable de tipo entero, la cual funciona como un índice que sirve para señalar las celdas.

D. Declaración

Los arreglos se declaran de forma similar a las variables. Primero, se define el nombre del arreglo; seguido y entre paréntesis, se crea su longitud; después se define el tipo de dato del arreglo, ya sea entero, decimal (*double*) o cadena y, finalmente, determinamos el dominio.

Declaración de un vector que almacena números enteros:

Notas: Arreglo [4] Enteros

Declaración de una matriz que almacena números decimales:

Notas: Arreglo [4] [3] Decimales

En la declaración, al igual que cualquier variable, hay un nombre significativo relacionado con lo que se va a almacenar (Notas), la palabra reservada "Arreglo" y luego el tamaño dependiendo de si es un vector (una dimensión) o una matriz (dos dimensiones). Finalmente, definimos el tipo de dato por almacenar en el arreglo.

E. Asignación de valor

Para asignar valor a un arreglo debemos especificar la celda en la cual deseamos esté dicho valor, esto se hace mediante el índice, que se encierra entre paréntesis, justo después del nombre del arreglo. El índice es el número de celda donde queremos almacenar un valor.

Para asignar un valor a un vector se ejecuta la siguiente instrucción:

Notas[2] = 9

En este caso, le estamos asignando al vector llamado Nota el valor 9 en su celda número dos. El dos entre paréntesis es el índice.

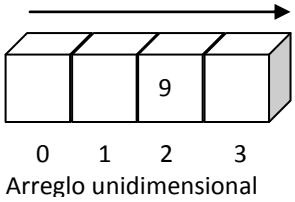


Figura 2. Asignación de valor a un arreglo

Para asignar un valor a una matriz, debemos considerar que, en vez de un valor para la celda correspondiente, asignamos dos valores, los cuales representarán la fila y columna deseadas, la instrucción sería la siguiente:

Notas[2] [3] =95

Así, le estamos asignando a la matriz llamada Nota el valor 95 en la celda que coincide con la fila dos y la columna tres. Los números dos y tres entre paréntesis son los índices.

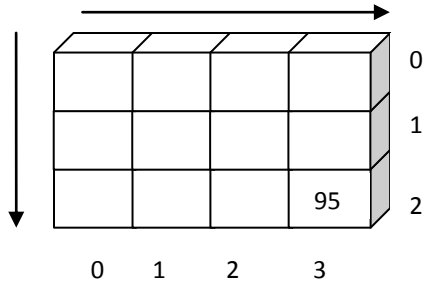


Figura 3. Asignación de valor a una matriz

Para abordar con detalle el tema, estudiar las páginas de la 190 a la 197 del libro.

F. Definición de método

En programación, los métodos nos ayudan a organizar el planteamiento de las soluciones, esto se logra segmentando el problema y planteando por aparte ciertas funcionalidades. Si estas son definidas adecuadamente, incluso pueden reutilizarse para otras soluciones, de manera que hace más eficiente nuestro trabajo.

Sin embargo, al emplear los métodos, debemos tener claro que no es tan simple como solo dividir la solución en partes, también es necesario analizar cómo definir aquellos con funciones específicas y autónomas.

G. *Tipos de métodos*

Existen básicamente dos tipos de métodos: los que no devuelven o si valor a donde son llamados.

- **Métodos que no devuelven valor:** realizan una funcionalidad en particular y no devuelven un resultado al finalizar las acciones.
- **Métodos que devuelven valor:** efectúan una funcionalidad en particular y devuelven un resultado al terminar las acciones.

H. *Estructura y declaración de métodos*

La estructura general de un método debe tener dos partes: nombre significativo y parámetros (puede no tenerlos); en el caso de los métodos que devuelven valor, cuatro: nombre significativo, parámetros (puede carecer de estos), indicador de tipo que devuelve y retornar el resultado correspondiente.

Métodos que no devuelven valor

```
Método NombreSignificativo (Val parámetro1: Entero,
                            parámetro2: Carácter)
    Acciones
    .
    .
    .
Fin Método NombreSignificativo
```

Métodos que devuelven valor

```
Método NombreSignificativo (Val parámetro1: Entero,
                            parámetro2: Carácter): Entero
    Declaraciones
        Variables
            resultado:Entero
```

```
        Acciones
        .
        .
        .
        Return resultado
    Fin Método NombreSignificativo
```

I. Utilización o llamado de métodos

Una vez definidos los métodos que necesitamos para la solución al problema planteado, hacemos el llamado o utilización de este de la siguiente manera:

Métodos que no devuelven valor

```
    Método Principal
        Declaraciones
            Variables
                parámetro1: Entero
                parámetro2: Carácter

        Acciones
        .
        .
        .
        NombreSignificativo (parámetro1, parámetro2)
        .
        .
        .
    Fin Método Principal
```

Métodos que devuelven valor

```
    Método Principal
        Declaraciones
            Variables
                resultadoCalculo:Entero
                parámetro1: Entero
                parámetro2: Carácter
```

Acciones

.

.

.

resultadoCalculo = **NombreSignificativo** (**parámetro1**, **parámetro2**)

.

.

.

Fin Método Principal

NOTA: para efectos del curso, se asume que todos los parámetros se pasan por valor, no hay parámetros por referencia, esto se ve en detalle en los próximos cursos de programación.

Para una aproximación detallada del tema, estudiar las páginas de la 213 a la 222 del libro.

J. Actividades adicionales

1. Investigue en la Web sobre el método de ordenamiento de burbuja en un vector o arreglo unidimensional. Escriba su algoritmo; luego, haga el diagrama de flujo para este caso, pruébelo y verifíquelo.

2. Realice solo el diagrama de flujo en Raptor, que sume, independientemente, los elementos positivos y negativos de la siguiente matriz:

-10	22	31
42	-51	-9
20	73	88
4	1	-6

3. Para el siguiente enunciado, elabore el algoritmo y luego implemente el diagrama de flujo:

Un centro médico privado requiere una aplicación informática para la distribución de los pacientes en sus cinco pisos y veinticinco habitaciones (hay cinco habitaciones por piso).

La clínica tiene un edificio donde se atienden cinco áreas: Urgencias médicas (piso 1), Medicina general (piso 2), Cardiología (piso 3), Hepatología (piso 4) y un sector preferencial (piso 5).

Conforme los pacientes van llegando al centro médico se solicitan sus datos, se genera un código temporal del paciente (solo por la estadía en la clínica), se hace un primer diagnóstico, se le asigna un área de la clínica (número de piso), un número de habitación (del 1 al 5) y se calcula y almacena el total de pacientes internados por piso.

En la clínica hay un período del día para ingreso de pacientes, por lo que, al finalizar la tarde, se genera un reporte de pacientes internados.

El programa debe realizar lo siguiente:

- Solicitar la edad del paciente.
- Quien registra el ingreso del paciente debe digitar el área de donde proviene el diagnóstico preliminar (el programa no hace el diagnóstico, solo se digita el área) y el programa determina el piso.
- Con el área (número de piso) y la habitación (número consecutivo de 1 a 5) se genera el código numérico del paciente (ejemplo: piso 2, habitación 3: código del paciente es el número 23).
- Se debe preguntar si se desea incluir más pacientes.
- Al finalizar el ingreso de pacientes, se genera el reporte final que contiene los siguientes datos: ubicación (piso) y código del paciente. En la última línea del reporte, se debe mostrar la cantidad de pacientes

internados en todo el centro médico y los pisos con más y menos pacientes.

- Un especialista en Gerontología llega todas las tardes a la clínica y se le debe generar un reporte de pacientes de la tercera edad, ordenado ascendente o descendentemente por edad (según lo desee el especialista). Los datos del reporte para el especialista son: código del paciente y edad.
- Se debe habilitar una consulta de habitaciones disponibles por piso, si el especialista indica un piso, se le debe mostrar la primera habitación disponible (sin paciente) de ese piso, esto por si eventualmente el médico decide reubicar al paciente (en este programa no se debe hacer la reubicación).

Para la solución debe:

- Utilizar un arreglo multidimensional (matriz) en el cual se asigne, en la primera columna, el número de piso.
- De la segunda a la sexta columnas, asignar el código de paciente, y en la última, debe aparecer la cantidad de pacientes del piso.
- En la habitación sin paciente, escribir un cero para indicar que está sin asignar o ya se dio de alta.

	Habitaciones					Total de pacientes por piso
Pisos						
5	51	52	53	54	55	5
4	0	0	0	0	0	0
3	31	0	0	34	0	2
2	21	22	23	24	25	5
1	11	0	13	14	15	4

- Si la edad del paciente es mayor a 70, se debe incluir el código del paciente y su edad (pacientes de la tercera edad) en un arreglo bidimensional.

1	2	3	4	5	6	...	25
11	**23**	**24**	**52**	**53**	**54**	**...**	
85	**70**	**91**	**2**	**80**	**84**	**...**	

- Si en un área de la clínica se trata de internar a más de 5 pacientes, se debe desplegar el mensaje “Lo sentimos, no hay espacio disponible en esta especialidad”.

Tome en cuenta los siguientes aspectos:

- El código del paciente está compuesto por dos números.
- El sistema asigna el piso con base en el área digitada.

Referencias

Cairó, Osvaldo. (2010). *Metodología de la programación algoritmos, diagrama de flujos y programas*. (3. ª ed). México: Editorial Alfaomega Grupo Editor, S. A.

Jonsson, Anders. (2007-2008). *Datos y variables.* Consultado en <www.dtic.upf.edu/~jonsson/fp07/datos.ppt>

López, L. y Ramírez, F. (2011). *Lógica para computación.* México: Editorial Alfaomega Grupo Editor, S. A.

Morales H., R. (2008). Guía de estudio para el curso Lógica para computación, código: 3071. UNED.

Universidad de Antioquia. (2009). *Introducción a las computadoras. Estructuras de control repetitivas en C.* Consultado en <http://ciencias.udea.edu.co/programas/pregrado/CNM-130/docs/clase9sinp.pdf>