P-Manuals T-II

Manual de Prácticas de Programación

Rosa María, ZÚÑIGA-RUBIO *Coordinador*

ECORFAN®

Manual de Prácticas de Programación

Primera Edición

Maricela MARMOLEJO-HARO
Ana Begoña ESPINOZA-LARIOS
Oscar Arturo ROBLES-MARTÍNEZ
Rosa María, ZÚÑIGA-RUBIO

Universidad Autónoma de Nayarit

ECORFAN-México

Manual de Prácticas de Programación

Coordinador
ZÚÑIGA-RUBIO, Rosa María

Autores
MARMOLEJO-HARO, Maricela
ESPINOZA-LARIOS, Ana Begoña
ROBLES-MARTÍNEZ, Oscar Arturo
ZÚÑIGA-RUBIO, Rosa María

Diseñador de Edición
SORIANO-VELASCO, Jesús. BsC.

Producción Tipográfica
TREJO-RAMOS, Iván. BsC.

Producción WEB
ESCAMILLA-BOUCHAN, Imelda. PhD.

Producción Digital
LUNA-SOTO, Vladimir. PhD.

Área de Conocimiento
Unidad Académica del Sur-Ixtlán

Unidad Académica
Informática

Academia
Programación

Editora en Jefe
RAMOS-ESCAMILLA, María. PhD

ISBN: 978-607-8534-09-8
Sello Editorial ECORFAN: 607-8534
Número de Control PM: 2017-02
Clasificación: PM (2017):060616-0102

Ciudad de la Cultura Amado Nervo. Boulevard Tepic-Xalisco S/N C.P. 63190 Tepic, Nayarit. México.

Contenido

Prólogo

La Academia de Programación de la Universidad Autónoma de Nayarit – Extensión Ixtlán, pone a disposición de estudiantes y docentes principalmente, el Manual de Prácticas de Programación I, como herramienta de apoyo en la enseñanza – aprendizaje. Utilizando el lenguaje C en los ejemplos y la base de casos almacenada en una CD que acompaña al manual. A fin que permita al estudiante adquirir y ejercitar las competencias pertinentes al ámbito de programación en la informática, así como utilizar las habilidades adquiridas para el desarrollo de aplicaciones que contribuyan a la solución de diversos problemas mediante la computadora. Se ha considerado pertinente realizar un conjunto de prácticas que se puedan utilizar en la licenciatura en Informática, Sistemas Computacionales o en cualquier programa educativo que aborde esta temática, en un nivel superior o medio superior, respondiendo principalmente a necesidades manifestadas en nuestro contexto, pero sabedores de la problemática similar que enfrentan otras universidades. En ese tenor, se decide compartir el presente material.

El Manual de Prácticas es una herramienta de apoyo a las clases presénciales, a la vez que favorece y hace una aportación sustancial a la educación a distancia en materia de Programación, utilizando para eso el lenguaje de programación C, tiene como fin desarrollar las habilidades de los aprendizajes de programación I, en los temas de Introducción a la programación, Estructuras selectivas, Estructuras Repetitivas, Funciones, Arreglos Unidimensionales y Bidimensionales o Matrices. Se pretende que el docente y alumno tengan al alcance un material que puedan utilizar en el momento que se estén desarrollando los temas de esta unidad de aprendizaje, pero que auxilie al estudiante aún sin la presencia del maestro o asesor del tema de ser necesario, para ello se han incluido diversos casos de problemas que pudieran surgir al realizar un programa y algunas de las posibles soluciones.

En el proceso enseñanza aprendizaje de esta materia es muy importante contar con material didáctico que permita ver diversos ejemplos completos de los programas, en donde el alumno pueda retomar el tema para realizar la práctica y ejercicios complementarios. Esperamos que el lector encuentre en el manual una herramienta útil, para desarrollar o reforzar los saberes de las diferentes temáticas que se abordan en la introducción a la programación mediante el uso del Razonamiento Basado en Casos.

Academia de Programación de la UAN – Extensión Sur

Introducción

Hoy en día se pueden agilizar los procesos manuales y automatizarlos, ahorrando con ello tiempo, dinero y esfuerzo, todo eso gracias a la utilización de los lenguajes de programación, los cuales pueden ser de enfoques de bajo, medio y alto nivel.
El manejo de un lenguaje de programación en particular, permite codificar y dar solución a determinados problemas, implicando con ello la utilización de la lógica de programación por parte de los alumnos y el conocimiento de los diferentes tipos de datos, estructuras de control, repetición, anidamientos, arreglos, funciones, entre otros conocimientos básicos requeridos.

"Un profesor comprometido con la educación busca actualizarse, conocer y proporcionar herramientas que faciliten el aprendizaje a sus alumnos, resolver problemas o suplir carencias con creatividad. Es en esa dinámica que surge el proyecto abordado en este documento.

Entre las materias más difíciles de cursar en las licenciaturas de informática, se encuentran las que requieren el desarrollo de la lógica de programación, área en que el estudiante demanda atención por parte de los docentes, tutores o asesores que le faciliten u orienten en la adquisición del conocimiento de la materia" (Zúñiga, 2006).

Como docentes de educación superior, trabajar con unidades de aprendizajes relacionadas con la programación implica un gran reto, porque no todos los alumnos desarrollan la lógica requerida en la licenciatura en informática sólo con el trabajo dentro del aula.

"Lo ideal sería que cada estudiante tuviera a su alcance un profesor (experto) en el momento y durante el tiempo requerido, que le asesorara para resolver problemas, lo cual resulta imposible, pues el número de estudiantes siempre rebasará con mucho al de asesores disponibles en el momento y lugar que se les necesita. Esto afecta el desempeño del estudiante, pues al carecer de la información relevante que contribuya a gestionar su conocimiento se desliga un deficiente nivel de programación y reprobación de la materia" (ídem).

Lo anterior marcó la pauta para que los docentes que participan en la Academia de Programación de la Unidad Académica del Sur - Ixtlán optaran por realizar un material que permita al alumno conocer, comprender y practicar las diferentes temáticas, relacionadas con aprender a programar; se analizó la necesidad de diseñar un manual que contemple teoría y práctica, dividido en ejercicios resueltos y otros sólo propuestos, que permitan al alumno comprender y practicar las temas que se van trabajando en el aula de forma extra clase.

El Manual de Prácticas de Programación aborda temáticas como: tipos de datos, identificadores y operadores aritméticos, lectura y escritura de datos, estructura de control selectiva: Si-Entonces (If), operadores relaciónales y lógicos, estructura de selección múltiple: casos- switch, estructuras de control cíclicas: Mientras (while), Desde o Para (for), y Hacer...Mientras (do while), funciones con y sin retorno: Sin parámetros y realizadas por el usuario, funciones con retorno y con parámetros: funciones sin retorno y con parámetros realizadas por el usuario, arreglos unidimensionales y arreglos bidimensionales.

En cada uno de los diferentes temas se ofrece una introducción o explicación de la parte teórica, se especifican una serie de ejercicios resueltos y por último a manera de ejercicios propuestos se da la oportunidad al alumno de retroalimentar.

Se espera que éste material contribuya al desarrollo de la lógica de programación por parte del alumno y que los ejercicios resueltos marquen la pauta para que pueda resolver los ejercicios propuestos, sin olvidar que la práctica de las diferentes temáticas permitirán que el alumno se familiarice y comprenda la forma de programar y resolver problemas utilizando un lenguaje de programación, en éste caso el lenguaje C.

Pasos a realizar:

Las prácticas deben de realizarse en el orden propuesto en este documento, para mayor efectividad:

1. La lectura de comprensión es obligatoria para dar solución a los problemas propuestos, por eso, antes de intentar realizar los ejercicios se presentan de manera resumida las bases teóricas indispensables del curso.

2. Después de leer y comprender la temática abordada en la práctica, se procede a dar solución a cada uno de los problemas planteados. Pero, si aún así existen interrogantes en el estudiante, este puede consultar la base de casos de posibles dificultades anexa al Manual de Prácticas en el CD que se encuentra al final del documento.

3. Al menos en el primer problema propuesto en cada práctica, se cuenta con una pequeña base de casos de dificultades que pudiera encontrarse el estudiante al tratar de darle solución al problema, estos casos se presentan en un archivo de Microsoft Power Point a los cuales el estudiante tiene acceso en este mismo material. Se consulta si se tiene problemas en la elaboración de código del archivo fuente, en donde se presenta un caso igual o similar al que tiene el estudiante al hacer su programa y propone una o varias soluciones, dependiendo del problema que se trate. Sin embargo, la recomendación es consultarla aunque haya realizado exitosamente el programa, ya que podrá conocer la forma de dar solución a una problemática que posiblemente se le presente al realizar otro problema.

Todos los elementos abordados en esta página son los mismos que se aplican en cada una de las prácticas:

Explicación para el desarrollo de la práctica

El estudiante procederá a realizar su práctica solucionando el problema mediante un lenguaje de programación específico, en este caso C, consultando la base de casos tantas veces como lo considere necesario. Una vez concluida la práctica el estudiante informará al facilitador para que éste la revise, la registre en la lista y más tarde reciba el reporte de la práctica realizada, pero en caso de no estar presente el facilitador se sugiere almacenarla en un soporte electrónico y más tarde enviar el archivo fuente a la dirección de correo que con anterioridad le entregó el facilitador para que este lo revise, registre y si lo considera necesario pregunte al estudiante sobre la forma en que dio solución al problema. Aclarando que lo expresado en esta página no necesariamente aplicará para quienes utilizan en material de manera independiente a la unidad de aprendizaje que se hace referencia.

Material:

- Libreta de anotaciones
- Pluma
- Computadora
- Software para programar
- Memoria de almacenamiento secundaria

Sistema de evaluación

La práctica será integrada al portafolio de evidencias del estudiante. Se calificará bajo los siguientes criterios: puntualidad en la entrega, pertinencia, ejecución correcta, documentación. En su totalidad al portafolio le corresponde un 40% de la calificación final, pero se ponderará el porcentaje en base al cumplimiento de los criterios antes mencionados.

Tabla 1 Rubrica Trabajo de Portafolio

	R1	**R2**	**R3**
Criterios	**60%**	**80%**	**100%**
Puntualidad en la entrega			
Pertinencia			
Correcta ejecución			
Documentación			

- Obtendrá un 60% si la solución planteada en el programa es pertinente al problema señalado y si se ejecuta correctamente (sin errores), pero es entregado en un nuevo plazo señalado por el mismo facilitador.

- Obtiene un 80% si la solución planteada en el programa es pertinente al problema señalado, si se ejecuta correctamente (sin errores) y se entrega en fecha y hora solicitada por el facilitador, pero no está documentado.

- Obtiene un 100% si la solución planteada en el programa es pertinente al problema señalado, si se ejecuta correctamente (sin errores), si se entrega en fecha y hora solicitada por el facilitador, si es documentado.

Práctica 1 Tipos de Datos, Identificadores, Operadores Aritméticos, Lectura y Escritura de Datos

Objetivo específico de la práctica

Que el estudiante identifique, declare y use correctamente los tipos de datos, variables, constantes, los formatos de entrada o salida de estos, así como la lectura y escritura de datos, uso de los operadores de asignación y aritméticos en los programas que construya.

Criterios de desempeño

El estudiante estará capacitado para identificar, declarar y utilizar los tipos de datos, variables, constantes, operadores de asignación y aritméticos cuando:

- Aplique las reglas de identificadores en la declaración de las funciones, variables y constantes.
- Identifique una variable, constante o función mal y bien declarada.
- Diferencie una variable de una constante.
- Distinga los tipos de datos y sus formatos.
- Utilice adecuadamente los formatos de entrada o salida de los diversos tipos de datos.
- Manipule correctamente las funciones de entrada/salida de datos: scanf(), printf(), puts() y gets(), para leer y escribir.
- Relacione y utilice correctamente identificadores de variable y/o constantes en operaciones aritméticas.
- Resuelva problemas mediante un programa informático en que utilice correctamente las instrucciones de E/S ya señaladas, los identificadores, tipos de datos y operadores de asignación y aritméticos.

Resultado esperado en relación a los criterios de desempeño específicos de la práctica

- La conceptualización por parte del estudiante de los términos variable, constante, tipos de datos, operador de asignación y operadores lógicos.
- Dominio de la sintaxis – formato para usar las instrucciones que permitan declarar y utilizar funciones elementales de e/s, tipos de datos, variables y constantes, así como operadores aritméticos.
- La ejemplificación de las partes de un programa.
- La capacidad de elegir el tipo de datos más adecuado, variable o constante requerida.
- Demostración de la habilidad del estudiante para identificar, declarar y usar identificadores válidos, mediante su aplicación en un programa informático.
- La modificación – corrección de errores de sintaxis relacionados con los tipos de datos e identificadores en un programa informático.
- Calcular y asignar resultados a variables mediante operaciones aritméticas.
- Utilizar funciones para leer y escribir.

Introducción

Cuando se habla de “Programación Informática”, se hace referencia al proceso de planificar y desarrollar la resolución de un problema mediante una computadora. En su sentido más simple se trata de diseñar y codificar un programa que al ejecutarse en una computadora, resuelva un problema (CETTICO;1999; p.41).

En general un programa es un conjunto de instrucciones que se le dan a la computadora para que haga algo, le dicen a la computadora cómo ejecutar una tarea específica.

Un programa se construye, se escribe en un lenguaje de programación determinado por el programador, siendo esta la persona diseñadora y escritora del programa (Joyanes y Zahonero; 2003; p.19). Y son los lenguajes de programación secuencias de caracteres, o sea de letras, números, símbolos retornos de carro y espacios.

Se puede definir un lenguaje de programación como un lenguaje que reconoce la computadora, a través de una serie de instrucciones que permitan resolver un problema, las cuales tienen un significado concreto y "entendible". Estas instrucciones son: las palabras reservadas (keywords), los identificadores y los símbolos, la sintaxis del lenguaje definirá cómo se combinarán todos estos para producir un código ejecutable por el ordenador.

Se le conoce como identificadores a las variables, constantes, tipos de datos, nombres de funciones, nombres de archivos incluidos y parámetros en definición de funciones.

Las palabras reservadas son las que provee el lenguaje de programación para realizar ciertas acciones, por lo tanto estas no pueden ser usadas como nombres de identificadores (variables, funciones, constantes, etc.). Estas palabras pueden ser diversas, dependerá del lenguaje de programación que se use, en este caso utilizamos C y algunas de las palabras reservadas en este son: main, break, while, for, swich, continue, printf, scanf, define, etc.

Una variable es una caja donde se puede almacenar un dato, pero de modo que se pueda ir cambiando su contenido (Castro, et al; 1993; p. 4). Cada variable debe ser declarada como de alguno de los tipos primitivos, tipos estructurados incorporados al lenguaje o también de tipos definidos por el usuario como variables estructuradas.

Ejemplos de definición de variables en base a tipos primitivos son los siguientes:

```
int x, y;
int edad, contador=0;
float acumulador=0;
int metros_ancho, metros_largo;
char resp, opcion='S';
```

Constantes: son valores que no cambian durante la ejecución del programa (Joyanes; 1994; p. 18). Ejemplos de definición de constantes son los siguientes:

```
const int edad = 18;
const float PI = 3.14159
```

const char *CAPITAL = "Tepic";

Las constantes pueden ser declaradas globalmente, es decir fuera de las funciones que compongan el programa; o localmente, es decir dentro del ámbito de una función en particular.

Tipos de Datos Primitivos en C++

Tabla 1.1 Constantes

Tipo	Rango de valores
Unsigned char	0 a 255
Char	-128 a 127
Enum	-32,768 a 32,767
Unsigned int	0 a 65,535
Short int	-32,768 a 32,767
Int	-32,768 a 32,767
Unsigned long	0 a 4,294,967,295
Long	-2,147,483,648 a 2,147,483,647
Float	3.4 * (10**-38) a 3.4 * (10**+38)
Double	1.7 * (10**-308) a 1.7 * (10**+308)
Long double	3.4 * (10**-4932) a 1.1 * (10**+4932)

La **asignación** es una operación que coloca un valor en una dirección de la memoria RAM. El lenguaje C/C++ utiliza como símbolo de asignación el igual **(=).**
El valor de la expresión a la derecha debe ser de un tipo de datos compatible con la única variable a la izquierda, en caso contrario se producirán resultados inesperados e inclusive el bloqueo de la máquina.

```
y = y + 5;
salario=4000.50;
Edo_civil='S';
```

En el ejemplo anterior a la variable **y** se le suma el número 5, este nuevo valor obtenido es asignado a la misma variable y (en la memoria RAM), borrando el inicial

Los operadores aritméticos

Los operadores aritméticos sirven para realizar operaciones aritméticas básicas. Los operadores aritméticos en "C" siguen las reglas algebraicas típicas de la jerarquía o prioridad en matemáticas. Estas reglas especifican el orden en que se van a llevar a cabo las operaciones aritméticas, en donde las expresiones interiores a paréntesis se evalúan primero, a continuación, se realizan los operadores unitarios – los cuales se abordaran más ampliamente en otras prácticas, seguido por los operadores de multiplicación, división, suma y resta.

Tal como era de esperarse los operadores aritméticos, mostrados después de este párrafo, comprenden las cuatro operaciones básicas, suma, resta, multiplicación y división, con un agregado, el operador módulo.

Tabla 1.2 Operadores aritméticos

Símbolo	Descripción	Ejemplo
+	Suma	a + b
-	Resta	a - b
*	Multiplicación	a * b
/	División	a / b
%	Modulo	a % b

El operador módulo (%) se utiliza para calcular el resto del cociente entre dos enteros, y no puede ser aplicado a variables del tipo float ó double .
Por ejemplo

```
a += b;      /* equivale :   a = a + b     */
a -= b;      /* equivale :   a = a – b     */
a *= b;      /* equivale :   a = a * b     */
a /= b;      /* equivale :   a = a / b     */
/a %= b;     /* equivale :   a = a % b     */
```

Prioridad de los operadores aritméticos

Según Joyanes (1994; p. 22), las expresiones que tienen dos o más operandos requieren unas reglas matemáticas que permitan determinar el orden de las operaciones, se denominan reglas de prioridad o precedencia y son:

A. Las operaciones que están encerradas entre paréntesis se evalúan primero. Si existen diferentes paréntesis anidados (interiores unos a otros), las expresiones más internas se evalúan primero.
B. Las operaciones aritméticas dentro de una expresión suelen seguir el siguiente orden de prioridad.

1. Operador exponencial (^ o bien **).
2. Operador *, /.
3. Operador +, -.
4. Operadores div y mod.

En caso de coincidir varios operadores de igual prioridad en una expresión o subexpresión encerrada entre paréntesis, el orden de prioridad en este caso es de izquierda a derecha.

Funciones – Instrucciones de salida

Uso del printf ()

Formato

```
printf("Texto y/o formato de salida",variable o constante);
printf("Texto y/o formato de salida",variable o constante);
printf("Texto y/o formato de salida",operación aritmética);
printf("Texto");
printf("Texto y/o formato de salida",operación lógica)
```

Ejemplos

```
printf(“Estado civil: “);
printf(“Salario:  %lf”, sal);
printf(“%d”,a>b);
printf(“%f”,123*23);
printf(“Francisco tiene %d años y Pedro %d”,ed1,ed2);
```

Uso del puts ()

Formato

```
puts(“texto”);
puts(variable cadena);
puts(concatenación);
```

Ejemplos

```
puts(“Hola“);
puts(nom);
puts(“Nombre: “+nom);// donde nom es variable tipo caracter
```

Funciones – Instrucciones de entrada

La entrada de datos a un programa puede tener diversas fuentes, pero la entrada que se considera ahora es a través del teclado, asociado al archivo estándar de entrada stdio. La función más utilizada, por su versatilidad, para entrada formateada es scanf() mientras que para salida es printf(). En donde los códigos de formato más comunes se les puede agregar una l o L, siendo el significado de la l largo, aplicada a float %lf indicando tipo double y aplicado a int - %ld indica entero largo.

Uso del scanf()

Formato

```
scanf(“formato”,&variable);                 // para leer una sola variable
scanf(“formatos”,&var1,&var2…,&varn);  // para leer varias variables
```

Ejemplos

```
scanf(“%d”,&ed);
scanf(“%d  %d  %d”,&a,&b,&c);
scanf(“%f”,&sal);
scanf(“%s”,&nom);
```

Uso del gets()

Formato

```
gets(variable cadena);
```

Ejemplo

```
gets(nom);
```

Códigos y formatos más utilizados y su significado, según presentan Joyanes y Zahonero (2003; p.108):

%d El dato se convierte en entero decimal.
%o El dato entero se convierte en octal.
%x El dato entero se convierte en hexadecimal.
%u El dato entero se convierte en entero sin signo.
%c El dato se considera de tipo carácter.
%f El dato se considera de tipo float. Se convierte a notación decimal, con parte entera y los dígitos de precisión.
%g El datos se considera de tipo float. Se convierte según en código %e o %f dependiendo de cuál sea la presentación más corta.
%s El dato ha de ser una cadena de caracteres.
%lf El dato se considera de tipo double

Estructura general de un programa en C

Los elementos que constituyen un programa en C son principalmente funciones. Un programa contiene una serie de directivas #include que permiten incluir en el mismo archivo de cabecera que a su vez constará de funciones y datos predefinidos en ellos.

A continuación se presenta la estructura de un programa:

```
#include < >              //Directivas del prepocesador – dentro de <> va un nombre
                          //de archivo con extensión h – está sección es obligatoria
#define          //Macros de procesador - no siempre se utilizan
/*-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o— Declaraciones globales – no siempre se utilizan prototipos de funciones variables */
//-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o- Función principal main – sección obligatoria

main()
{
        declaraciones locales
        sentencias
}
//-o-o-o-o-o-o-o-o-o-o Definiciones de otras funciones – no siempre se utilizan
Tipo1 func1(…)
{
        …….
}
```

Ejemplo

```
#include <stdio.h>
#include <conio.h>

main()                    // si indica void main() entonces no indica la instrucción return 0
{                         // las dos en una línea indican un comentario solamente
        clrscr();         // función que limpia pantalla
        printf ("Hola – estamos probando un programa");
        return 0;
```

}

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

Resuelva al menos uno de los problemas del uno al cuatro y el número cinco:

1. La presión, el volumen y la temperatura de una masa de aire se relacionan por la fórmula: masa = (presión * volumen)/(0.37 * (temperatura + 460))
2. Don Juan vendió a 3 personas varias hectáreas de tierra a cada una. Cada hectárea cuesta 30000. Don Juan quiere saber el número de hectáreas que vendió en total, lo que le pagará cada uno de ellos (considerando un descuento del 7% por cada hectárea) y lo que le pagarán en conjunto considerando también el descuento.
3. Calcular el número de pulsaciones que una persona debe tener por cada 10 segundos de ejercicio, si la fórmula es: num. pulsaciones = (220 - edad)/10
4. Calcular el nuevo salario de un obrero si obtuvo un incremento del 25% sobre su salario anterior.
5. Cinco personas quieren conocer el total que pagaran por viajar a la ciudad de Celaya, dos de ellas viajarán en camión de primera y las otras tres en uno de segunda, el camión de segunda cobra el 30% menos que el de primera.

- Obtener el área y el perímetro de las siguientes figuras: cuadrado, rectángulo y un triángulo.
- Una tienda ofrece el descuento del 15% sobre el total de la compra y un cliente desea saber cuánto deberá pagar finalmente por su compra.
- Convertir una cantidad de grados Celsius a grado farenheith.

Práctica 2 Estructura de control selectiva si-entonces (if), operadores relaciónales y lógicos

Objetivo específico de la práctica

Con esta práctica se pretende que el estudiante declare, use y evalué correctamente las operaciones lógicas en la estructura de control selectiva if-else, que conozca y aplique correctamente la estructura de control antes señalada.

Criterios de desempeño

El estudiante estará capacitado para identificar, declarar y utilizar los operadores relacionales y lógicos. Además de conocer y utilizar la estructura de control selectiva simple if-else (si entonces – si no) cuando:

- Identifique los operadores relacionales.
- Identifique los operadores lógicos.
- Memorice los operadores relacionales y lógicos.
- Comprenda el uso de los operadores relacionales y lógicos, en una operación lógica.
- Aplique las reglas de evaluación de las operaciones lógicas.
- Diferencie los operadores relacionales de los lógicos.
- Diferencie una operación aritmética de una lógica.
- Construya correctamente una operación lógica.
- Evalué correctamente una operación lógica.
- Relacione correctamente las operaciones lógicas con el uso del si entonces – sino (if-else).
- Relacione correctamente las operaciones lógicas con el uso del si entonces (if).
- Resuelva problemas mediante un programa informático en que utilice correctamente las operaciones lógicas con la estructura de control selectiva simple si entonces – sino (if-else). Además de usar correctamente las instrucciones de las practicas anteriores.

Resultado esperado en relación a los criterios de desempeño específicos de la práctica

- La conceptualización por parte del estudiante de los términos operación lógica, estructura de control selectiva si entonces - sino (If-else).
- Dominio de la sintaxis – formato para usar la instrucción selectiva de control simple si entonces (if) y si entonces- sino (if-else).
- La ejemplificación de uso de la estructura if-else.
- La capacidad de elegir el operador o los operadores relacionales y lógicos más adecuados que intervengan en la operación lógica requerida.
- Demostración de la habilidad del estudiante para identificar, declarar y usar operadores relacionales y lógicos válidos, mediante su aplicación en un programa informático.
- La modificación – corrección de errores de sintaxis relacionados con los operadores relacionales y lógicos en un programa informático, específicamente en la estructura de control selectiva si entonces-sino (if-else).
- Aplicación de la habilidad del estudiante para utilizar la estructura de control selectiva si entonces-sino (if-else) para dar solución a un problema mediante un programa informático.

Introducción

Un programa, basándose en los datos que reciba, puede elegir entre distintas acciones a seguir. Sin embargo si se quiere que tome decisiones, se le debe indicar específicamente qué información debe utilizar, cómo evaluarla y que acciones seguir. Este tipo de acción se denomina bifurcación, y se ejecuta basándose en la evaluación lógica de los datos.

La forma más sencilla de bifurcación condicional es la sentencia Si-Entonces, la cual hace que un programa ejecute el bloque de código el cual está a continuación, si es cierta la condición, en caso contrario no lo ejecutará. En el lenguaje C la instrucción Entonces (Then) está tácita y no se debe colocar.

Lo primero que hace una instrucción Si (if) es evaluar la información que se le ha proporcionado en forma de expresión booleana. Esta evaluación produce uno de dos resultados posibles: cierto (True) o falso (False).Si la evaluación es cierta el algoritmo seguirá en el bloque de instrucciones que está inmediatamente después, si el resultado es falso, la secuencia del algoritmo se salta este bloque de instrucciones.

En el C/C++ el conjunto de instrucciones que siguen a la condición lógica deben ir entre llaves { } para abrir y cerrar respectivamente. Si el código a efectuar después de la condición consta de una sola instrucción, no es necesario utilizar los delimitadores: { }.
En el lenguaje C, Falso queda representado por un valor entero nulo (cero) y Verdadero por cualquier número distinto de cero. En donde todas las expresiones lógicas tienen que utilizar operadores relacionales y dar como resultado un valor falso o verdadero.

Operadores relacionales

Tabla 2.1 Operadores relacionales

Símbolo	Descripción	Ejemplo
<	menor que	(a < b)
>	mayor que	(a >b)
< =	menor o igual que	(a < = b)
>=	mayor o igual que	(a >= b)
= =	igual que	(a == b)
! =	distinto que	(a != b)

Un error típico es confundir el operador de asignación (=) con el operador de igualdad (==).

Operadores Lógicos

Hay tres operadores que realizan las conectividades lógicas Y (AND), O (OR) y NEGACIÓN (NOT) y están descritos a continuación

Tabla 2.2 Operadores Lógicos

Símbolo	Descripción	Ejemplo
&&	Y (AND)	(a>b) && (c < d)
||	O (OR)	(a>b) || (c < d)
!	NEGACION (NOT)	!(a>b)

Los operadores **"&&", "||"** y **"!"** relacionan expresiones lógicas, formando a su vez nuevas expresiones lógicas. Sintaxis:

<expresión1> && <expresión2><expresión1> || <expresión2>!<expresión>

El operador **"&&"** equivale al "**AND**" o "**Y**"; devuelve "true" sólo si las dos expresiones evaluadas son "true" o distintas de cero, en caso contrario devuelve "false" o cero. Si la primera expresión evaluada es "false", la segunda no se evalúa.

A continuación se muestra la tabla de verdad del operador &&:

Expresión1	Expresión2	Expresión1 && Expresión2
false	ignorada	false
true	false	false
true	true	true

El operador **"||"** equivale al "**OR**" u "**O** inclusivo"; devuelve "true" si cualquiera de las expresiones evaluadas es "true" o distinta de cero, en caso contrario devuelve "false" o cero. Si la primera expresión evaluada es "true", la segunda no se evalúa.

Expresión1	Expresión2	Expresión1 || Expresión2
false	false	false
false	true	true
true	ignorada	true

El operador **"!"** es equivalente al "**NOT**", o "**NO**", y devuelve "true" sólo si la expresión evaluada es "false" o cero, en caso contrario devuelve "false".
La expresión "!E" es equivalente a (0 == E).

Expresión	!Expresión
false	true
true	false

La prioridad de los operadores lógicos es la siguiente:

1. NOT
2. AND
3. OR

La prioridad de los operadores en general es la siguiente:

1. ( ) se evalúan primero los paréntesis.
2. Exponente y raíz.
3. Multiplicación, división, mod, not.
4. Suma, resta, and.
5. >, <, >=, <=, <>, =, or.

Operadores de Incremento y Decremento

Tabla 2.3 Operadores de Incremento y Decremento

Símbolo	**Descripción**	**Ejemplo**
++	incremento	++i ó i++
--	decremento	--i ó i--

Estructura de control selectiva simple Instrucción Si - Entonces

Sintaxis de la instrucción Si (If) en la forma más simple:

```
...
if (expresión lógica)
{
     ... ;
     instrucción(es);
     ... ;
}
... ;
... ;
... ;
if (expresión lógica)
     Instrucción_única;
... ;
... ;
... ;
```

Ejemplo con if:

```
#include <stdio.h>
#include <conio.h>
void main()
{
  int salario;
  printf("\n Salario que gana: ");
  scanf("%d",&salario);
  if (salario>4000)
     printf("Buen salario: ");
  getch();                              //función que lee un carácter
}
```

Ejemplo con if-else:

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int salario;
 printf("\n Salario que gana: ");
 scanf("%d",&salario);
 if (salario>4000)
    printf("Buen salario: ");
 else
    printf("Mal salario");
 getch();                                    //función que lee un carácter
}
```

Ejemplo con if anidados:

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int salario,ed;
 printf("\n Edad: ");                        scanf("%d",&ed);
 printf("\n Salario que gana: ");            scanf("%d",&salario);
 if (salario>4000)
 {
    if (ed>18)
        printf("Buen salario y edad adecuada ");
    else
        printf("Buen salario pero edad inadecuada");
 }
 else
    printf("Mal salario");
 getch();
}
```

Ejemplo con operadores lógicos:

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int salario,ed;
 printf("\n Edad: ");                        scanf("%d",&ed);
 printf("\n Salario que gana: ");            scanf("%d",&salario);
 if (salario>4000 && ed>18)
 {
        printf("Buen salario y edad adecuada ");
    }
 else
    printf("Mal salario o edad inadecuada");
 getch();
}
```

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

Resuelva los problemas seis, ocho y al menos uno de los últimos problemas

6. Problema: Leer la edad de una persona, determinar y señalar si la persona es mayor o menor de edad, utilizando la estructura de control selectiva if-else para solucionar el problema.
7. Problema: Leer tres números diferentes y determinar cuál es el mayor de ellos, utilizando la estructura de control selectiva if-else para solucionar el problema y anidamientos.
8. Una persona debe 1000 de luz, 3000 de tel, 1500 de renta, 600 de gas, 3000 de colegiatura. Ella quiere saber si alcanzará a pagar con el salario de su marido y si es así, cuánto le quedará para otros gastos.
9. Leer tres enteros y mostrar el mediano de ellos (no el mayor ni el menor, sino el medio). Utilice operadores lógicos – relacionales y también con anidamientos.

- Determinar si un alumno aprueba o reprueba el curso, sabiendo que aprobará si su promedio de tres calificaciones es mayor o igual a 60; reprueba en caso contrario.
- Leer dos números, si son iguales que los multiplique, si el primero es mayor que el segundo que los reste y sino que los sume.
- Determinar si un número es positivo, negativo o neutro.

Práctica 3 Estructura de Selección Múltiple Casos- switch

Objetivo específico de la práctica

Que el estudiante conozca, identifique y aplique correctamente la estructura de control de selección múltiple switch.

Criterios de desempeño

El estudiante estará capacitado para identificar, declarar y utilizar la estructura de control selectiva múltiple switch (según sea el caso) cuando:

- Identifique y utilice los tipos de datos que se pueden evaluar en el switch().
- Memorice el formato, los operadores relacionales y lógicos.
- Conozca la sintaxis del switch.
- Comprenda el funcionamiento del switch.
- Resuelva problemas mediante un programa informático en que utilice correctamente la estructura de control de selección múltiple. Además de usar correctamente las instrucciones de las practicas anteriores.

Resultado esperado en relación a los criterios de desempeño específicos de la práctica

- La conceptualización por parte del estudiante de los términos estructura de control de selección múltiple switch (según sea el caso).
- Dominio de la sintaxis – formato para usar la estructura de control de selección múltiple switch (según sea el caso).
- La ejemplificación de uso de la estructura de control de selección múltiple switch (según sea el caso).
- La modificación – corrección de errores de sintaxis relacionados con la estructura de control de selección múltiple switch (según sea el caso).
- Aplicación de la habilidad del estudiante para utilizar la estructura de control de selección múltiple switch (según sea el caso) para dar solución a un problema mediante un programa informático.

Introducción

El **switch** es una instrucción que utiliza C - C++ de selección múltiple, se usa cuando las alternativas o casos pueden ser varios, se puede elegir de entre varios casos el que sea igual al valor comparado. Por lo regular se utilizan tipos de datos simples como el entero o de carácter. Tiene como ventaja que no se limita a una sola opción, sino que puede hacer una selección de entre múltiples casos, optimizando código al no tener que hacer tanto anidamiento si se utiliza el if - else.

El formato cómo se trabaja es el siguiente:

```
... ;
switch (variable entera o carácter)
{
   case opc1: instrucción(es);
          break;
   case opc2: instrucción(es);
          break;
          ...
   case opcN: instrucción(es);
          break;
   default:
         instrucción(es)
 }
 ... ;
 ... ;
```

Un ejemplo de su uso es:

Dado el número de un día de la semana, hacer aparecer el nombre del día correspondiente:

```
#include <stdio.h>
#include <conio.h>
void main()
{
   int Dia;
   clrscr();
   printf("\n Digite el número de un día de la semana(1 - 7):  ");
   scanf("%d",&Dia);

   switch (Dia)
   {
     case 1: printf("\n Es día es LUNES");          break;
     case 2: printf("\n El día es MARTES");        break;
     case 3: printf( "\n El día es MIERCOLES");    break;
     case 4: printf( "El día es JUEVES");                  break;
     case 5: printf( "El día es VIERNES)";         break;
     case 6: printf( "El día es SABADO");          break;
     case 7: printf( "El día es DOMINGO");         break;
     default: printf("\aERROR, no es un entero válido");
   }
   getch();
}
```

Otro ejemplo:

```
#include <stdio.h>
#include <conio.h>
void main()
{
char c;
printf("\n MENU :") ;
printf("\n       A = Altas ") ;
printf("\n       B = Bajas   ") ;
printf("\n       C = Consultas    ") ;
printf("\n\n  Presione la letra correcta: ") ;
c = getche() ;
switch (c)
 {
 case 'A' :
          printf("\n Agregar cliente ") ;
          break ;
 case 'B' :
          printf("\nEliminar cliente") ;
          break ;
 case 'C' :
          printf("\n Imprimir registro de cliente") ;
          break ;
 case '\n':                  //presiono la tecla enter-intro
          printf("\n¡No ha seleccionada nada!" ) ;
          break ;
 default  :
          printf("\n caracter incorrecto" ) ;
          break ;
 }
}
```

Completa los siguientes enunciados:

1. Es una instrucción que utiliza C - C++ de selección múltiple, puede elegir de entre varios casos el que sea igual al valor comparado: _____

2. Esta estructura de control simple se utiliza para elegir entre dos opciones solamente: _____

3. Si la evaluación es cierta el algoritmo seguirá en el bloque de instrucciones que está inmediatamente después, si el resultado es falso, la secuencia del algoritmo se salta este bloque de instrucciones: _____

4. Utiliza a menudo tipos simples de datos como de carácter o enteros, puede elegir entre varios casos y no limitarse a una sola opción, proporcionando una forma clara de bifurcación múltiple:_____

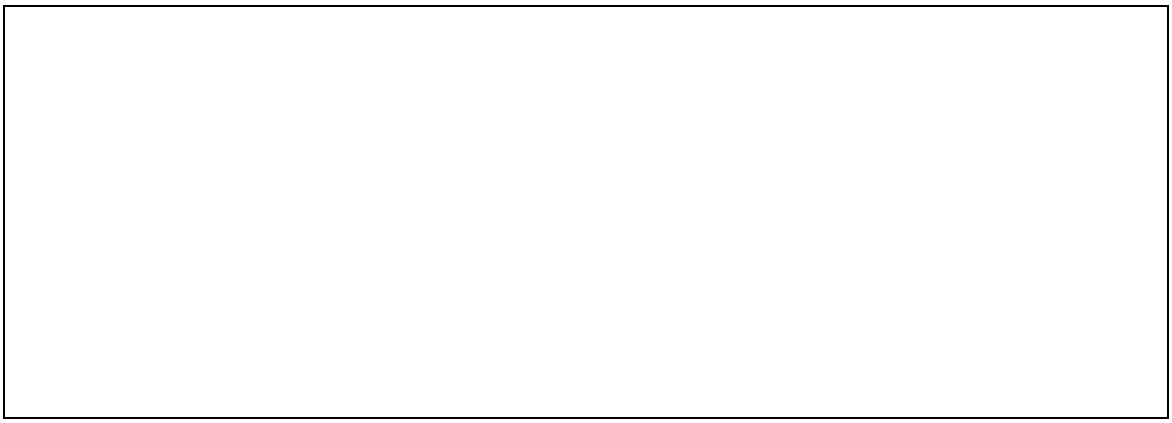
5. Es una instrucción que ha de evaluar la información que se le ha proporcionado en forma de expresión booleana. Esta evaluación produce uno de dos resultados posibles: cierto (True) o falso (False): _____

Encuentra y señale los errores del siguiente programa:

```
include <stdio.h>
include <conio.h>
void main()
{
integer opc=0;
printf("\n MENU :") ;
printf("\n       200 = Altas ") ;
printf("\n       100 = Bajas   ") ;
printf("\n       300 = Consultas    ") ;
printf("\n\n  Introduce la opción elegida: ") ;
c = getche() ;
switch (c)
 {
  case '100' :
          printf("\n Agregar cliente ") ;
          break ;
  case '200' :
          printf("\nEliminar cliente") ;

  case '300' :
          printf("\n Imprimir registro de cliente") ;
          break ;
  default  :
          printf("\n caracter incorrecto" ) ;
          break ;
 }
}
```

Escriba el pseudocódigo que permita leer una vocal y la muestre en pantalla, señalando cuando lea un carácter diferente a una vocal.



-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

c) Resuelva los problemas:

10. Introduzca un número del 1 al 12 y muestre el mes del año al que corresponde, utilizando la estructura de control selectiva switch() para solucionar el problema.

11. En este programa utiliza anidamientos en los casos. Una persona quiere saber el salario que gana mensualmente, el pago base por quincena es de $ 3000.00 el cual cambiará dependiendo de la localidad en que se trabaja, las localidades pueden ser: Ahuacatlán, Ixtlán, Jala, Amatlán de Cañas y Chapalilla; en caso de no ser ninguna de las localidades planteadas se pintará "salario indefinido" mostrando solo el salario base. Si la localidad es Ixtlán se aumentará un 20% en cada quincena, para Jala un 5%, Ahuacatlán un 10%, Chapalilla un 15% y Amatlán de Cañas se le aumentan $ 200.00.

Práctica 4 Estructura de Control Cíclica Mientras (while)

Objetivo específico de la práctica

Que el estudiante conozca, identifique y aplique correctamente la estructura de control cíclica Mientras (while).

Criterios de desempeño

El estudiante estará capacitado para identificar, declarar y utilizar la estructura de control cíclica Mientras (while) cuando:

- Sepa utilizar contadores y acumuladores.
- Memorice el formato para declarar un ciclo con while (mientras).
- Diferencie la estructura cíclica while de las demás (for y do while).
- Conozca la sintaxis del while.
- Comprenda el funcionamiento del while (mientras).
- Resuelva problemas mediante un programa informático en que utilice correctamente la estructura de control cíclica while (mientras). Además de usar correctamente las instrucciones de las practicas anteriores.

Resultado esperado en relación a los criterios de desempeño específicos de la práctica

- La conceptualización por parte del estudiante de los términos de la estructura de control de control cíclica while (mientras), contador y acumulador.
- Dominio de la sintaxis – formato para usar la estructura de control cíclica while (mientras).
- La ejemplificación de uso de la estructura de control cíclica while (mientras).
- La modificación – corrección de errores de sintaxis relacionados con la estructura de control cíclica while (mientras).
- Aplicación de la habilidad del estudiante para declarar y manipular contadores.
- Aplicación de la habilidad del estudiante para declarar y manipular acumuladores.
- Aplicación de la habilidad del estudiante para utilizar la estructura de control cíclica while (mientras) para dar solución a un problema mediante un programa informático.

Introducción

En diversas ocasiones nos encontramos que para dar solución a un problema escribimos un número de líneas de código exageradamente extenso, es en estas situaciones cuando se hace necesario el uso de los llamados **ciclos o bucles**, los cuales permiten ejecutar repetidamente un bloque de instrucciones, optimizando así el código y ahorrándonos el tener un programa demasiado extenso. La instrucción Mientras (while) soluciona este problema, permitiendo ejecutar reiteradamente iteraciones (ciclos, bucles, lazos, rizos).

De esta forma será posible "devolvernos" en la secuencia lógica del algoritmo.

While - mientras

El **while** es una de las tres iteraciones posibles en C (las otras serían el for y el do while). Su sintaxis podría expresarse de la siguiente forma:

```
....
while (Exp. booleana)
{
   ....
   Instrucción(es);
   ....
}
....
```

El conjunto de instrucciones que contengan las llaves del while, se ejecutarán mientras sea verdadera la expresión booleana que controle el ciclo (while)
Cuando la instrucción mientras (while) evalúe la expresión booleana que controla el ciclo como falsa, se ejecutarán en las instrucciones que se encuentran después de la llave que cierra el while en forma secuencial.

Ejemplo:

```
While (condición)
{                    //llave de inicio del while
  printf (“hola”);                  // Instrucción dentro del while
  printf  (“\n”);            // instrucción dentro del while
}          //llave de fin del while
printf(“ Termina”);                //instrucción fuera de while, se ejecuta cuando la
condición printf (“ el while “);            //sea falsa
```

La instrucción while debe ir seguida de una expresión lógica encerrada entre paréntesis, que al ser evaluada, si es verdadera se irá ejecutando el código hasta encontrar el terminador }, en este momento el flujo del algoritmo se "devuelve" trasladándose "arriba" a la instrucción while, nuevamente; aquí la expresión es evaluada de nuevo, si es verdadera entra la secuencia otra vez al ámbito de la instrucción, y así en repetidas ocasiones hasta que la condición que controla el ciclo se vuelve falsa, en este momento el control del programa se traslada después de la llave de cerrar } y continua ejecutando las instrucciones que se encuentran después de la llave } que cierra.

Cuando la instrucción de control (condición) no toma nunca el valor de falso se producirá un ciclo infinito, y habría que interrumpir el programa con la secuencia de teclas ctrl+break, o con alguna otra instrucción que lo obligará a salir, lo cual no es lo más recomendado según los teóricos de la programación estructurada. Si al evaluarse la primera vez la condición de control, esta llegara a ser falsa, no se ejecutarían las instrucciones contenidas en el ámbito del while, sino que seguiría secuencialmente el programa.

A continuación se presenta un algoritmo para sumar los 10 primeros números:

```
#include <stdio.h>
#include <conio.h>
void main ()
{
    int numero = 0, suma = 0;        // numero hace la función de contador y suma de acumulador
    while (numero < 10)              // condición – expresión lógica
    {
       numero = numero + 1;          // incremento del contador numero el cual controla el ciclo
        suma = suma + numero;        //acumulación de numero
    }
    printf("Sumatoria del 1 al 10 =  %d", suma);
}
```

En forma tabular las variables del algoritmo anterior tomarán los siguientes valores:

Contador Número	**Acumulador suma**		**número<10**
0	0		0<10 Verdad
1	1		1<10 Verdad
2	3		2<10 Verdad
3	6		3<10 Verdad
4	10		4<10 Verdad
5	15		5<10 Verdad
6	21		6<10 Verdad
7	28		7<10 Verdad
8	36		8<10 Verdad
9	45		9<10 Verdad
10	55	1	0<10 Falso

En pantalla, al ejecutarlo se verá así:

Sumatoria del a al 10 = 55

Completa los siguientes enunciados:

1. Cuando la instrucción _____ evalúe la expresión booleana que controla el ciclo como falsa, la ejecución continuará secuencialmente después de la instrucción que marca el fin del ámbito de la instrucción, o sea la llave de cerrar }. La expresión lógica que controla el ciclo _____es evaluada en la parte superior.

2. ¿El resultado es falso o verdadero?, desglose resultados, a=70; b=90; c=100; d=1.5;
   printf("\n\n %d",(((c==b) && (b>a) && (d>-4)) || ((a!=c) && (c<b)) && (d>0)));

3. Cuando la instrucción evalúe la expresión booleana que controla el ciclo como falsa, la ejecución lógica del algoritmo continuará secuencialmente después de la instrucción que marca el fin del ámbito de la instrucción _____

5. Se utiliza para especificar un bucle condicional que se ejecuta al menos una vez. Este se suele dar en algunas circunstancias en que precisamente se requiere su ejecución de al menos una vez con posibilidades de más veces: _____

6. El ciclo que produce la instrucción se efectuará, ejecutando las instrucciones que contiene – dentro de las llaves, mientras sea verdadera la expresión booleana que lo está controlando. Este ciclo no entra ni una sola vez si la condición no se cumple y su condición va entre paréntesis después de la instrucción._____

Indique la salida en pantalla del siguiente código:

```
int a=0, con=0;acu=10;
while (con<10 && acu<100)
{
        con++;
        acu =acu+20;
        con = con +1;
        printf("\n acumulador = %d  y contador = %d");
}
```

-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-

Resuelva los problemas:

12. Leer siete calificaciones, promediar y pintar el resultado

13. Leer siete calificaciones, promediar y pintar el resultado. Si el promedio es superior o igual a 60 pintar acredita sino pinta no acredita.

14. Tres personas van de vacaciones a la playa de Nuevo Vallarta, se hospedarán en el hotel de cinco estrellas "Monarcas", en donde el costo por habitación varía según el número de días que se hospede el cliente y la edad de este.

- Personas mayores de 60 años que se hospeden más de 5 días se tiene un costo del precio base menos el 40%.
- Personas mayores de edad y menores de 61 años pagaran la tarifa base por día. Sólo se les descuenta el 20% si se hospedan más de 9 días.
- Para personas menores de edad se hace un descuento del 50% sobre el precio base, independientemente de los días que se hospeden.
- Se requiere conocer la cantidad que pagar cada una de las personas, así como el descuento que recibe. También se requiere saber el total que pagaron en conjunto (incluido el descuento).

15. Se quiere formar un equipo de desarrolladores de software, conformado por seis personas, el requisito indispensable para formar dicho equipo es una calificación igual o mayor a 90. Si la persona tiene más de 89 de calificación entonces se le preguntará la edad y su domicilio, en caso de no cumplir con este requisito se le indicar que no es aceptado.

Práctica 5 Estructura de Control Cíclica Desde o Para (for)

Objetivo específico de la práctica

Que el estudiante conozca, identifique y aplique correctamente la estructura de control cíclica Desde o Hasta (for).

Criterios de desempeño

El estudiante estará capacitado para identificar, declarar y utilizar la estructura de control cíclica Desde o Hasta (for) cuando:

- Sepa utilizar contadores y acumuladores. Quizás mejor.
- Memorice el formato para declarar un ciclo con Desde o Hasta (for).
- Diferencie la estructura cíclica for de las demás (while y do while).
- Conozca la sintaxis del for.
- Comprenda el funcionamiento del Desde o Hasta (for).
- Resuelva problemas mediante un programa informático en que utilice correctamente la estructura de control cíclica Desde o Hasta (for). Además de usar correctamente las instrucciones de las practicas anteriores.

Resultado esperado en relación a los criterios de desempeño específicos de la práctica

- La conceptualización por parte del estudiante de los términos de la estructura de control cíclica Desde o Hasta (for), contador y acumulador.
- Dominio de la sintaxis – formato para usar la estructura de control cíclica Desde o Hasta (for).
- La ejemplificación de uso de la estructura de control cíclica Desde o Hasta (for).
- La modificación – corrección de errores de sintaxis relacionados con la estructura de control cíclica Desde o Hasta (for).
- Aplicación de la habilidad del estudiante para declarar y manipular tanto acumuladores como contadores.
- Aplicación de la habilidad del estudiante para utilizar la estructura de control cíclica Desde o Hasta (for) para dar solución a un problema mediante un programa informático.

Introducción

La mayoría de programas requieren que una determinada cantidad de instrucciones se ejecuten más de una vez. Hasta el momento la única forma de hacerlo sería o copiándolas varias veces, las instrucciones, o copiarlas dentro de una función e invocarla repetidamente. La instrucción Desde o Para (for) soluciona este problema, permitiendo ejecutar reiteradamente iteraciones (ciclos, bucles).

For – desde o para

El bucle for es el más adecuado más usado en bucles controlados por un contador, en los que un conjunto de sentencias se ejecutan una vez por cada valor de un rango especificado, aunque este último comentario no significa que no se pueda incrementer de dos en dos, hacer otra incrementación o decrementación.

El for es una de las tres iteraciones posibles en C. Su sintaxis podría expresarse de la siguiente forma:

```
for (conta=ValorIni; conta<=ValorFin; conta=conta+incremento)
{
   ...;
   Instrucción(es);
   ...;
}
```

El ciclo que produce la instrucción desde o hasta (for) se efectuará, ejecutando las instrucciones que contiene, mientras sea verdadera la expresión booleana que lo está controlando.Cuando la instrucción desde o hasta (for) evalúe la expresión booleana que controla el ciclo como falsa, la ejecución continuará secuencialmente después de la instrucción que marca el fin del ámbito de la instrucción, o sea la llave de cerrar }. La expresión lógica que controla el ciclo for es evaluada en la parte superior.

Cuidado: Si la instrucción de control no toma nunca el valor de falso se producirá un ciclo infinito, y habría que interrumpir el programa con la secuencia de teclas Ctrl+Break, o con alguna otra instrucción que lo obligará a salir, lo cual no es lo más recomendado según los teóricos de la programación.

A continuación, se presenta un ejemplo calcular el promedio de las calificaciones de 6 materias

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int con, acum=0, calif;
 float promedio;
 clrscr();
 for (con=1; con<=6; con++)     // uso frecuente del for controlado por con
 {
   printf("\n\n Introduce calificación: ");
   scanf("%d",&calif);
   acum=acum+calif;
 }
 promedio=acum/6;
 printf("\n\n El Promedio General:  %.2f",promedio);
 getch();
 }
```

Completa los siguientes enunciados:

1. La forma más sencilla de bifurcación condicional es la sentencia ________________________, la cual hace que un programa ejecute el bloque de código el cual está a continuación, si es cierta la condición, en caso contrario no lo ejecutará.

2. En el C/C++ el conjunto de instrucciones que siguen a la condición lógica deben ir entre _______________________ para abrir y cerrar respectivamente. Si el código a efectuar después de la condición consta de una sola instrucción, no es necesario utilizar los delimitadores: ____

3. Los __________________________ permite ejecutar reiteradamente iteraciones un conjunto de instrucciones varias veces.

4. Esta estructura de control cíclica tiene la condición al final del ciclo y no al principio: ____

Localice y escriba los errores en el siguiente código:

```
#include <stdio.h>
#include <conio.h>
void main()
{
 int con, acum=10, calif=10;
 float promedio;
 clrscr();
 for (con=1; con<=6; con++)
 {
   printf("\n\n Introduce calificación: ");
   scanf("%d",&calif);
   acum=acum+con;
 }
 promedio=acum/6;
 printf("\n\n El Promedio General:  %.2d",promedio);
 getch();
}
```

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

Resuelva los problemas:

16. Leer siete calificaciones, promediar y pintar el resultado.

17. Leer siete calificaciones, promediar y pintar el resultado. Si el promedio es superior o igual a 60 pintar acredita sino pinta no acredita

18. Tres personas van de vacaciones a la playa de Nuevo Vallarta, se hospedarán en el hotel de cinco estrellas "Monarcas", en donde el costo por habitación varía según el número de días que se hospede el cliente y la edad de este.

- Para personas mayores de 60 años que se hospeden más de 5 días se tiene un costo del precio base menos el 40%.
- Personas mayores de edad y menores de 61 años pagaran la tarifa base por día. Sólo se les descuenta el 20% si se hospedan más de 9 días.
- Para personas menores de edad se hace un descuento del 50% sobre el precio base, independientemente de los días que se hospeden.
- Se requiere conocer la cantidad que pagar cada una de las personas, así como el descuento que recibe. También se requiere saber el total que pagaron en conjunto (incluido el descuento).

19. Se quiere formar un equipo de desarrolladores de software, conformado por seis personas, el requisito indispensable para formar dicho equipo es una calificación igual o mayor a 90.Si la persona tiene más de 89 de calificación entonces se le preguntará la edad y su domicilio, en caso de no cumplir con este requisito se le indicar que no es aceptado.

Práctica 6 Estructura de Control Cíclica Hacer...Mientras (do while)

Objetivo específico de la práctica

Que el estudiante conozca, identifique y aplique correctamente la estructura de control cíclica Hacer Mientras (do while).

Criterios de desempeño

El estudiante estará capacitado para identificar, declarar y utilizar la estructura de control cíclica Hacer Mientras (do while) cuando:

- Sepa utilizar contadores y acumuladores.
- Memorice el formato para declarar un ciclo con Hacer Mientras (do while).
- Diferencie la estructura cíclica do while de las demás (while y for while).
- Conozca la sintaxis del do while.
- Comprenda el funcionamiento del Hacer Mientras (do while).
- Resuelva problemas mediante un programa informático en que utilice correctamente la estructura de control cíclica Hacer Mientras (do while). Además de usar correctamente las instrucciones de las practicas anteriores.

Resultado esperado en relación a los criterios de desempeño específicos de la práctica:

- La conceptualización por parte del estudiante de los términos de la estructura de control cíclica Hacer Mientras (do while), contador y acumulador.
- Dominio de la sintaxis – formato para usar la estructura de control cíclica Hacer Mientras (do while).
- La ejemplificación de uso de la estructura de control cíclica Hacer Mientras (do while).
- La modificación – corrección de errores de sintaxis relacionados con la estructura de control cíclica Hacer Mientras (do while).
- Aplicación de la habilidad del estudiante para declarar y manipular contadores.
- Aplicación de la habilidad del estudiante para declarar y manipular tanto contadores como acumuladores.
- Aplicación de la habilidad del estudiante para utilizar la estructura de control cíclica Hacer Mientras (do while) para dar solución a un problema mediante un programa informático.

Do while – hacer … mientras

La sentencia do-while se utiliza para especificar un bucle condicional que se ejecuta al menos una vez. Este se suele dar en algunas circunstancias en que precisamente se requiere su ejecución de al menos una vez con posibilidades de más veces.

Su sintaxis podría expresarse de la siguiente forma:

```
....
do{
   ....
   Instrucción(es);
   ....
}while (Exp. booleana);
....
```

La diferencia fundamental entre el while y el do-while es que este último se ejecuta siempre al menos una vez, sea cual sea el resultado de expresión (condición booleana). El ciclo que produce la instrucción hacer mientras (do while) se efectuará, ejecutando las instrucciones que contiene, al menos una vez y continuara ejecutándolas mientras sea verdadera la expresión booleana que lo está controlando.

Cuando la instrucción hacer mientras (do while) evalúe la expresión booleana que controla el ciclo como falsa, la ejecución lógica del algoritmo continuará secuencialmente. Debe estar bien claro que la expresión lógica la cual controla el ciclo do while es evaluada en la inferior, de manera que da paso a ejecutarse al menos una vez el contenido de este.

Al igual que en las demás estructuras de control cíclicas, si la instrucción de control no toma nunca el valor de falso se producirá un ciclo infinito, y habría que interrumpir el programa con la secuencia de teclas Ctrl+Break, o con alguna otra instrucción que lo obligará a salir.

A continuación se presenta un ejemplo que permite sumar los 10 primeros números:

```
void  main ()
{
     int numero = 0, suma = 0;                  //importante la inicialización de
                                                //contador y acumulador
     do{
          numero = numero + 1;                  //incremento de contador
          suma = suma + numero;                 //acumulación
       }while (numero < 10);
     printf ("\n Sumatoria del 1 al 10 =  %d ",suma);
}
```

Completa los siguientes enunciados:

1. El ciclo que produce la instrucción _____ se efectuará, ejecutando las instrucciones que contiene, al menos una vez y continuara ejecutándolas mientras sea verdadera la expresión booleana que lo está controlando.

2. La expresión lógica la cual controla el ciclo es evaluada en la inferior, de manera que da paso a ejecutarse al menos una vez el contenido de este. _____

¿Qué salida a pantalla se producirá con el siguiente código:

```
void  main ()
{
     int numero = 1, suma = 100;
     do{
          numero = numero + 1;
          suma = suma + numero;
       }while (numero < 10);
     printf ("\n Sumatoria del 1 al 10 =  %d ",suma);
}
```

-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-

Resuelva los problemas (de estos problemas no se hicieron bases de casos):

20. Leer siete calificaciones, promediar y pintar el resultado.

21. Leer siete calificaciones, promediar y pintar el resultado. Si el promedio es superior o igual a 60 pintar acredita sino pinta no acredita.

22. Tres personas van de vacaciones a la playa de Nuevo Vallarta, se hospedarán en el hotel de cinco estrellas "Monarcas", en donde el costo por habitación varía según el número de días que se hospede el cliente y la edad de este.

- Personas mayores de 60 años que se hospeden más de 5 días se tiene un costo del precio base menos el 40%.
- Personas mayores de edad y menores de 61 años pagaran la tarifa base por día. Sólo se les descuenta el 20% si se hospedan más de 9 días.
- Para personas menores de edad se hace un descuento del 50% sobre el precio base, independientemente de los días que se hospeden.
- Se requiere conocer la cantidad que pagar cada una de las personas, así como el descuento que recibe. También se requiere saber el total que pagaron en conjunto (incluido el descuento).

23. Se quiere formar un equipo de desarrolladores de software, conformado por seis personas, el requisito indispensable para formar dicho equipo es una calificación igual o mayor a 90.Si la persona tiene más de 89 de calificación entonces se le preguntará la edad y su domicilio, en caso de no cumplir con este requisito se le indicar que no es aceptado.

Práctica 7 Funciones con y sin retorno (Sin parámetros y realizadas por el usuario)

Objetivo específico de la práctica

Con esta práctica se pretende que el estudiante conozca, identifique, cree, declare y utilice funciones internas y definidas por el usuario, sin retorno y sin parámetros, cuando el problema a solucionar mediante un programa así lo requiera.

Criterios de desempeño

El estudiante estará capacitado para identificar, declarar y utilizar las funciones internas y definidas por el usuario, sin - con retorno y sin parámetros cuando:

- Memorice el formato para declarar las funciones internas más comunes de C. con - sin retorno y sin parámetros.
- Aprenda el formato para declarar las funciones definidas por el usuario, con - sin retorno y sin parámetros.
- Distinga una función que no retorna valores y otras que si lo hacen.
- Diferencie una función con parámetros de otras sin ellos.
- Comprenda el funcionamiento de las funciones sin parámetros y sin – con retorno.
- Resuelva problemas mediante un programa informático en que utilice correctamente las funciones desarrolladas por él mismo, sin retorno y sin parámetros. Además de usar correctamente las instrucciones de las practicas anteriores.

Resultado esperado en relación a los criterios de desempeño específicos de la práctica

- La conceptualización por parte del estudiante de los términos función, función interna, función definida por el usuario, retorno, parámetros, variables globales, variables locales.
- Dominio de la sintaxis – formato para usar una función interna de C sin – con parámetro y sin retorno.
- Dominio de la sintaxis – formato para usar una función sin parámetro y sin – con retorno definida por el usuario.
- Ejemplificación de uso de la función sin - con retorno, sin parámetro o ambas.
- La modificación – corrección de errores de sintaxis relacionados con las funciones sin parámetro, sin – con retorno o ambas.
- Aplicación de la habilidad del estudiante para declarar y manipular variables locales y globales, funciones sin - con retorno, sin parámetro o ambas.
- Aplicación de los saberes adquiridos por el estudiante para utilizar las funciones con y sin retorno, sin parámetros o ambas, para dar solución a un problema mediante un programa informático.

Introducción

Una función es un mini programa dentro de un programa. Las funciones contienen varias sentencias bajo un solo nombre, que un programa puede utilizar una o más veces para ejecutar dichas sentencias. Las funciones ahorran espacio reduciendo repeticiones y haciendo más fácil la programación, proporcionando un medio de dividir un proyecto grande en módulos pequeños más manejables (Joyanes y Zahonero; 2003; p.209). Estos mismos autores señalan que la funciones son un conjunto de sentencias que se pueden llamar desde cualquier parte del programa y aclaran que las funciones en C no se pueden anidar, es decir, no se pueden declarar dentro de otra función.

Nota: no confundir función con llamada; una función puede llamar a tantas otras como desee.

Todo compilador comercial trae una gran cantidad de Librerías de toda índole, matemáticas, de entrada - salida, de manejo de textos, de manejo de gráficos, etc, que solucionan la mayor parte de los problemas básicos de programación. Sin embargo será inevitable que en algún momento se tenga que crear sus propias funciones y las indicaciones para lograrlo serán las que veremos más adelante.

Algunos de los conceptos que se deben tener claros al trabajar con funciones son los **argumentos** que se les envían a las funciones, son los VALORES de las variables y NO las variables mismas. En otras palabras, cuando se invoca una función de la forma pow10(a) en realidad se está copiando en el "stack" de la memoria el valor que tiene en ese momento la variable a, la función podrá usar este valor para sus cálculos, pero está garantizado que los mismos no afectan en absoluto a la variable en sí misma.

Como veremos más adelante, es posible que una función modifique a una variable, pero para ello, será necesario comunicarle la DIRECCION EN MEMORIA de dicha variable.Las funciones pueden ó no devolver valores al programa invocante. Hay funciones que tan sólo realizan acciones, como por ejemplo clrscr(), que borra la pantalla de video, y por lo tanto no retornan ningún dato de interés; en cambio otras efectúan cálculos, devolviendo los resultados de los mismos.

La invocación a estos dos tipos de funciones difiere algo, por ejemplo escribiremos:

```
clrscr()                                                    ;

c = getch() ;
```

Donde en el segundo caso el valor retornado por la función se asigna a la variable c. Obviamente ésta deberá tener el tipo correcto para alojarla.

Antes de escribir una función es necesario informarle al Compilador los tamaños de los valores que se le enviarán en el stack y el tamaño de los valores que ella retornará al programa invocante.

Estas informaciones están contenidas en la declaración del **prototipo de la función.**

Ejemplos de prototipos:

```
float funci (int i, double j) ;          // Función con retorno y con parámetros
double funci2 (void) ;     // Función con retorno y sin parámetros
otra_mas (long p);                       // Función con retorno int (por default) y sin
parámetros
void otrafun ();                         // Función sin retorno y sin parámetro
void   lultima (long double sal) ;// Función sin retorno y con parámetros
```

El primer término del prototipo se refiere al tipo del dato que retornará la función; en de no indicarse C considera **por omisión** el tipo int, como el caso del tercer ejemplo. Sin embargo, aunque la función devuelva este tipo de dato, para evitar malas interpretaciones es conveniente explicitarlo.

Cuando no se requiere o espere un retorno de valor, se debe indicar por medio de la palabra **void** – que significa sin valor.

La declaración debe anteceder al cuerpo principal del programa y del mismo cuerpo – definición de la función, por lo regular indicados después de las cabeceras de las librerías. Es normal, por razones de legibilidad de la documentación, encontrar todas las declaraciones de las funciones usadas en el programa, en el HEADER del mismo, junto con los include de los archivos *.h que tienen los prototipos de las funciones de Librería.

Si una ó más de nuestras funciones son usadas habitualmente, podemos disponer su prototipo en un archivo de texto, e incluirlo las veces que necesitemos. Tal y como se muestra en el siguiente ejemplo.

```
//nombre: ejem2.h
// archivo que alberga dos funciones que se llaman desde fun.cpp
# include <stdio.h>

void sal()
{
 int edad;
 printf("\n Edad: ");
 scanf("%d",&edad);
 printf("\n Salario: ");
 }

 int prueba(int ed)
 {
  int salario=0;
  if (ed>=18)
    scanf("%d",&salario);
  else
    printf("\n\n No percibe salario porque es menor de edad: ");
  return salario*2;
 }
```

```
//nombre del archivo: fun.cpp
# include <stdio.h>
# include <conio.h>
# include "ejem2.h"
void main()
{
```

```
  clrscr();
  int salari;
  sal();              // llamado a la función prueba que se encuentra en ejem2.h
  salari=prueba(30);// llamado a la función prueba que se encuentra en ejem2.h
  printf("\n\n El salario ganado es: %d",salari);
  getch();
}
```

La definición debe comenzar con un encabezamiento, que debe coincidir totalmente con el prototipo declarado para la misma, y a continuación del mismo, encerradas por llaves se escribirán las sentencias que la componen.

Variables Globales

Las variables globales se conocen a lo largo de todo el programa y se pueden usar en cualquier parte del código, mantienen su valor durante toda la ejecución del programa, se crean al declararlas en cualquier lugar que esté fuera de la función y pueden ser accedidas por cualquier expresión independientemente de la función en la que se encuentre la función. (Schild; 1995; p. 19,20)

Este tipo de variables son automáticamente inicializadas a cero cuando el programa comienza a ejecutarse. Son accesibles a todas las funciones que estén declaradas en el mismo, por lo que cualquiera de ellas podrá actuar sobre el valor de las mismas.

Las variables globales son muy útiles cuando se usan los mismos datos en varias funciones del programa. Sin embargo se debe evitar el uso de variables innecesarias por lo siguiente:

1. Utilizan memoria todo el tiempo de ejecución del programa, no solo cuando se necesita.
2. El uso de una variable global cuando se podría usar una local resta generalidad a la función, ya que depende de algo que debe estar definido fuera de ella.
3. El uso de un gran número de variables globales puede producir errores en el programa debido a efectos secundarios desconocidos y no deseados. (Schild; 1995; p. 21)

Variables Locales

Las variables que se declaran dentro de una función se denominan variables locales o variables automáticas, debido a que en C se puede usar la palabra clave opcional auto para declararlas. Estas variables pueden ser referenciadas sólo por sentencias que estén dentro del bloque en el que han sido declaradas es decir no son reconocidas fuera de su propio bloque de código. Sólo existen mientras se está ejecutando el bloque en el que son declaradas es decir se crea al entrar al bloque y se destruye al salir de él. Conociéndose ese bloque como función. (Schild; 1995; p. 17)

A estas variables se les destina un espacio cuando se las define dentro de una función, y dicho espacio se borra cuando la misma devuelve el control del programa, a quien la haya invocado, permitiendo que estas no ocupen memoria simultáneamente en el tiempo, y solo la ocupen cuando se las necesita, para luego, una vez usadas desaparecer.

El identificador ó nombre que se la haya dado a una variable es sólo relevante entonces, para la función que la haya definido, pudiendo existir entonces variables que tengan el mismo nombre, pero definidas en funciones distintas, sin que haya peligro alguno de confusión .

La ubicación de estas variables locales, se crea en el momento de correr el programa, por lo que no poseen una dirección prefijada, esto impide que el compilador las pueda inicializar previamente. Reacuérdese entonces que, si no se las inicializa expresamente en el momento de su definición, su valor será indeterminado - basura.

Estructura de una función realizada por el usuario sin parámetros y con retorno

```
Tipoderetorno nombredelafuncion()
{
        cuerpo de la función

}
```

Primero se indica el tipo de dato que esperamos se retorne, cuando se omite el tipo de dato C considera como retorno por omisión un valor entero. El nombre de la función debe cumplir con los mismos requisitos de declaración de un identificador, enseguida de nombre se colocan paréntesis, los cuales en este caso no lleva parámetros.

Ejemplo

```
        #include <stdio.h>
#include <conio.h>

int suma();             //prototipo de una función con retorno y sin parámetros

void main()
{
 clrscr();
 int su;
 su=suma();             //llamado a la función que al ejecutarse retornan un valor
                         //que es asignado a la variable su que se muestra enseguida
 printf("\n Resultado de la suma: %d",su);
 getch();
}

int suma()              // nombre y cuerpo de la función, observe que no lleva el ;
al final
{
 int a,b,sum;
 printf("\n introduce el primer valor: ");          scanf("%d",&a);
 printf("\n introduce el segundo valor: ");   scanf("%d",&b);
 sum=a+b;
 return sum
}
```

El prototipo lleva incluido el tipo de dato de retorno, nombre de la función y los paréntesis, seguido por el ;. El prototipo se indica antes del cuerpo principal del programa.

El llamado se la función se hizo dentro del cuerpo del programa principal, asignando el valor retornado a una variable que más tarde se pinto mediante el printf().

Estructura de una función realizada por el usuario sin parámetros y sin retorno

```
void nombredelafuncion()

{
        cuerpo de la función
        .
        .
}
```

Ejemplo:

```
#include <stdio.h>
#include <conio.h>

void suma();            //prototipo de una función sin y sin parámetros, observe que

void main()
{
 clrscr();
 int su;
 suma();                //llamado a la función que al ejecutarse no retorna un valor
                        //y el resultado de la suma es mostrado por la misma función
 getch();
}

void suma()             //nombre y cuerpo de la función
{
 int a,b,sum;
 printf("\n introduce el primer valor: ");              scanf("%d",&a);
 printf("\n introduce el segundo valor: ");   scanf("%d",&b);
 sum=a+b;
 printf("\n Resultado de suma %d más %d= %d",a,b,sum);
}
```

El prototipo lleva incluido la palabra void porque está función no retorna un valor, nombre de la función y los paréntesis, seguido por el **;**.

El llamado se la función se hizo dentro del cuerpo del programa principal, sin asignar ningún valor a ninguna variable ya que la función no retorna valor alguno debido a que se muestran dentro del cuerpo de la función con la instrucción printf(). .

-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-

Resuelva los problemas

24. Leer siete calificaciones, promediar y pintar el resultado. utilizando funciones realizadas por el usuario sin retorno y sin parámetro.
25. Leer siete calificaciones, promediar y pintar el resultado. Si el promedio es superior o igual a 60 pintar acredita sino pinta no acredita. utilizando funciones realizadas por el usuario con retorno y sin parámetro.
26. Tres personas van de vacaciones a la playa de Nuevo Vallarta, se hospedarán en el hotel de cinco estrellas "Monarcas", en donde el costo por habitación varía según el número de días que se hospede el cliente y la edad de este.

- Personas mayores de 60 años que se hospeden más de 5 días se tiene un costo del precio base menos el 40%.
- Personas mayores de edad y menores de 61 años pagaran la tarifa base por día. Sólo se les descuenta el 20% si se hospedan más de 9 días.
- Para personas menores de edad se hace un descuento del 50% sobre el precio base, independientemente de los días que se hospeden.
- Se requiere conocer la cantidad que pagar cada una de las personas, así como el descuento que recibe. También se requiere saber el total que pagaron en conjunto (incluido el descuento). Utilizando funciones realizadas por el usuario con retorno y con parámetro.

27. Se quiere formar un equipo de desarrolladores de software, conformado por seis personas, el requisito indispensable para formar dicho equipo es una calificación igual o mayor a 90. Si la persona tiene más de 89 de calificación entonces se le preguntará la edad y su domicilio, en caso de no cumplir con este requisito se le indicar que no es aceptado. Utilizando funciones realizadas por el usuario con retorno y sin parámetros.

- Realiza un programa para una tienda mexicana que vende productos nacional e internacionalmente, en el cual es necesario cotizar a diferentes monedas partiendo de un precio en pesos; para ello el encargado de mostrador tendrá que introducir el nombre del producto y el precio del mismo para que el programa se encargue de hacer las cotizaciones después de hacer el incremento del IVA. Mostrando como salida los datos del producto adquirido, cantidad, iva, total a pagar, tomando en consideración la cotización de las diferentes monedas en que se vende el producto

  Cotizaciones:
  Dólar = 11
  Marco = 9
  Libra = 5

  Utilizando funciones realizadas por el usuario con retorno y con parámetro.

- Realiza un programa donde pidas el nombre de un alumno junto con 5 calificaciones utilizando un for; posteriormente indiques si aprobó o no, tomando en cuenta que la cantidad mínima aprobatoria es de 70, en caso de aprobar que indique Excelente si obtuvo un 10, muy bien si es 9, bien si es 8 o bien regular si es 7. El usuario podrá calcular los promedios hasta que decida ya no hacerlo. Utiliza diferentes funciones para la solución al problema y mostrar los mensajes (nombre del alumno, promedio y mensaje correspondiente).

– Realiza un programa llamada “PEDIDOS” el cual utilice funciones para realizar sus procesos de cotización dentro de una tienda departamental con venta seccionada en tres funciones: medicina, electrónica y ropa. Cada rubro será el nombre de una función, el usuario se encargara de decidir que producto desea adquirir, dependiendo del departamento al que pertenezca se hará un descuento de

Medicina 5%	Jarabe $50	Electrónica 20%	Computador $ 8500	Ropa 10%	Vestido $ 1750
	Ampolleta $30		Scanner $ 2500		Pantalón $ 450
	Capsulas $80		Impresora $ 1800		Blusa $ 320

Dando como salida el nombre del departamento que vende, el nombre y la cantidad de productos comprados, junto con el descuento dependiendo del departamento. Utilizando funciones realizadas por el usuario con retorno y con parámetros.

Práctica 8 Funciones con retorno y con parámetros.Funciones sin retorno y con parámetros (realizadas por el usuario)

Objetivo específico de la práctica

Que el estudiante conozca, identifique, cree, declare y utilice funciones internas y definidas por el usuario, con – sin retorno y con parámetros, cuando el problema a solucionar mediante un programa así lo requiera.

Criterios de desempeño

El estudiante estará capacitado para identificar, declarar y utilizar las funciones internas y definidas por el usuario, con - sin retorno y con parámetros cuando:

- Memorice el formato para declarar las funciones internas más comunes de C. con - sin retorno y con parámetros.
- Aprenda el formato para declarar las funciones definidas por el usuario, con - sin retorno y con parámetros.
- Distinga una función que no retorna valores y otras que si lo hace.
- Diferencie una función con parámetros de otras sin ellos.
- Comprenda el funcionamiento de las funciones sin – con parámetros y sin – con retorno.
- Resuelva problemas mediante un programa informático en que utilice correctamente las funciones desarrolladas por él mismo, con - sin retorno y sin - con parámetros. Además de usar correctamente las instrucciones de las practicas anteriores.

Resultado esperado en relación a los criterios de desempeño específicos de la práctica

- La conceptualización por parte del estudiante de los términos función, función interna, función definida por el usuario, retorno, parámetros, variables globales, variables locales.
- Dominio de la sintaxis – formato para usar una función interna de C sin – con parámetro y sin – con retorno
- Dominio de la sintaxis – formato para usar una función sin - con parámetro y sin – con retorno definida por el usuario
- Ejemplificación de uso de la función sin - con retorno, sin – con parámetro o ambas.
- La modificación – corrección de errores de sintaxis relacionados con las funciones sin parámetro, sin – con retorno o ambas
- Aplicación de la habilidad del estudiante para declarar y manipular variables locales y globales, funciones sin - con retorno, sin – con parámetro o ambas.
- Aplicación de la habilidad del estudiante para utilizar las funciones con y sin retorno, sin – con parámetros o ambas, para dar solución a un problema mediante un programa informático.

Introducción

La parte teórica aplica la misma de la práctica anterior, por eso se aborda directamente desde el formato y ejemplos.

Funciones con retorno y con parámetros

```
#include <stdio.h>
#include <conio.h>

int  suma(int a, int b);              // Prototipos de las funciones, observe que se espera retorno entero
int multi(int c, int d);              // y ambas funciones esperan recibir dos valores

main()
{
 clrscr();
 int sum, num1, num2;
 printf("\n Introduce el primer número: ");          scanf("%d",&num1);
 printf("\n Introduce el segundo número: ");         scanf("%d",&num2);
 sum=suma(num1,num2);        // llamado a la función suma, la cual envia dos valores enteros
                             // a la función y espera le retorne un valor que asignará a sum.
                             // Mientras que la función multi es llamada dentro de la función
                             // printf() y se pinta directamente el valor que retorno sin
                             // asignarse a ninguna variable
 printf("\n\nLa suma de %d más %d es igual a %d",num1,num2,sum);
 printf("\n\nLa multiplicacción de %d por %d es igual a %d",num1,num2,mul(num1,num2));
 return 0;
}

int suma (int a, int b)             //cuerpo de la función suma, observe que los parámetros son del
{                          //mismo tipo que los valores que le enviaron desde el cuerpo principal
 return a+b;               //del programa al llamarla y que el valor retornado es del mismo tipo
}                                   //que se indico se esperaba el retorno

int multi(int c, int d)             //cuerpo de la función multi
{
 return                                                                          c*d;
}
```

Funciones sin retorno y con parámetros

Analicemos por medio de un ejemplo dichas funciones:

```
#include <stdio.h>
#include <conio.h>

void suma(int a, int b);// Prototipos de las funciones, observe que no se espera retorno

void main()
{
 clrscr();
 int sum, num1, num2;
 printf("\n Introduce el primer número: ");          scanf("%d",&num1);
 printf("\n Introduce el segundo número: ");  scanf("%d",&num2);
 suma(num1,num2);   // llamado a la función suma, la cual envia dos valores enteros
                    //observe que no se espera retorno por lo tanto no se asigna a ninguna
                    //variable, mientras que al ejecutarse la función se pinta el resultado
                    // de la suma de los dos valores enviados desde el cuerpo principal del
                    // programa y recibidos y utilizados en la función
}

void suma (int a, int b)
{
 int su;
```

```
 su=a+b;
 printf("\n la suma de %d + %d = %d",a,b,su);
}
```

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

Resuelva los problemas:

28. Leer siete calificaciones, promediar y pintar el resultado. Si el promedio es superior o igual a 60 pintar acredita sino pinta no acredita. utilizando funciones realizadas por el usuario con retorno y sin parámetro

29. Tres personas van de vacaciones a la playa de Nuevo Vallarta, se hospedarán en el hotel de cinco estrellas "Monarcas", en donde el costo por habitación varía según el número de días que se hospede el cliente y la edad de este.

- Personas mayores de 60 años que se hospeden más de 5 días se tiene un costo del precio base menos el 40%.
- Personas mayores de edad y menores de 61 años pagaran la tarifa base por día. Sólo se les descuenta el 20% si se hospedan más de 9 días.
- Para personas menores de edad se hace un descuento del 50% sobre el precio base, independientemente de los días que se hospeden.
- Se requiere conocer la cantidad que pagar cada una de las personas, así como el descuento que recibe. También se requiere saber el total que pagaron en conjunto (incluido el descuento). Utilizando funciones realizadas por el usuario con retorno y con parámetro

30. Realice un programa que calcule el total a pagar por la familia Suárez Martínez, ellos tienen 4 hijos, pero sólo los que tienen edades de 18 a 35 años reciben una beca del 50% siempre y cuando el salario que percibe la Familia sea inferior a los 5000 pesos mensuales. Se quiere conocer la cantidad que pagará la familia de colegiatura por cada uno de sus hijos, la cantidad que reciben en becas y el total a pagar por los cuatro. Solucione el problema utilizando funciones.

Completa los siguientes enunciados:

1.- Escribe la sintaxis para declarar una función llamada suma sin parámetros y sin retorno. ____

2.- Escribe la sintaxis para declarar una función llamada result que envíe los parámetros enteros x,y ____

3.- Escribe la sintaxis para llamar la función suma. ______________________________

4.- Escribe la sintaxis para declarar una función llamada pago que envié los parámetros cant, precio y retorne total. ____

5.- Escribe la instrucción para llamar a la función pago. __________________________

Práctica 9 Arreglos Unidimensionales

Objetivo específico de la práctica

Que el estudiante realice una gran cantidad de prácticas para que conozca, identifique, cree, declare y utilice arreglos unidimensionales para cuando se le presente un problema de arreglos unidimensionales le pueda solucionar.

Criterios de desempeño

El estudiante estará capacitado para identificar, declarar y utilizar arreglos unidimensionales cuando:

- Memorice el formato para declarar un arreglo unidimensional con valores constantes.
- Aprenda la forma de declarar, asignar, leer y escribir un arreglo unidimensional.
- Distinga un arreglo unidimensional de uno bidimensional (matriz).
- Comprenda el funcionamiento de los arreglos.
- Sepa determinar cuándo utilizar los arreglos y cuando no.
- Utilice adecuadamente los arreglos en operaciones aritméticas o lógicas.
- Resuelva problemas mediante un programa informático en que utilice correctamente uno o más arreglos unidimensionales. Además de usar correctamente las instrucciones de las practicas anteriores.

Resultado esperado en relación a los criterios de desempeño específicos de la práctica.

- La conceptualización por parte del estudiante de los términos arreglo, arreglo unidimensional, arreglo bidimensional, dimensionamiento.
- Dominio de la sintaxis – formato para usar arreglos unidimensionales.
- Ejemplificación de uso de los arreglos unidimensionales.
- La modificación – corrección de errores de sintaxis relacionados con los arreglos unidimensionales.
- Lectura y almacenaje de valores en las posiciones correspondientes de un arreglo.
- Realizar operaciones con los valores contenidos en un arreglo unidimensional.
- Presentación en pantalla del contenido de uno o más arreglos unidimensionales.
- Aplicación de los saberes del estudiante para declarar y manipular arreglos unidimensionales.
- Aplicación del conocimiento del estudiante para utilizar los arreglos unidimensionales, para dar solución a un problema mediante un programa informático.

Introducción

Existen diversos tipos de datos estructurados, uno de ellos son los arrays.Los **arrays** permiten agrupar datos usando un mismo identificador. Todos los elementos de un array son del mismo tipo, y para acceder a cada elemento se usan subíndices.

Los valores para el número de elementos deben ser constantes, y se pueden usar tantas dimensiones como queramos, limitado sólo por la memoria disponible.

Arreglos Unidimensionales

Cuando sólo se usa una dimensión se suele hablar de listas o vectores, cuando se usan dos, de tablas.

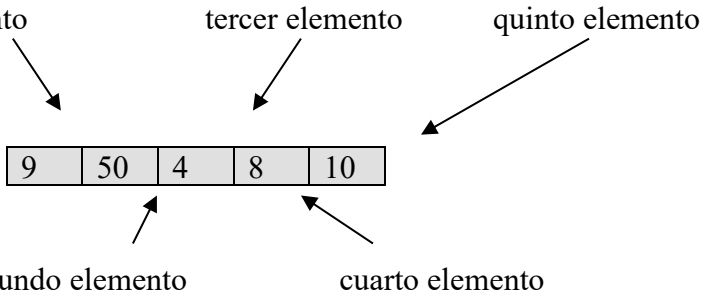
Un Array es una colección de datos del mismo tipo, que se almacena en posiciones consecutivas de memoria y recibe un nombre común (Joyanes, 2000). Para referirse a un determinado elemento de un array se deberá utilizar un índice, que especifique su posición relativa en el array. En donde los arrays pueden ser unidimensionales, también llamados Vectores.

Todo identificador debe ser declarado, los Array no son una excepción y lo primero que se debe hacer es crear el tipo, para luego poder declarar datos de dicho tipo. (Joyanes, 2000).

Sintaxis:

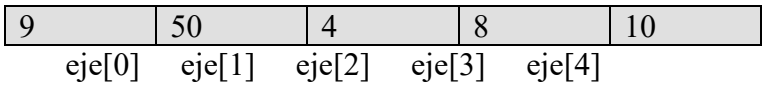
<tipo> <identificador>**[**<núm_elemen>**]**;

Ejemplo: int eje[5];



Para accesar un elemento específico del arreglo se hace referencia a su número de celda o valor del índice. El índice para un arreglo debe ser de tipo entero.

vector: int eje[5];



eje[0] se refiere al primer elementos del vector y tiene almacenado el entero 9, el número 0 es el índice. El valor de 9 ha podido ser asignado directamente en el código por medio de la instrucción:

eje[0]=9;

X[2] es el tercer elemento del vector y tiene almacenado un valor de 50, el número 2 es el índice. El valor de 50 en la posición 3 a podido ser pintado, por ejemplo por teclado con la siguiente instrucción:

printf(“%d”,eje[2]);

eje[4] es el quinto elemento del vector y tiene almacenado un valor de 10, el número 4 es el índice. El valor de 10 en la posición 4 a podido ser leido, por ejemplo por teclado con la siguiente instrucción:

scanf(“%d”,eje[4])

Cada elemento del vector eje, al ser tipo int, ocupa en memoria 2 byte, por lo tanto, el arreglo eje ocupará 2 * 5 = 10 bytes en la memoria RAM del computador. Similarmente podríamos hablar para vectores de otros tipos de datos.

Cuando se declara un arreglo, se declara el tipo de datos del arreglo y su tamaño. Ejemplos de eso puden ser las siguientes declaraciones:

```
int num[5], salarios[10];
float precio[100], IVA[100];
float mensualidades[max];
char nombres[20], apellidos[25];
char letras[26];
```

Como se ve, el número de elementos del arreglo estará dado por el número colocado entre [ ] al declararlo, esto se llama **dimensionamiento**. Este, el dimensionamiento, es obligatorio, a menos que se dé implícitamente el tamaño del arreglo por medio de la inicialización del mismo como en:

```
char Opciones[] =  {'a', 'b', 'c', 'd'};
int Enteros[] = {4, 13, 0, 5, 6, -23, 7, 345, 45, -12};
```

En el primer ejemplo, Operador será un arreglo de cuatro posiciones, y queda inicializado con los correspondientes caracteres dados entre llaves. El segundo ejemplo el vector Enteros tendrá 10 posiciones.

Los elementos de una variable arreglo, no reciben valores cuando el arreglo se declara (tendrán "basura"). Por esto es conveniente inicializar o efectuar lectura, asignándoles valores antes de efectuar ninguna otra operación con ellos.

A los elementos de un arreglo se asignan valores de forma parecida que a cualquier otra variable.

```
Numeros[3] = 150;
Mensualidad[I] = Anualidades[I-1] * 1.5;
Precio[artículo] = costo[articulo] + numero[articulo];
Printf( "Precio máximo = $ %d", Precio[articulo]);
scanf("%d",&Numeros[i]);
gets(nombres[persona]);
```

En los arreglos anteriores, los índices: I, artículo y persona, deben tener valores enteros, los cuales coincidan con las posiciones válidas del arreglo correspondiente. Recuerde si el arreglo es de N elementos, las posiciones válidas van desde el cero hasta N menos uno.

-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-

Crea las siguientes instrucciones

1. Declara un arreglo para almacenar 50 números enteros
   Escribe la instrucción:

2. Declara un arreglo para almacenar 30 números reales
   Escribe la instrucción:

3. Declara un arreglo para almacenar la palabra hola mundo
   Escribe la instrucción:

4. Crea la instrucción para almacenar el dato 15 en la posición 5 de un arreglo A.
   Int A[6];
   Escribe la instrucción:

5. Crea la instrucción para almacenar el carácter o en la posición 4 del arreglo A
   Char A[5];
   Escribe la instrucción:

6. Crea la instrucción para sumar la posición 3 y 5 del arreglo A.
   Int A[ ]=2,4,6,8,10,12;
   Escribe la instrucción:

9. Crear la instrucción para sumar la posición 2 del arreglo A y la posición 3 del Arreglo B. en la posición 1 del arreglo A.
   Int A[ ]=2,4,5,6,7;
   Int B[ ]=3,6,9,12,15;
   Escribe la instrucción:

10. Crea la Instrucción para restar el valor de la posición 4 del arreglo B con el valor de la posición 1 del arreglo A.
    Int A[ ]=2,4,5,6,7;
    Int B[ ]=3,6,9,12,15;
    Escribe la instrucción:

12. Corrige la siguiente instrucción tiene un error de sintaxis.
    FloatP[6];
    Escribre la instrucción correcta:

Explica, documenta y realiza el siguiente programa. (Archivo PAU3.cpp)

```
#include <stdio.h>     //librerías
#include <conio.h>     //librerías

void main()                    //funcción principal
{
 int cues[2]={3,1};          int resp[2];        int opc,i,bien=0; // declaración de variables
 clrscr();                                                          // limpia pantalla
 printf("\n Presione el número de la opción más adecuada a la pregunta: \n ");

for(i=0;i<2;i++)
 {

switch(i)
   {
   case 0: printf("\n %d.- Equivale a 1024 bytes: ",i+1);
               printf("\n 1)Mb      2)Gb         3)Kb \nRespuesta: ");
               scanf("%d",&opc);     resp[i]=opc;
               if(opc==cues[i])
               {
                   bien++;
                   printf("\n                    CORRECTO");
               }
               else
                   printf("\n                    INCORRECTO");
               break;
   case 1: printf("\n %d.- Su valor puede cambiar, no así su nombre: ",i+1);
               printf("\n 1)Variable   2)Constante        3)Función  \nRespuesta: ");
               scanf("%d",&opc);              resp[i]=opc;
               if(opc==cues[i])
               {
                   bien++;
                   printf("\n                    CORRECTO");
               }
               else
                   printf("\n                    INCORRECTO");
               break;
   }
  }

  printf("\n El número de aciertos son: %d", bien);
  getch();
}
```

-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-

31. Realice un programa que calcule el total a pagar por la familia Suárez Martínez, ellos tienen 4 hijos, pero sólo los que tienen edades de 18 a 35 años reciben una beca del 50% siempre y cuando el salario que percibe la Familia sea inferior a los 5000 pesos mensuales. Se quiere conocer la cantidad que pagará la familia de colegiatura por cada uno de sus hijos, la cantidad que reciben en becas y el total a pagar por los cuatro.

32. Termine el programa propuesto en el ejemplo, ahora integre otras 8 preguntas relacionadas con la informática, muestre el número de aciertos y desaciertos, así como el número de pregunta en que la respuesta fue incorrecta (PAU.CPP).

Explica y documenta lo que realiza el siguiente programa.

```
#include <stdio.h>
#include < stdlib.h>
Int main(void)
{
   Int x, tabla[100];
   for (x=1;x<=100; x++)
          {
          tabla[x]=x;
          }

   for (x=1;x<=100; x++)
          {
          printf("%d\n",tabla[x]);
          }
  system ("pause");
  return 0;
}
```

Explica y documenta lo que realiza el siguiente programa. (Salón de clases o Centro de cómputo) (Archivo PAU6.cpp)

```
#include <stdio.h>
#include < stdlib.h>
Int main(void)
{
   Int x, tablas[100];
   for (x=1;x<=100; x++)
          {
          tabla[x]=x;
          }

   for (x=100;x>=1; x--)
          {
          printf("%d\n",tabla[x]);
          }
  system ("pause");
  return 0;
}
```

Explica y documenta lo que realiza el siguiente programa. (Salón de clases o Centro de cómputo) (Archivo PAU7.cpp)

```
#include <stdio.h>
#include < stdlib.h>
int main()
{
  Int aux, números[10];
  Int i,j,n=10;
 for (i=0; i<n; i++)
          {
          printf("Escriba un número");
          scanf ("%f",&números[i]);
          }

  for (i=0; i<n-1; i++)
   {
  for(j=i+1; j<n; j++)
         {
```

```
          If(números[i]<numeros[j])
        {
            aux=numeros[i];
           numeros[i]=numeros[j];
            numeros[j]=aux;
        }
        }
  }

 for (i=n-1; i>=0;i--)
 {
    printf("%f \n",números[i]);
 }
 System ("PAUSE");
 Return 0;
}
```

Explica y documenta lo que realiza el siguiente programa. (Salón de clases o Centro de cómputo (editarlo y copilarlo) (Archivo PAU8.cpp)

```
#include <stdio.h>
#include < stdlib.h>
int main()
{
   Int aux, numero1[5], numero2[5],numero3[10];
  Int i,j;
        for (i=0; i<5; i++)
        {
        printf("Escriba un número");
        scanf ("%d",&numero1[i]);
        }

        for (i=0; i<5; i++)
        {
        printf("Escriba un número");
        scanf ("%d",&numero2[i]);
        }

for (i=0; i<5; i++)
        {
          numero3[i]=numero1[i];
```

```
        }
for (i=0; i<5; i++)
        {
          numero3[5+i]=numero2[i];
        }

for (i=0; i<10; i++)
        {
         printf("%d/n",numero3[i]);
                }

  system ("pause");
  return 0;
}
```

Práctica 10 Arreglos Bidimensionales

Objetivo específico de la práctica

El estudiante conozca, identifique, cree, declare y utilice correctamente arreglos bidimensionales cuando el problema a solucionar mediante un programa así lo requiera.

Criterios de desempeño

El estudiante estará capacitado para identificar, declarar y utilizar arreglos bidimensionales cuando:

- Comprenda qué es una dimensión o más.
- Memorice el formato para declarar un arreglo bidimensional con valores constantes.
- Aprenda el formato para declarar un arreglo bidimensional.
- Distinga un arreglo unidimensional de uno bidimensional (matriz), tridimensional, etc..
- Comprenda el funcionamiento de los arreglos unidimensionales y bidimensionales principalmente.
- Sepa determinar cuándo utilizar los arreglos y cuando no.
- Resuelva problemas mediante un programa informático en que utilice correctamente uno o más arreglos unidimensionales y bidimensionales. Además de usar correctamente las instrucciones de las practicas anteriores.

Resultado esperado en relación a los criterios de desempeño específicos de las prácticas

- La conceptualización por parte del estudiante de los términos arreglo, arreglo unidimensional, arreglo bidimensional, dimensionamiento.
- Dominio de la sintaxis – formato para usar arreglos bidimensionales.
- Ejemplificación de uso de los arreglos bidimensionales.
- La modificación – corrección de errores de sintaxis relacionados con los arreglos bidimensionales.
- Lectura y almacenaje valores en las posiciones correspondientes de un arreglo bidimensional.
- Realizar operaciones con los valores contenidos en un arreglo bidimensional.
- Presentación en pantalla del contenido de uno o más arreglos bidimensionales.
- Aplicación de la habilidad del estudiante para declarar y manipular arreglos bidimensionales.
- Aplicación de la habilidad del estudiante para utilizar los arreglos bidimensionales, para dar solución a un problema mediante un programa informático.

Arreglos Multidimensionales

Los arreglos pueden tener múltiples dimensiones, aquí trataremos solamente los de dos dimensiones, mejor conocidos como matrices o tablas.

Un Array bidimensional es un vector de vectores. Es por tanto un conjunto de elementos del mismo tipo en que el orden de los componentes es significativo y en el que se necesitan especificar dos subíndices para poder identificar a cada elemento del array. (Joyanes, 2000).

	1	2	…	1,j
1	7,5	2,3	…	…
2	3,4	6,3	…	…
…	…	…	…	…
i				m[i ,j]

El dimensionamiento en lenguaje C/C++ se efectua de la siguiente forma: se coloca el tipo, a continuación el identificador correspondiente para el nombre de la variable y luego encerrado entre llaves [ ] el tamaño de las filas y el tamaño de las columnas, así por ejemplo se podría dar esta definición:

int Matriz[5][6];

Esta sería una variable estructurada compuesta por 5 filas y 6 columnas, de la cual nos hacemos la siguiente idea o abstracción:

	0	**1**	**2**	**3**	**4**	**5**
0	34	23	67	123	1	5
1	0	-5	45	87	2	4
2	76	15	13	111	3	3
3	53	69	-7	73	4	2
4	19	23	0	33	5	1

Las cinco filas se numeran desde el 0 hasta el 4, las seis columnas se numerarán desde el 0 hasta el 5. De esta forma en la fila 0 columna 0 está almacenado el número entero 34, está posición se denotará como Matriz[0][0], y se lee como "Matriz su cero cero".

En la posición Matriz[4][1], el 4 corresponde a la fila y el 1 corresponde a la columna, está almacenado un 23.

-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-0-

Crea las siguientes Instrucciones

1. Declara un Matriz de 2 por 2 para almacenar números enteros
   Escribe la instrucción:

2. Declara un Matriz de 3 por 4 almacenar números reales
   Escribe la instrucción:

3. Declara un matriz de 5 por 5 para almacenar números enteros largos.

Escribe la instrucción:

4. Crea la instrucción para almacenar el dato 15 en la posición fila 2, columna 2 de la Matriz A
   Int A[6][6];
   Escribe la instrucción:

5. Crear la instrucción de una Matriz de 4 por 4 y Crea la instrucción para almacenar el dato 6.3 en la posición Fila 4, columna 2 de la matriz A.
   Escribe la instrucción:
   Escribe la instrucción:

6. Crea la instrucción para sumar el valor de la fila 1 y columna 1 de la matriz M1. con lel valor de la fila 1 y la columna 1 de la M2;
   Int M1[2][2];
   Int M2[2][2];
   Escribe la instrucción:

7. Crear la instrucción para multiplicar el valor de la fila 1 columna 1 de la matriz A y el valor de la fila 0 y la columna 0 de la Matriz B en una variable X.
   Int A[2][2];
   Int B[2][2];
   Escribe la instrucción:

8. Crea la Instrucción para comparar el valor de la fila 1 columna 1 de la matriz A y el valor de la fila 0 y la columna 0 de la Matriz B. Si es verdad que escribe el valor de la matriz A es Mayor.
   Int A[2][2];
   Int B[2][2];
   Escribe la instrucción:

Explica lo que hace el siguiente Programa, documenta y para que se utiliza los arreglos bidimensionales

```
#include <stdio.h>
#include <conio.h>
void main()
{
  clrscr();
  int fil,fil2,n,x=3,y=4;
  float cal, acu=0, pro=0;
  int objeto[5][3];
   clrscr();
   printf(" \n\n MATERIAS              ESTUDIANTES        PROMEDIO\n");
   printf("                              1          2          3          \n\n");
   y=7;
   for(fil=0;fil<5;fil++)
   {
     x=3;            gotoxy(x,y);
     printf("\n   %d .-              ",fil+1);
     acu=0;                   pro=0;     x=25;
     //////////////////////////////////////////////////////////////////////
     for(fil2=0;fil2<3;fil2++)
     {
       gotoxy(x,y+1);
       scanf("%d",&objeto[fil][fil2]);
       acu=acu+objeto[fil][fil2];
       x=x+8;
      }
     y++;
pro=acu/3;       gotoxy(x,y);    printf(" = %.2f",pro);
  }
  getch();
}
```

-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-o-

33. Cuatro hermanos van de vacaciones con sus respectivas familias a Carelles por un número desconocido de días, llevan a sus esposas y 2 hijos cada uno de ellos. Por lo tanto, el precio por días se tiene que preguntar para cada familia y varía según la edad y sexo de cada persona.

a. La mujeres mayores de edad que tengan menos de 50 años reciben 20% de descuento
b. Los hombres menores de edad reciben un 30% de descuento

- Se quiere conocer el total que pagar cada uno de los hermanos - familia,
- considerando el descuento recibido
- El descuento que recibió cada hermano - familia
- El total que pagaran en conjunto todos los hermanos - familias
- El número en general de mujeres que recibieron descuento
- El número de hombres en general que recibieron descuento
- La cantidad que paga cada persona (cada una de las 16 personas) – matriz
- (Realice el programa primero sin funciones y luego trate de realizarlo utilizando funciones)

34 Realice un programa en donde se capturen los votos de una muestra realizada a 10 personas de cada municipio y muestre los resultados obtenidos para los partidos políticos del PAN, PRI y PRD en los municipios de Ahuacatlán, Ixtlán, Jala y Amatlán.

- Un programa pida datos al usuario los datos de una matriz de 2x2 y muestra su traspuesta (el resultado de intercambiar filas por columnas).
- Un programa que pida al usuario los datos de una matriz de 3x3, y calcule y muestre su matriz inversa.
- Un programa que pida al usuario los datos de dos matrices de 2x2, y calcule y muestre su producto.

Referencias

CASTRO J., Cucker F., Messeguer X., Rubio A., Solano L. y Valles B. (1993). Curso de Programación. Madrid España. McGraw Hill p. 4.

CANELLO P., Primeros pasos en C, Cursos de Computación. http://www.pablin.com.ar/computer/cursos

CETTICO Centro de Trasferencia Tecnológica en Informática y Comunicaciones Universidad Politécnica de Madrid. (1999). Programación sin secretos. Madrid, España. p. 41,193.

JOYANES L. (1994). Fundamentos de Programación. Algoritmos y Estructura de Datos. Madrid España. McGraw-Hill. p. 18, 22.

JOYANES L., Rodríguez L., Fernández M. (2000). Fundamentos de programación: Libro de problemas. McGraw-Hill. España.

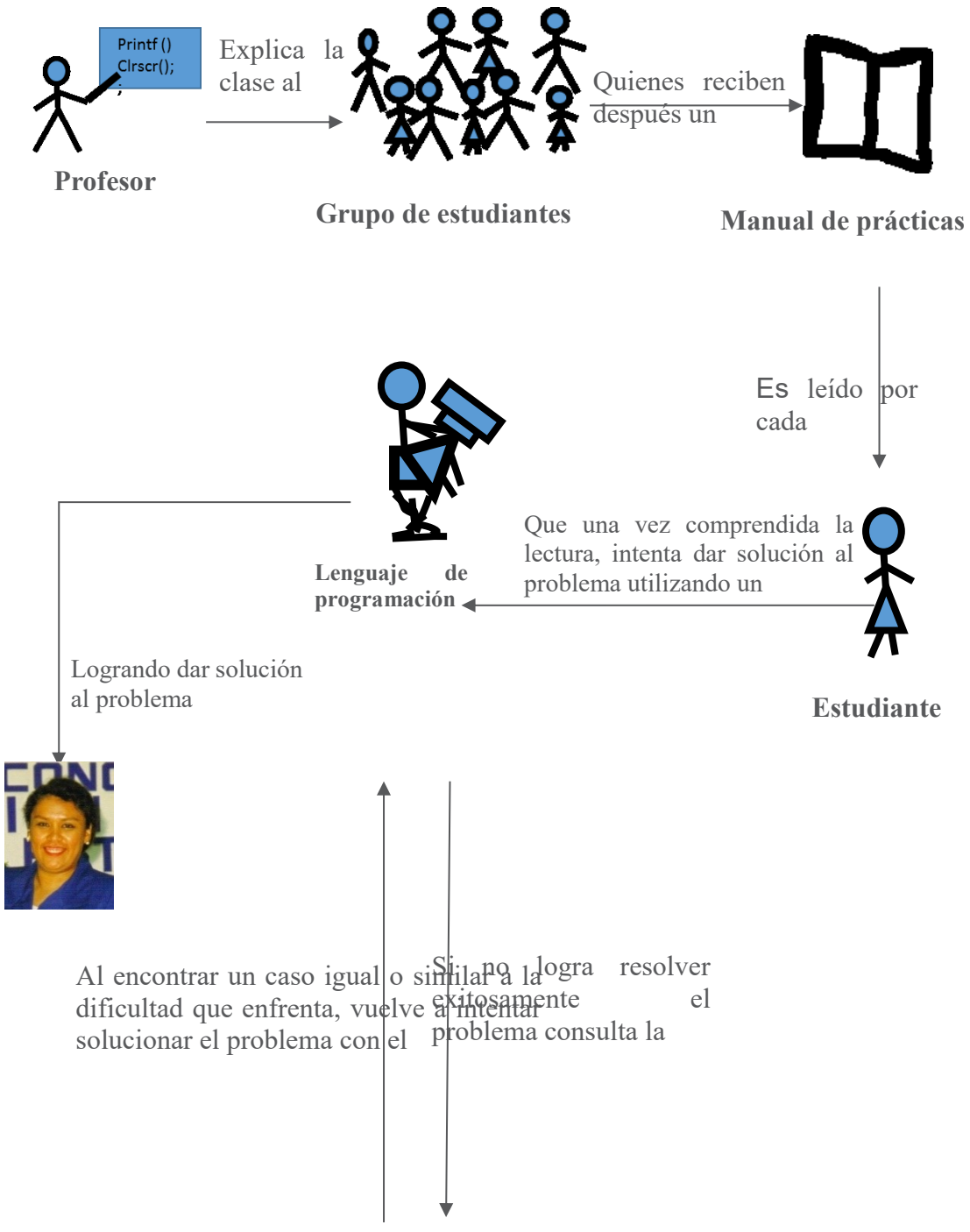
JOYANES L., Zahonero I. (2003). Programación en C "metodología, estructura de Datos y objetos. McGraw-Hill. p. 19,108 y 209.

ZÚÑIGA R. Aplicación de las Técnicas de Razonamientos Basado en Casos como Herramienta de Apoyo en la Enseñanza de Programación. [Tesis de Maestría]. Universidad de la Habana; 2006.

Anexos

Aclaración: La confiabilidad del material sacado de http://www.pablin.com.ar/computer/cursos fue comprobado para mayor credibilidad del manual, debido a que el portal no incluye los elementos de calidad característicos de un sitio web de su tipo. Sin embargo, los conceptos extraído para integrarlo en algunas prácticas si cumplen con tales requerimientos.

Metodología para usar el material de apoyo de la enseñanza – aprendizaje de programación I



Apéndice A. Consejo Editor Universidad Autónoma de Nayarit

Apéndice B. Consejo Editor ECORFAN

9 786078 534098
ISBN: 978-607-8534-09-8