UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA MEXICALI

Formato para Prácticas de Laboratorio

PROGRAMA EDUCATIVO	PLAN DE ESTUDIO	CLAVE DE UNIDAD DE APRENDIZAJE	NOMBRE DE LA UNIDAD DE APRENDIZAJE
LSC	2009-2	11294	Programación Estructurada

PRÁCTICA No.	LABORATORIO DE	Licenciados en Sistemas Computacionales	DURACIÓN (HORAS)
6	NOMBRE DE LA PRÁCTICA	Funciones Simples	2

1. INTRODUCCIÓN

El lenguaje C está basado en el concepto de bloques. Un bloque es una subrutina que contiene una o más sentencias de C. Cada función tiene un nombre y ejecuta una tarea específica. En general se le puede dar cualquier nombre a las funciones excepto *main()* que es la función principal. Las funciones permiten al programador modularizar un programa.

2. OBJETIVO (COMPETENCIA)

El alumno aplicará las funciones en programas para desarrollarlos de una manera eficientemente estructurada.

3. FUNDAMENTO

Funciones
A cada función del programa se le debe asignar un nombre para poder referirnos a ella cuando deseamos que se ejecute. Si un programa contiene varias funciones, sus definiciones deben aparecer en cualquier orden, pero deben ser independientes unas de otras. Existen dos tipos de funciones:

1. **Funciones de biblioteca.** Estas son parte de C y el usuario no hace más que utilizarlas.
2. **Funciones definidas por el usuario.** Creadas por el programador.

Formuló	Revisó	Autorizó
LSC. Natalia Rodríguez Castellón	I.C. Josefina Mariscal Camacho	Dr. David Isaías Rosas Almeida
Nombre y Firma del Maestro	Nombre y Firma del Responsable de Programa Educativo	Nombre y Firma del Director / Representante de la Dirección

Una función es una sección de código de C que tiene un nombre independiente, que ejecuta una tarea específica y que opcionalmente regresa un valor al programa que la llama.

El uso de funciones definidas por el programador permite dividir un programa grande en un cierto número de componentes más pequeños, cada uno de las cuales con un propósito único e identificable.

Creación de una función

1)Prototipo de Funcion(Declaración)

La declaración de funciones se realiza inmediatamente después de las librerías, ya que son procesos globales de un programa, la declaración de función lleva ; al final.

Existen 3 tipos de funciones

Funciones simples

Void nombre_funcion(void)

Donde *void* significa *nulo*, *nombre_funcion* representa el nombre que tendrá la función y debe ser relacionado con la tarea específica que realiza.

Este tipo de función es la más sencilla, ya que no recibe parámetros, ni retorna valor al programa que la llama, para hacer llamado de este tipo de función basta con escribir el nombre de la función seguido de paréntesis ().

Ejemplo*:* *nombre_funcion();*

2)Definición de la función(Cuerpo de la función)

Es de hecho la función, la definición contiene el código que forma parte de la tarea específica que ejecutara la función, la primera línea de la definición de función es el encabezado, el cual debe ser idéntico a la declaración de la función excepto por el ; (punto y coma), después se agrega el cuerpo de la función el cual es el conjunto de las sentencias que forman parte de la tarea específica.

Funciones que reciben parámetros y no retornan valor

void nombre_funcion(tipo arg1, tipo arg2, ….. tipo argn)

Cuando utilizamos este tipo de funciones es porque se requiere enviar valores a la función la cual los toma en sus parámetros que son variables locales, esto significa que son conocidas solo en la función así como aquellas que son declaradas dentro de la misma. La mayor parte de las funciones tienen una lista de parámetros. Los parámetros proporcionan la forma de comunicar información entre funciones.

Funciones que reciben parámetros y retornan valor

Tipo_retorno nombre_funcion(tipo arg1, tipo arg2, …..tipo argn)

Este tipo de funciones regresan un valor al programa que la llama ya que en ocasiones es necesario hacerlo dependiendo de la situación problemática a resolver.

Las reglas de ámbito de un lenguaje son las reglas que controlan si un fragmento de código conoce o tiene acceso a otro fragmento de código o de datos.

Una *variable local* es aquella cuyo ámbito se restringe a la función que la ha declarado se dice entonces que la variable es local a esa función. Esto implica que esa variable sólo va a poder ser manipulada en dicha sección, y no se podrá hacer referencia fuera de dicha sección. Cualquier variable que se defina dentro de las llaves del cuerpo de una función se interpreta como una variable local a esa función.

Una *variable global* es aquella que se define fuera del cuerpo de cualquier función, normalmente al principio del programa, después de la definición de los archivos de biblioteca (#include), de la definición de constantes simbólicas y antes de cualquier función. El ámbito de una variable global son todas las funciones que componen el programa, cualquier función puede acceder a dichas variables para leer y escribir en ellas. Es decir, se puede hacer referencia a su dirección de memoria en cualquier parte del programa.

Ejemplo de un programa con funciones simples, matrices, utilizando variables globales.

```
#include<stdio.h>
#include<string.h>
int matriz[4][4];
void altas(void);
void cons_gral(void);
void mayor(void);
int main()
{ int op;

do{
      printf("Registrar Matriz.....[1]\n");
      printf("Consulta General.....[2]\n");
      printf("Numero Mayor... .....[3]\n");
      printf("salir.....[4]\n");
      printf("Opcion deseada.....[ ]\b\b");
      scanf("%d",&op);
      switch(op)
      {
            case 1: altas();
                  break;
            case 2: cons_gral();
                  break;
            case 3: mayor();
            break;
            case 4: printf("Adios....");
```

```
                break;
            default: printf("Opcion Invalida");
        }
}while(op!=4);
return 0;
}

void altas(void)
{
        int x, y;

        for(x=0;x<4;x++)
                for(y=0;y<4;y++)
                { printf("Proporcione el valor para la posicion %d,%d:",x,y);
                        scanf("%d",&matriz[x][y]);
                }
}

void cons_gral(void)
{

        int x, y;

        for(x=0;x<4;x++){
                for(y=0;y<4;y++)
                { printf("%d\t",matriz[x][y]);
                }
        printf("\n");
                }
}


void mayor(void)
{ int mayor=0,x,y;

        for(x=0;x<4;x++)
        {
                for(y=0;y<4;y++)
                    {
                      if(matriz[x][y]>mayor)
                        mayor=matriz[x][y];
                    }
        }
                printf("El numero mayor encontrado es %d\n",mayor);

}
```

4.

UNIVERSIDAD AUTÓNOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERÍA MEXICALI

Formato para Prácticas de Laboratorio

A) EQUIPO NECESARIO	MATERIAL DE APOYO
Computadoras con Linux instalado	Práctica impresa y apuntes de clase

B) DESARROLLO DE LA PRÁCTICA

El alumno realizara el programa correspondiente al día de la clase.

Martes

La "Floreria Natz" requiere de un programa que le permita llevar el control de arreglos florales solicitados por sus clientes, el programa deberá mostrar las siguientes opciones:

1) Registrar Cliente(Deberá capturar los siguientes datos: Numero de cliente, Nombre, número de arreglos florales solicitados en el mes)
2) Consulta General (Deberá mostrar los datos de los clientes ordenados por número de cliente)
3) Cliente Distinguido (Mostrar los datos del cliente que tenga mayor número de arreglos florales solicitados en el mes, mostrar un mensaje que se ha ganado un arreglo floral de un precio no mayor a 350.00 pesos)
4) Salir

Jueves

El laboratorio "Mayer", requiere de un programa que le permita llevar el control de los análisis solicitados por sus clientes, el programa deberá mostrar las siguientes opciones:

1) Registrar Solicitud(Deberá capturar los siguientes datos:, numero de recibo, Nombre, tipo de análisis, costo y fecha de entrega)
2) Consulta General (Deberá mostrar los datos de los análisis solicitados ordenados por número de recibo)
3) Análisis mas solicitado (Mostrar los datos de los clientes que coinciden con el tipo de análisis más solicitado)
4) Salir

Viernes

La escuela " El que nada sabe…", requiere de un programa que le permita llevar el control de sus alumnos y calificaciones, el programa deberá mostrar las siguientes opciones:

1) Registrar Alumnos (Deberá capturar los siguientes datos: Numero de control del alumno, nombre y calificaciones para 3 exámenes parciales)
2) Consulta General (Deberá mostrar los datos de los alumnos, ordenados por nombre)
3) Promedios (Mostrar los datos de los alumnos así como el promedio de cada uno y al final mostrar el promedio general)
4) Salir

C) CÁLCULOS (SI APLICA) Y REPORTE

Se revisarán los programas haciendo pruebas 1 o más veces.

5. RESULTADOS Y CONCLUSIONES

El alumno será capaz de elaborar programas estructurados utilizando funciones sencillas.

6. ANEXOS

Las prácticas se realizaran dependiendo del día que corresponda la clase.

7. REFERENCIAS