UNIVERSIDAD AUTONOMA METROPOLITANA

Introducción a la Programación

Práctica 3:

"Funciones, Arreglos, Matrices y Archivos."

Fecha de Elaboración:
Martes, 27 de mayo de 2008

Fecha de Entrega:
Miércoles, 4 de junio de 2008

Contenido

1 Objetivos 2
2 Introducción 2
3 Desarrollo 3
3.1 Actividad 1 "Funciones, paso de parámetros, valor de regreso y el uso de arreglos" 3
3.2 Actividad 2 "Funciones y Matrices" 10
3.3 Actividad 3 "Funciones y cadenas" 10
3.4 Actividad 4 "Funciones y Archivos" 13
4 Resultados 13
4.1 Resultados actividad 1 13
4.2 Resultados actividad 2 13
4.3 Resultados actividad 3 13
4.4 Resultados actividad 4 13
5 Cuestionario 13
6 Conclusiones 15
7 Referencias. 15
8 Actividades Extras 15

AVISOS:

- La práctica puede desarrollarse con a lo más un compañero.
- Por cada día de retardo en la práctica se bajara 2 puntos. No se permite la entrega después de una semana de la fecha límite de la entrega.
- Enviar a la cuenta de correo un paquete zipeado que contendrá: un documento con los resultados de cada una de las actividades, así como los archivos fuentes (con extensión C).
- En el subject del correo escribir: ***IP- practica 3- <nombres completos>***.
- **CBI**: Los compañeros de CBI deben de entregar las actividades extras.

1 Objetivos

1) Utilizar el diseño de programación ascendente y descendente en el enfoque de **programación modular** (enfoque "divide y vencerás") para la resolución de problemas mediante el uso de **módulos** que serán traducidos a funciones (y/o procedimientos) en el lenguaje C.
2) Definir la **declaración e invocación de funciones** dentro de un programa fuente.
3) Utilizar los mecanismos de **paso de parámetros** en las funciones para la resolución de subproblemas dentro de un programa.
4) Utilizar el mecanismo de **valor de regreso** que se define en el uso de funciones para devolver el resultado de alguna operación que se desarrollo dentro de una función.
5) Determinar como declarar, utilizar y enviar como paso de parámetros, **arreglos** dentro de un programa.
6) Determinar como declarar, utilizar y enviar como paso de parámetros, **matrices** dentro de un programa.
7) Determinar como declarar, utilizar y enviar como paso de parámetros, **cadenas** dentro de un programa.
8) Definir el uso de **archivos** para su apertura, lectura y escritura.

2 Introducción

En el desarrollo de esta práctica se efectuara la fase de codificación ingresando en el IDE el código proporcionado para el desarrollo de ejercicios, apoyándose en la carta de estructura y la definición de las funciones.

Ayuda complementaria

- Notas del Curso de Introducción a al Programación en el Capitulo 4, 5 y 6.
- Guía Básica para el uso de C.
- Referencia de la bibliografía dentro de las notas del curso.

3 Desarrollo

La sección del desarrollo de la práctica se divide en una serie de actividades que permiten englobar, desarrollar y alcanzar cada uno de los objetivos de la práctica. Es necesario leer con atención cada una de las actividades a desarrollar.

Configurando el IDE Turbo C – Versión 2

Puede consultar la práctica 1 y 2 en donde se describieron dos mecanismo que permiten configurar el ambiente del uso del IDE Turbo C.

3.1 Actividad 1 "Funciones, paso de parámetros, valor de regreso y el uso de arreglos"

Con el desarrollo de esta actividad se alcanzará los objetivos, 2, 3, y 5 de la práctica.

Funciones

Una función es un conjunto de sentencias (instrucciones) que cumplen un solo propósito bien definido. Pueden recibir una serie de ***parámetros*** que permitan efectuar una ***serie de instrucciones*** y el resultado generado es ***regresado*** por la función en el punto en donde fue ***invocada***. Estas funciones pueden ser invocadas (llamadas) por otras funciones, como por ejemplo la función principal (***main***).

Definición 12: La estructura de **definición** de una **función**:

Pseudocódigo

```
Nombre_funcion ( parametro1, parametro2 )
Comienza
        <inicialización (declaración) de variables locales>
        <instrucciones >
        <instrucciones >
        regresar (  <valor> )
Termina
```

En Código C

```
tipo_retorno Nombre_funcion ( <tipo> parametro1,  <tipo> parametro2)
{
        /*Declaración de variables locales*/
        <tipo> nombre_var;

        /*Instrucciones */

        return  ( <valor>);
}
```

Análisis de la estructura de **función** en ***Lenguaje C***

Concepto	Descripción
tipo_retorno	Tipo de valor devuelto por la función.
Nombre_funcion	Identificador del nombre de función.
(<tipo> parametro1, <tipo> parametro2)	Declaración de la lista de parámetros, cada parámetro es separado por comas, contiene el tipo de parámetro (tipificados) y el identificador de la variable.
{ --- }	Cuerpo de la función
Declaración de variables locales	La declaración de variables **sólo** son 'visibles' dentro de la función.
return (<valor>)	Valor de regreso de la función.

Procedimientos

Los procedimientos son '***como***' funciones pero con la diferencia que ***no regresan*** un resultado específico. Un procedimiento tiene el objetivo principal de desarrollar una serie de instrucciones sin considerar un valor de regreso.

En el Lenguaje C, se hace uso de la **palabra reservada *void***, la cual indica que el procedimiento (función) no regresa valor alguno.

Definición 13: La estructura de **definición** de un **procedimiento**:

Pseudocódigo

```
Nombre_procedimiento( parametro1, parametro2 )
Comienza
        <inicialización (declaración) de variables locales>
        <instrucciones >
        <instrucciones >
Termina
```

En Código C

```
void Nombre_procedimiento ( <tipo> parametro1,  <tipo> parametro2)
{
        /*Declaración de variables locales*/
        <tipo> nombre_var;

        /*Instrucciones */
}
```

Invocación de Procedimiento y Funciones

Primero se describen en pseudocódigo la ***declaración*** de las funciones o procedimientos que se utilizaran y al final se define la estructura del programa principal. En el programa principal es en donde se efectúa la ***invocación*** de las funciones o procedimientos.

El resultado de la invocación de una función, es asignado a una variable, permitiendo así hacer uso del resultado generado dentro del programa principal. En el siguiente tema se explica algunas características de los parámetros.

Definición 14: La estructura de **invocación** de **funciones** y **procedimientos**:

Pseudocódigo

```
Comienza
        valor ← Nombre_funcion ( parametro1, parametro2 )
        Nombre_procedimiento( parametro1, parametro2 )
Termina
```

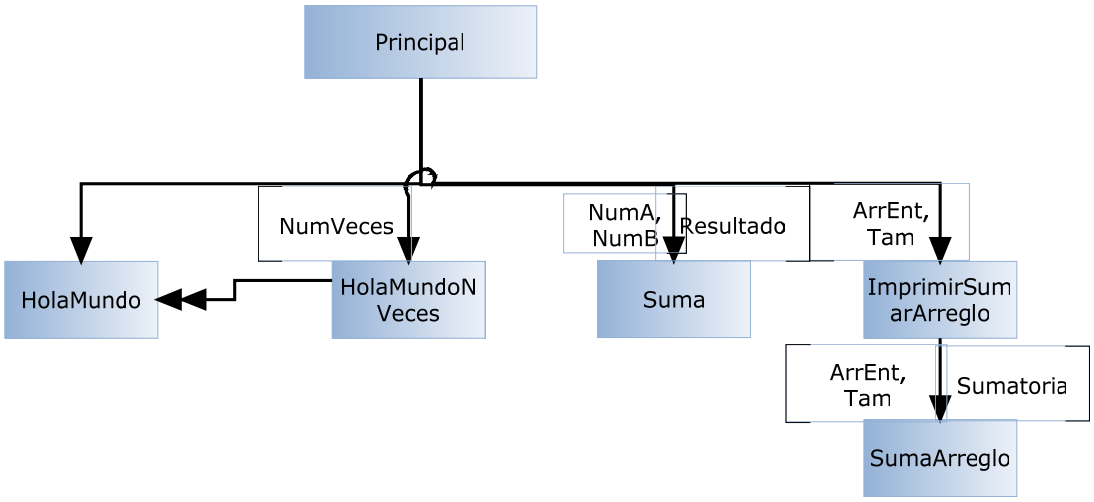
En Código C

```
{
 valor = Nombre_funcion (parametro1,  parametro2);
  Nombre_procedimiento (parametro1,  parametro2);
}
```

Problema

1. "***Funciones***"

   Elabore un programa que refleje la carta de estructura siguiente.



El programa deberá de invocar a las funciones y procedimientos siguientes:

Procedimiento "**HolaMundo**"

Descripción: Se encarga de enviar un mensaje de "hola mundo" a pantalla.
Parámetros: ninguno.
Valor de regreso: ninguno.

Procedimiento "**HolaMundoNVeces**"

Descripción: Se encarga de invocar a la funcion ***HolaMundo*** un número determinado de veces utilizando la estructura de control iterativa **for**, el número de veces que se repetirá será pasado como parámetro a la función.
Parámetros: numero de tipo entero.
Valor de regreso: ninguno.

Función "**Suma**"

Descripción: Dado dos parámetros se efectúa la suma de ambos y el resultado es el valor de regreso de la función.
Parámetros: Dos valores flotantes.
Valor de regreso: Resultado de la operación.

Procedimiento "**ImprimirSumarArreglo**"

Descripción: Se pasa como parámetro un arreglo de enteros y el tamaño del arreglo, la función se encarga de enviar a pantalla el contenido del arreglo. Se invoca a la función "**SumaArreglo**" para obtener la sumatoria y mandar a imprimir el resultado.
Parámetros: Arreglo de enteros, y el tamaño del mismo.
Valor de regreso: Ninguno.

Función "**SumaArreglo**"

Descripción: Se pasa como parámetro un arreglo de enteros y el tamaño del arreglo, la función se encarga de efectuar la suma de los valores contenidos en el arreglo y regresar el resultado.
Parámetros: Arreglo de enteros, y el tamaño del mismo.
Valor de regreso: El resultado de la operación de la sumatoria.

Actividad

Efectué la codificación del siguiente código, complete las secciones de código que hacen falta para las funciones.

Código C

```
/*
  Archivo: 1Funciones.c
  Descripción: Este hace uso de funciones y procedimientos para:
                   Enviar un mensaje a pantalla.
                   Enviar un número determinado de mensajes a pantalla.
                   Efectuar la suma de dos valores y regresar el resultado.
                   Imprimir el contenido de un arreglo e invocar una función
                   que efectué la sumatoria de los valores contenidos dentro
                   del arreglo.
             Objetivo:
              Utilización de Funciones y Procedimiento, paso de parametros,
              valor de regreso y arreglos.
  Autor: Basurto Páez Gustavo
  Fecha: 25 de mayo de 2008
*/
/*Incluir librerías utilizadas en el programas*/
#include <stdio.h>

/**
*Descripción: Procedimiento que manda a imprimir un mensaje de hola mundo
* Entrada : ninguno
* Salida : ninguno
**/
void HolaMundo()
{
    /*Declaración de variables locales*/

    printf(" \n\n\t Inicion del Procedimiento HolaMundo \n");
    printf( " \t\t\"Hola Mundo con Funciones \"  ");
}

/**
*Descripcion: Se encarga de invocar a la funcion HolaMundo un
*              número determinado de veces utilizando la estructura
*              de control iterativa for, el número de veces que
*              se repetirá será pasado como parámetro a la función
* Entrada : numero de tipo entero.
* Salida : ninguno.
**/
void HolaMundoNVeces(int numVeces)
{
    /*Declaración de variables locales*/
    int cont = 0;

    printf(" \n\n\t Inicion del Procedimiento HolaMundoNVeces \n");
    //Inicio del ciclo for, el contador empieza desde 0
    //se vaidad hasta que cont sea menor a numVeces
    for(cont =0 ; cont < numVeces; cont++)
    {
         //SE MANDA A INVOCAR AL PROCEDIMIENTO
         HolaMundo();
    } //fin for
}
```

```
/**
*Descripcion: Dado dos parámetros se efectúa la suma de ambos
*              y el resultado es el valor de regreso de la función
* Entrada : Dos valores flotantes.
* Salida : Resultado de la operación.
**/
float Suma(float NumA, float NumB)
{
    /*Declaración de variables locales*/
    float Resultado = 0.0;

    printf(" \n\n\t Inicion del Procedimiento Suma \n");

    //COMPLETA LA OPERACION DE SUMA


    //SE REGRESA EL VALOR
    return (Resultado);
}

/**
*Descripcion: Se pasa como parámetro un arreglo de enteros y el tamaño
*              del arreglo, la función se encarga de enviar a pantalla
*              el contenido del arreglo.
*              Se invoca a la función "SumaArreglo" para obtener la
*              sumatoria y mandar a imprimir el resultado.
* Entrada : Arreglo de enteros, y el tamaño del mismo.
* Salida : Ninguno.
**/
void ImprimirSumarArreglo(int ArrEnt[], int Tam)
{
    /*Declaración de variables locales*/
    int sumatoria = 0;
    int cont=0;

    printf(" \n\n\t Inicion del Procedimiento ImprimirSumarArreglo \n");

    //COMPLETA LA ESTRUCTURA PARA MANDAR A IMPRIMIR EL CONTENIDO DEL ARREGLO
    for (   ;   ;   )
    {
        printf(" \n Arreglo [%d]= %d ", cont, ArrEnt[cont]);
    }

    //COMPLETE LA INVOCACION DE LA FUNCION SUMAARREGLO

    printf("\n\n El valor de la sumatoria es: %d", sumatoria);

}

/**
*Descripcion: Se pasa como parámetro un arreglo de enteros y el
*              tamaño del arreglo, la función se encarga de efectuar
*              la suma de los valores contenidos en el arreglo y
*              regresar el resultado.
* Entrada : Arreglo de enteros, y el tamaño del mismo.
* Salida : El resultado de la operación de la sumatoria.
**/
int SumaArreglo(int ArrEnt[], int Tam)
{
    /*Declaración de variables locales*/
    int suma = 0;
    int cont = 0;

    printf(" \n\n\t Inicion del Procedimiento SumaArreglo \n");
```

```
    //COMPLETA LA ESTRUCTURA PARA EFECTUAR LA SUMA DEL ARREGLO
    for (   ;   ;    )
    {

    }
    //COMPLETE EL VALOR DE REGRESO

 }

/***FUNCION PRINCIPAL***/
int main()
{
     /*Declaracion (e inicializacion) de variables*/
    int numEntero = 0 ;
    float NumA, NumB;
    float Res = 0.0;
    int arregloEnt []= { 0, 1 , 2, 3, 4, 5, 6, 7, 8, 9 };
    int tamanio =10;

    printf("\t Programa Funciones... \n\n ");

    printf("\n\t Invocacion al procedimiento HolaMundo... \n\n ");
    //INVOCACION AL PROCEDIMIENTO
    HolaMundo();

    printf("\n\nPresione una tecla para continuar\n");
    getch();

    printf("\n\n\n\t Invocacion al procedimiento HolaMundoNVeces... \n\n ");
    printf(" Proporcione un numero entero: " );
    scanf("%d", &numEntero);

    //INVOCACION AL PROCEDIMIENTO
    HolaMundoNVeces(numEntero);

    printf("\n\nPresione una tecla para continuar\n");
    getch();

    printf("\n\n\n\t Invocacion a la funcion Suma... \n\n ");
    printf(" Proporcione un numero flotante: " );
    scanf("%f", &NumA);
    printf(" Proporcione un numero flotante: " );
    scanf("%f", &NumB);
    //INVOCACION DE LA FUNCION
    Res = Suma(NumA, NumB);

    //COMPLETE ENVIANDO EL RESULTADO A PANTALLA

    printf("\n\nPresione una tecla para continuar\n");
    getch();

    printf("\n\n\n\t Invocacion al procedimiento ImprimirSumarArreglo... \n\n
");

    //INVOCACION AL PROCEDIMIENTO
    ImprimirSumarArreglo (arregloEnt , tamanio);

    printf("\n\nPresione una tecla para finalizar el programa \n");
    getch();


    return (0);
}
```

Responda las preguntas 1 y 2del cuestionario.

3.2 Actividad 2 “Funciones y Matrices”

Con el desarrollo de esta actividad se alcanzará los objetivos 1,2, 3, 4 y 6 de la práctica.

Para el desarrollo de esta actividad se puede hacer uso de la seccione **5.2** de las notas del curso, referentes al uso de arreglos multidimencionales**.**

Definición 10: Arreglos bidimensionales

Pseudocódigo

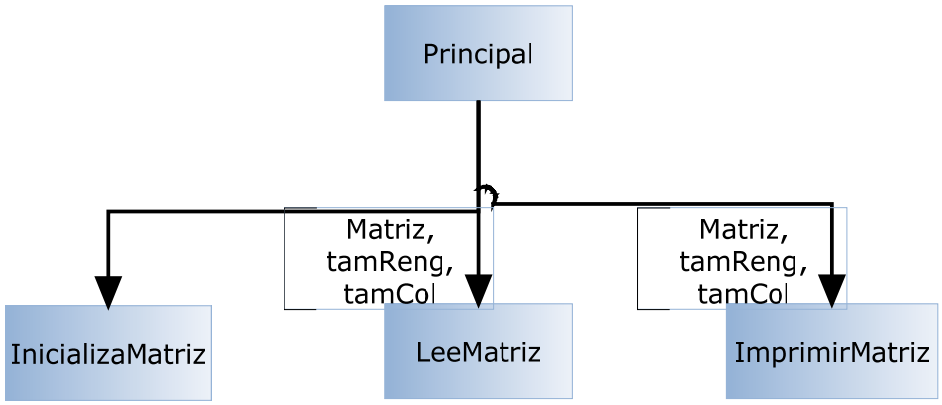
```
//Declaración de una matriz especifico
matrizEntero: { {5,      8,   -1,   0,  10},
                {100, 20,  -10,  -1,  10}   }

//Asignación de valor en el renglón 1 y columna 3
matrizEntero[1][3] ← 25

//Lectura de un valor en la posición de renglón 0 y columna 2
Valor ← matrizEntero [0][2]
```

En Código C

```
//Declaración de una matriz especifico
int matrizEntero [ 2 ][ 5 ] =  { {5,      8,   -1,   0,  10},
                                 {100, 20,  -10,  -1,  10}   } ;

//Asignación de valor en el renglón 1 y columna 3
matrizEntero[1] [3]  =  25

//Lectura de un valor en la posición del renglón 0 y columna 2
Valor = matrizEntero [0][2]
```

Actividad

El programa a desarrollar se descompone modularmente de la siguiente manera.



El programa deberá de invocar a las siguientes funciones y/o procedimientos:

Procedimiento “**InicializaMatriz**”

Descripción: Se encarga de inicializar en ceros el contenido de una matriz.

Parámetros: la matriz, el número de renglones y columnas.

Valor de regreso: por referencia la matriz en ceros.

Procedimiento “LeeMatriz”

Descripción: Se encarga de solicitar al usuario que ingrese elemento a elemento de la matriz, es posible utilizar la estructura de control iterativa **for**, incluyendo el anidamiento en un nivel.
Parámetros: la matriz, el número de renglones y columnas.
Valor de regreso: por referencia la matriz con nuevo valores.

Procedimiento “ImprimirMatriz”

Descripción: Se encarga de enviar a apantalla el contenido de la matriz en un formato adecuado.
Parámetros: la matriz, el número de renglones y columnas.
Valor de regreso: ninguno.

Actividad

Efectué la codificación del siguiente código, complete las secciones de código que hacen falta para las funciones.

Código C

```
/*
  Archivo: 2MatricesyFunciones.c
  Descripción: Este hace uso de funciones y procedimientos para:
                   Incilizar el contenido de una matriz en ceros.
                   Solicitar al usuario que ingrese informacion a la matriz
                   Imprimir el contenido de una matriz.
             Objetivo:
              Utilización de Funciones y Procedimiento, paso de parametros,
              con el uso de matrices.
  Autor: Basurto Páez Gustavo
  Fecha: 25 de mayo de 2008
*/
/*Incluir librerias utilizadas en el programas*/
#include <stdio.h>

//Declaracion de las constantes para determinar el tamanio de la matriz
#define MAXCOLUMNAS  4
#define MAXRENGLONES 3

//Prototipo de funciones.
void InicializaMatriz( int Matriz [MAXRENGLONES][MAXCOLUMNAS], int tamReng,
int tamCol);
void LeeMatriz( int Matriz [MAXRENGLONES][MAXCOLUMNAS], int tamReng, int
tamCol);
void ImprimirMatriz( int Matriz [MAXRENGLONES][MAXCOLUMNAS], int tamReng, int
tamCol);

int main()
{

  int matriz[MAXRENGLONES][MAXCOLUMNAS] ;

  printf("\t Programa Matrices y Funciones... \n\n ");

  printf("\n\t Invocacion al procedimiento InicializaMatriz... \n\n ");
  //COMPLETE LA INVOCACION AL PROCEDIMIENTO
  //Recuerde pasar en orden los parametros.


  printf("\n\nPresione una tecla para continuar\n");
  getch();
```

```
  printf("\n\n\n\t Invocacion al procedimiento LeeMatriz... \n\n ");
  //COMPLETE LA INVOCACION AL PROCEDIMIENTO
  //Recuerde pasar en orden los parametros.

  printf("\n\nPresione una tecla para continuar\n");
  getch();

  printf("\n\n\n\t Invocacion al procedimiento ImprimirMatriz... \n\n ");
  //COMPLETE LA INVOCACION AL PROCEDIMIENTO
  //Recuerde pasar en orden los parametros.

  printf("\n\nPresione una tecla para Finalizar el programa \n");
  getch();

  return (0);
}

/**
*Descripcion: Se pasa como parámetro una matriz, el numero de renglones
              y de columnas, para recorrer la matriz se utiliza dos ciclos
              for, y asi inicializar en cero la matriz.
* Entrada : La matriz, el número de renglones y columnas.
* Salida : Por referencia la matriz en ceros
**/
void InicializaMatriz( int Matriz [MAXRENGLONES][MAXCOLUMNAS], int tamReng,
int tamCol)
{
  /*Declaración de variables locales*/
  int contReng, contCol; //indices para acceder al contenido de la matriz

  printf(" \n\n\t Inicio del Procedimiento InicializaMatriz \n");

  //El primer for es para recorrer los renglones
  for(contReng =0; contReng<tamReng; contReng++ )
  {
        //El segundo for es para recorrer las columnas
        for(contCol =0; contCol<tamCol; contCol++ )
        {
            //Se accede al contenido para colocar el valor 0
            Matriz[contReng][contCol] = 0;
        }
  }

  printf(" \n\n\t Fin del Procedimiento InicializaMatriz \n");
}

/**
*Descripcion: Se encarga de solicitar al usuario que ingrese elemento
*              a elemento de la matriz, es posible utilizar
*              la estructura de control iterativa for, incluyendo
*              el anidamiento en un nivel.
* Entrada : La matriz, el número de renglones y columnas.
* Salida : Por referencia la matriz con nuevo valores
**/
void LeeMatriz( int Matriz [MAXRENGLONES][MAXCOLUMNAS], int tamReng, int
tamCol)
{
  /*Declaración de variables locales*/
  int contReng, contCol; //indices para acceder al contenido de la matriz
  //Se manda a solicitar que el usuario teclee la informacion
  printf(" \n\n\t Inicio del Procedimiento LeeMatriz \n");
  printf(" \n\n Debera de proporcionar los valores a ingresar a la matriz
\n");

  //ACOMPLETAR EL CODIGO
    //El primer for es para recorrer los renglones
  for(      ;       ;     )
  {
```

```
        //El segundo for es para recorrer las columnas
         for(     ;       ;      )
         {
              //se manda a solicitar un valor para almacenarlo
              //en renglon y columna dado
              printf("  \n  Teclee  el  valor  Matriz  [%d][%d]:  ",  contReng,
contCol);
             //COMPLETE LA LECTURA CON LA FUNCION scanf

         }
  }
}


/**
*Descripcion: Se encarga de enviar a apantalla el contenido de la matriz en
*              un formato adecuado.
* Entrada : La matriz, el número de renglones y columnas.
* Salida : Ninguno.
**/
void ImprimirMatriz( int Matriz [MAXRENGLONES][MAXCOLUMNAS], int tamReng, int
tamCol)
{
  /*Declaración de variables locales*/
  int contReng, contCol; //indices para acceder al contenido de la matriz

  printf(" \n\n\t Inicio del Procedimiento ImprimirMatriz \n");
  //Se manda a imprimir el contenido de la matriz
  printf(" \n\n El contenido de la matriz es el siguiente: \n");

  //COMPLETAR EL CODIGO PARA MANDAR A IMPRIMIR EL CONTENIDO DE LA MATRIZ
 //El primer for es para recorrer los englones
  for(  ;    ;    )
  {
         printf("\n\t [ ");

         //El segundo for es para recorrer las columnas
         for(   ;   ;    )
         {
              //se manda a impirmir accediendo al renglon y columna dado

         }
         printf(" ] ");
  }
}
```

Responda las preguntas 3, 4 y 5 del cuestionario.

3.3 Actividad 3 “Funciones y cadenas”

Con el desarrollo de esta actividad se alcanzara el objetivo 7 de la práctica.

Para el desarrollo de esta actividad se puede hacer uso de la seccione **5.3** de las notas del curso, referentes al uso de cadenas**.**

Definición 11: Cadenas

Pseudocódigo
//**Declaración** de una cadena cadena : {‘H’, ‘o’, ‘l’ , ‘a’ , ‘ ‘ , ‘a’ , ‘d’, ‘i’, ‘o’, ‘s’, ‘\0’ }

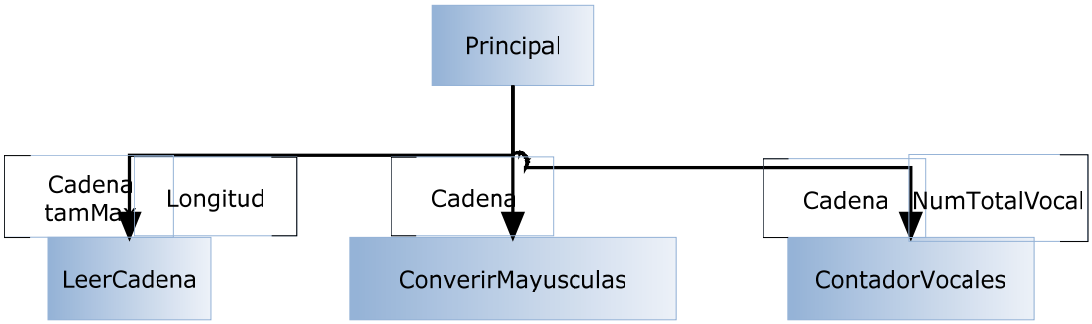
```
//Asignación de un nuevo carácter
cadena  [1] ← 'O'

//Lectura de un carácter de la cadena
Valor ← cadena [0]

//Escribir una cadena
Escribir (cadena )
```

En Código C

```
//Declaración de una cadena
char cadena [ ] =  {'H',  'o', 'l' ,  'a' ,   ' ' ,  'a' , 'd', 'i', 'o', 's', '\0' };
char cadena2 [] = "Hola y Adios\0"

//Asignación de un nuevo carácter en la posición 1 de cadena
cadena [1]   =  'O'

//Lectura de un carácter en la posición 3 de la cadena
valor = cadena  [3]

//Escribir una cadena a pantalla
printf ( "%s", cadena);
```

Problema:

Desarrolle la siguiente carta de estructura que hace uso de funciones y procedimientos para la manipulación de cadenas y caracteres.



El programa deberá de invocar a las siguientes funciones y/o procedimientos:

Función "**LeerCadena**"

Descripción: Se encarga solicitar al usuario que teclee una cadena de caracteres, es posible incluir símbolos como espacios, arrobas, etc. El programa deberá de regresar la longitud de la cadena tecleada.

Parámetros: la cadena como un arreglo de caracteres, el tamaño máximo de caracteres que puede almacenar la cadena.

Valor de regreso: por referencia la cadena con la información tecleada por el usuario.

Procedimiento "**ConverirMayusculas**"

Descripción: Se encarga de recorrer la cadena para determinar si un carácter es letra minúscula, entonces transformarla en mayúscula. ***Sugerencia*** utilizar la información de los caracteres ASCII, considerando que los caracteres también pueden manejarse como enteros.

Parámetros: la cadena de caracteres.
Valor de regreso: Ninguno.

Función "**ContadorVocales**"

Descripción: Se encarga recorrer la cadena de caracteres y determinar si es una vocal, en caso de que sea una vocal se incrementa un contador de vocales mismo que será regresado por la función. ***Sugerencia***: Para determinar si un carácter es vocal utilice la sentencia switch. Considera las vocales minúsculas y mayúsculas.
Parámetros: la cadena de caracteres.
Valor de regreso: número total de vocales encontradas.

Actividad

Efectué la codificación de acuerdo a las especificaciones de las funciones y a la carta de estructura proporcionada. Puede valerse de las notas del curso para determinar el manejo de cadenas.

Responda las preguntas 6 y 7 del cuestionario.

3.4 Actividad 4 "Funciones y Archivos"

Con el desarrollo de esta actividad se alcanzará el objetivo 8 de la práctica.

Definición: Estructura de **declaración** de un tipo de dato Archivo (FILE):

En Pseudocódigo

```
//La declaración de un tipo de dato Archivo no esta definida en Pseudocódigo.

//Apertura de un tipo de dato Archivo.
//   Abrir (  "<ruta y nombre del archivo>")
Abrir("C:\arch.txt")
```

En lenguaje C

```
//Declaración de un tipo de dato Archivo
FILE  *archivo;

//Apertura de una Archivo
//          fopen (nombre_archivo, modo_apertura)
fopen ("C:\\arch.txt", "r");
```

- **Modos de Apertura de un Archivo**

Modo	Significado
"r"	Abre el archivo para ***lectura***.
"w"	Crea y abre un nuevo archivo, si ya existía se ***pierde*** la información.
"a"	Abre el archivo ***existente*** para escribir al **final**.
"r+"	Abre archivo ya existente para **leer o escribir** la información.
"w+"	Crea un archivo para **leer o escribir**, si ya existía se ***pierde*** la información.
"a+"	Abre el archivo ***existente*** para **leer o escribir** al **final**.

Definición: Estructura de **cierre** de un tipo de dato Archivo (FILE):

En Pseudocódigo

```
//Cierre de un tipo de dato Archivo.
//   Cerrar (  <identificador_del_archivo> )
Cerrar(arch)
```

En lenguaje C

```
//Declaración de un tipo de dato Archivo
FILE  *archivo;

//Cierre de una Archivo
//          fclose (identificador_del_archivo)
fclose (archivo);
```

- **Lectura de Información de un Archivo**

Cuando se esta efectuando la lectura de la información de un archivo el compilador requiere de un indicador que le informe que se ha llegado al final del archivo y que ya no hay más datos a leer, este indicador es conocido como al Condición de Fin De Archivo (**End Of File - EOF**).

- **Lectura de caracteres *fgetc()***

Lectura de caracteres.

En Pseudocódigo

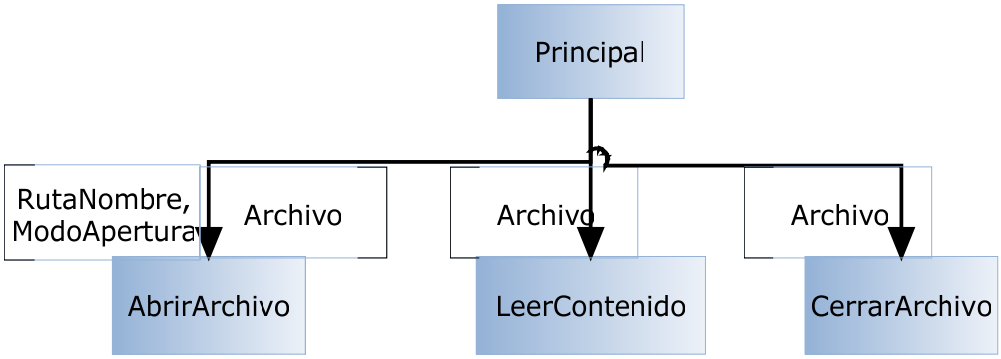
```
caract ← LeeC(archivo)

//lectura continua
caract = LeeC(archivo)
Mientras (  caract ¡= FIN )
Comienza

        caract = LeeC(archivo)
Termina
```

En lenguaje C

```
//Declaración de un tipo de dato Archivo
FILE  *archivo;
char carac;

//Apertura…..

//Lectura continua
carac  = fgetc( archivo);
while ( carac != EOF )
{
        ///Otras instrucciones
         carac  = fgetc( archivo);
}
```

Problema:

Desarrolle la siguiente carta de estructura que hace uso de funciones y procedimientos para la manipulación de archivos.



El programa deberá de invocar a las siguientes funciones y/o procedimientos:

Función "**AbrirArchivo**"

Descripción: Se encarga abrir el archivo solicitado de acuerdo a la cadena RutaNombre, y al modo de apertura. Se tiene que verificar que si se abrió correctamente el archivo. En caso de que no se haya abierto regresar NULL.
Parámetros: RutaNombre que es una cadena que contiene la ruta y nombre (con extensión) del archivo a abrir.
Valor de regreso: La referencia al archivo abierto o bien NULL en caso de que se hayan encontrado problemas al abrir.

Procedimiento "**LeerContenido**"

Descripción: Se encarga de leer el contenido de un archivo y desplegarlo en pantalla..
Parámetros: El descriptor del archivo abierto.
Valor de regreso: Ninguno.

Procedimiento "**CerrarArchivo**"

Descripción: Se encarga cerrar un archivo abierto.
Parámetros: El descriptor del archivo a cerrar.
Valor de regreso: Ninguno.

Actividad

Efectué la codificación de acuerdo a las especificaciones de las funciones y a la carta de estructura proporcionada. Puede valerse de las notas del curso para determinar el manejo de archivo.

Responda las preguntas 8 y 9 del cuestionario.

4 Resultados.

En esta sección se describen los resultados de cada una de las actividades, así como los problemas encontrados durante el desarrollo y el como se resolvieron cada uno de los problemas hasta llegar a completar la actividad.

4.1 Resultados actividad 1

<<En esta seccion se muestra los resultados de la actividad, bien de manera textual o grafica. Muchas veces es mejor considerar la forma grafica >>

4.2 Resultados actividad 2

4.3 Resultados actividad 3

4.4 Resultados actividad 4

Problemas encontrados a realizar la práctica

Id	Descripción	Solución
1	Error de sintaxis al codificar el programa.	Faltaba un punto y coma al finalizar la instrucción.

5 Cuestionario

Responda a las siguientes preguntas de acuerdo al desarrollo de la actividad

1. En la actividad 1, ¿Cómo fue su experiencia con el uso de funciones(y/o procedimientos ? En su respuesta considere la declaración e invocación de funciones, el paso de parámetros y el valor de regreso.

2. En la actividad 1, ¿Cuál fue su experiencia en el uso de arreglo y la estructura de control iterativa for?

3. En la actividad 2, ¿Comente su experiencia con el uso de matrices?

4. En la actividad 2 dentro de la función principal, ¿En que sección de código se debe de modificar para estar seguros que la matriz se inicializa en ceros?

5. En la actividad 2 se desarrollaron algunas funciones con el uso de matrices, ¿Es posible utilizar alguna de estas funciones o procedimientos dentro de otro programa? Justifique su respuesta.

6. En la actividad 3 indique ¿Cuál es la diferencia entre una cadena y un arreglo de caracteres?

7. En el programa desarrollado en la actividad 3, si utiliza la función **scanf("%s", cadena)**, en vez de la función que se definió dentro del programa, ¿Cuál es el inconveniente que presenta?

8. En el programa de la actividad 4 determine que tipos de formatos son adecuados para: la lectura de información, lectura y escritura de información, y escritura de información.
9. En el programa de la actividad 4, si se crea una función para escribir en un archivo, investigue y describa que tipo de funciones son posibles de utilizar.

6 Conclusiones.

Presentar en esta sección las conclusiones de la práctica de acuerdo a los objetivos planteados y a los resultados obtenidos.

7 Referencias.

En caso de consultar algunos documentos, libros o paginas Web como referencia.

8 Actividades Extras

Las siguientes actividades sirven como reforzar y cubrir los objetivos de la práctica.

Ejercicio Extra 1 "*Ordenar en forma creciente un arreglo*"

El programa debe de solicitar que el usuario teclee los valores de un arreglo, posteriormente se efectuaré el ordenamiento de forma creciente del contenido del arreglo. Al finalizar el programa deberá de desplegar el arreglo de manera ordenada. Utilice funciones para esta actividad.

Ejercicio Extra 2 "*Suma de matrices*"

Investigar como se lleva a cabo el proceso de sumar dos matrices y desarrolle un programa que reciba dos matrices y posteriormente efectué la suma de ambas. Al finalizar se debe de desplegar la matriz resultante de la operación.

Ejercicio Extra 3 "*Comparador de cadenas*"

Desarrolle un programa que dada dos cadenas proporcionadas por el usuario, el programa deberá determinar si dichas cadenas son iguales o no, además deberá de desplegar la longitud de cada una de ellas..

Ejercicio Extra 5 "*Escritura en un archivo*"

Reutilice el programa de la actividad 4 y agregue una función que permita escribir información a un archivo.