	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Adrian Ulises Mercado Martínez
Asignatura:	Fundamentos de programación
Grupo:	07
No de Práctica(s):	10
Integrante(s):	Arredondo Granados Gerardo Casillas Herrera Leonardo Didier Diaz Gonzalez Rivas Angel Iñaqui Galván Castro Martha Selene
No. de Equipo de cómputo empleado:	20
No. de Lista o Brigada:	04
Semestre:	2022-1
Fecha de entrega:	25 de Diciembre del 2021
Observaciones:	

CALIFICACIÓN:_____

Práctica 10
Arreglos multidimensionales

Índice

Objetivos

El alumno utilizará arreglos de dos dimensiones en la elaboración de programas que resuelvan problemas que requieran agrupar datos del mismo tipo, en estructuras que utilizan dos índices.

Introducción

En la práctica anterior, se vieron los arreglos de una dimensión, igualmente se mencionó que no sólo existen arreglos de una sola dimensión, en la práctica del día de hoy, veremos arreglos de dos dimensiones. Un arreglo de dos dimensiones, es un conjunto de datos del mismo tipo, esto del mismo tipo es muy importante, ya que no es posible almacenar tipos distintos de datos en los arreglos, con un tamaño ya establecido al momento de crearse. Para lograr acceder al arreglo de dos dimensiones se requiere el uso de dos índices, a diferencia de los arreglos de una dimensión en donde sólo se requería 1 índice.

Estaremos utilizando los arreglos para eficientar el programa, ya que al tener datos de un mismo tipo agrupados en un arreglo de dos dimensiones, nos será de mucha ayuda a lo largo del programa.

Arreglos

Un arreglo es una colección de posiciones de almacenamiento de datos, donde cada una tiene el mismo tipo de dato y el mismo nombre. Cada posición de almacenamiento en un arreglo es llamada un elemento del arreglo.

Arreglos multidimensionales.

El lenguaje C nos permite crear arreglos de n dimensiones, utilizando la sintaxis siguiente:

```
tipoDeDato nombre [tamaño][tamaño]...[tamaño];
```

Nombre se refiere al nombre que el arreglo tendrá, dependiendo para qué se utilizará este arreglo le podemos dar un nombre mucho más específico, lo que lo hace más fácil de identificar si es que trabajaremos con varios arreglos.
TipoDeDato es el tipo de dato que estaremos almacenando en el arreglo, ya sea un entero, un carácter, etc, lo importante es que sólo podremos almacenar un tipo de dato en el arreglo.
Tamaño se refiere al tamaño de cada dimensión del arreglo, como se puede notar en la sintaxis anterior, se hizo uso de puntos suspensivos, esto porque no sabemos de qué tamaño el programador querrá su arreglo, después de definir de cuántas dimensiones será su arreglo, lo siguiente es definir el tamaño de cada una de sus dimensiones.

En esta práctica, el alumno sólo se enfocará en arreglos de dos dimensiones, también conocidos como arreglos bidimensionales.

Apuntadores y su relación con arreglos bidimensionales.

Un apuntador contiene la dirección de una variable, es decir, si nosotros como usuarios queremos acceder a la dirección de memoria de una variable, nosotros haríamos uso de un apuntador, ya que a través de ellos se puede acceder rápidamente a la dirección del dato.
Recordando un poco la sintaxis para declarar un apuntador y asignarle la dirección de memoria de otra variable es:

```
tipoDeDato *apuntador, variable; //Para declarar un apuntador
apuntador = &variable; //Para asignar la dirección de memoria
```

Para declarar un apuntador se inicia con el carácter “*”.
Se hace uso del signo ampersand para indicar que queremos la dirección de alguna variable, como se puede ver en la sintaxis mostrada.

Algo para tomar en cuenta y que es muy importante al momento de usar los apuntadores, es que el apuntador sólo puede apuntar a las direcciones de memoria de la variable con el mismo tipo de dato que el apuntador.
Ya que en los arreglos, los elementos de este se guardan en la memoria de forma continúa, es muy recomendable y muy sencillo acceder a los elementos del arreglo con un apuntador.

Desarrollo

Códigos de arreglos multidimensionales usando for.

Programa 1:

- El programa que se muestra a continuación, genera un arreglo de dos dimensiones (bidimensional) y accede a cada uno de sus elementos, esto gracias a la posición del renglón y columna, usando ciclos for.

```c
#include <stdio.h>
/*
* tipo nombre [tam][tam]...[tam]
*/
int main(){
    int matriz[3][3]={{1,2,3},{4,5,6},{7,8,9}};
    int i,j;
    printf("Imprimir matriz\n");
    for(i=0;i<3;i++){
        for(j=0;j<3;j++){
            printf("%d, ", matriz[i][j]);
        }
    }
    printf("\n");
    return 0;
}
```

Programa 2:

- El programa a continuación al igual que el anterior, genera un arreglo bidimensional y accede a los elementos por la posición dada de su columna y renglón, a diferencia del programa anterior, es que el contenido de los elementos del arreglo es la suma de cada uno de sus índices.

```c
#include <stdio.h>

int main(){
    int i,j,a[5][5];
    for (i = 0; i<5; i++){
        for(j = 0; j<5; j++){
            a[i][j] = i+j;
            printf("\t%d, ", a[i][j]);
        }
        printf("\n");
    }
    return 0;
}
```

Códigos de arreglos multidimensionales usando while

Programa 1b:

- El siguiente programa genera un arreglo bidimensional y accede a cada uno de sus elementos por la posición indicada del renglón y columna, todo esto usando ciclos while.

```c
/**
 * tipo nombre[tam]...[tam];
**/
#include <stdio.h>

int main(){
    int matriz[3][3];
    int i,j;
    printf("Imprimir matriz\n");
    i=0;
    while(i<3){
        j=0;
        while(j<3){
            matriz[i][j]= i+j;
            printf("%d, ", matriz[i][j]);
            j++;
        }
        printf("\n");
        i++;
    }
return 0;
}
```

Códigos de arreglos multidimensionales usando do-while

Programa1.c:

- El siguiente programa genera un arreglo bidimensional y accede a cada uno de sus elementos por la posición indicada del renglón y columna, todo esto usando ciclos while.

```c
/**
 * tipo nombre[tam]...[tam];
**/
#include <stdio.h>

int main(){
    int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
    int i,j;
    printf("Imprimir matriz\n");
    i=0;
    do{
        j=0;
        do{
            printf("%d, ", matriz[i][j]);
            j++;
        }while(j<3);
        printf("\n");
        i++;
    }while(i<3);
return 0;
}
```

Programa 2c:

- El siguiente programa, genera un arreglo bidimensional para acceder a sus elementos por la posición dada por renglón y columna haciendo uso de ciclos do-while, el contenido de los elementos en este arreglo es la suma de cada uno de sus índices.

```c
/**
 * tipo nombre[tam]...[tam];
**/
#include <stdio.h>

int main(){
    int matriz[5][5];
    int i,j;
    printf("Imprimir matriz\n");
    i=0;
    do{
        j=0;
```

```
        do{
            matriz[i][j]=i+j;
            printf("%d, ", matriz[i][j]);
            j++;
        }while(j<5);
        printf("\n");
        i++;
    }while(i<5);
return 0;
}
```

Programa 3:

- El programa siguiente genera un arreglo bidimensional que tiene de tamaño máximo 10 renglones y 10 columnas, se almacenan datos tanto en los renglones y las columnas para después mostrar el contenido del arreglo haciendo uso de ciclos for.

```
#include <stdio.h>
int main(){
    int lista[10][10];
    int i, j;
    int renglon, columna;
    printf("\nDa el número de renglones y columnas separados con
coma\n");
    scanf("%d,%d", &renglon, &columna);
    if (((renglon >= 1) && (renglon <= 10)) && ((columna >= 1) &&
(columna <= 10))){
        for (i = 0; i <= renglon - 1; i++){
            for (j = 0; j <= columna - 1; j++){
                printf("\nNúmero para el elemento %d,%d del arreglo",
i, j);
                scanf("%d", &lista[i][j]);
            }
        }
        printf("\nLos valores dados son: \n");

        for (i = 0; i <= renglon - 1; i++){
            for (j = 0; j <= columna - 1; j++){
                printf("%d ", lista[i][j]);
            }
            printf("\n");
        }
    }
    else
        printf("Los valores dados no es válido");
    printf("\n");
    return 0;
}
```

Códigos de arreglos multidimensionales con apuntadores

Programa 4.c

- El siguiente programa genera un arreglo bidimensional y se accede a sus elementos con la ayuda de un apuntador y un ciclo for.

```c
#include <stdio.h>
int main(){
    int matriz[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int i, cont = 0, *ap;
    ap = *matriz;
    printf("Imprimir Matriz\n");
    for (i = 0; i < 9; i++){
        if (cont == 3){
            printf("\n");
            cont = 0;
        }
        printf("%d\t", *(ap + i));
        cont++;
    }
    printf("\n");
    return 0;
}
```

Programa4b.c

- El siguiente programa genera un arreglo bidimensional y se accede a los elementos del mismo haciendo uso de un apuntador y ciclo while

```c
#include <stdio.h>
int main(){
    int matriz[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int i, cont = 0, *ap;
    ap = *matriz;
    printf("Imprimir Matriz\n");
    i = 0;
    while (i < 9){
        if (cont == 3){
            printf("\n");
            cont = 0;
        }
        printf("%d\t", *(ap + i));
        cont++;
        i++;
    }
    printf("\n");
    return 0;
]
```

Programa4c.c

- El siguiente programa genera un arreglo de dos dimensiones y se accede a sus elementos haciendo uso de un apuntador y el ciclo do-while

```c
#include <stdio.h>
int main(){
    int matriz[3][3] = {{1, 2, 3}, {4, 5, 6}, {7, 8, 9}};
    int i, cont = 0, *ap;
    ap = *matriz;
    printf("Imprimir Matriz\n");
    i = 0;
    do{
        if (cont == 3) {
            printf("\n");
            cont = 0;
        }
        printf("%d\t", *(ap + i));
        cont++;
        i++;
    } while (i < 9);
    printf("\n");
    return 0;
}
```

Conclusiones

Conclusión Grupal

En la realización de esta práctica, se analizaron los arreglos multidimensionales haciendo uso de arreglos bidimensionales. Para acceder a cada elemento, se usaban dos ciclos for, while y do-while, uno anidado dentro de otro. En esta ocasión, los arreglos tenían dos índices, i para los renglones y j para las columnas. Al igual que en la práctica 9, los apuntadores fueron utilizados para almacenar datos del arreglo en una memoria.

Conclusión individual

Arredondo Granados Gerardo: Se cumplieron los objetivos de la práctica pues se hizo uso repetidamente de arreglos de dos dimensiones, se crearon diferentes programas con los diferentes ciclos, for, while, do-while y claro, los arreglos bidimensionales, en los diferentes programas lo que se buscaba era acceder a los diferentes elementos del arreglo, como un extra, al final de la práctica se hizo uso de apuntadores para complementar y agilizar aún más el código.

Casillas Herrera Leonardo Didier: Se utilizaron los arreglos de dos dimensiones y usamos las estructuras cíclicas con while, do-while y for para agrupar los datos del mismo tipo utilizando dos índices en los arreglos; también hicimos uso de los apuntadores para ver los elementos de cada arreglo, con todo lo anterior se cumplió el objetivo de la práctica.

Diaz Gonzalez Rivas Angel Iñaqui: En está práctica, se analizaron y se utilizaron arreglos de dos dimensiones, en donde se podía acceder a cada uno de sus elementos y almacenar datos a través de dos ciclos for, dos ciclos while o dos ciclos do-while. Para almacenar datos, también se usaron los apuntadores que permitieron acceder a cada elemento que estaban organizados en renglones.

Galvan Castro Martha Selene: En conclusión en esta practica se pude decir que se cumplio con los objetivos ya que con programa se estudio y analizo el uso un arreglo bidimensional o de dos dimensiones para el agrupamiento de datos con la ayuda de los ciclos do-while y for, ademas que se vio el uso de apuntadores y como complementan el uso de los arreglos .

Referencias

- El lenguaje de programación C. Brian W. Kernighan, Dennis M. Ritchie, segunda edición, USA, Pearson Educación 1991.