	**Carátula para entrega de prácticas**
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: *Marco Antonio Martínez Quintana.*

Asignatura*: Fundamentos de programación*

Grupo*: 3*

No de Práctica(s): *11*

Integrante(s): *Rosario Vázquez José André*

No. de Equipo de cómputo empleado:

No. de Lista o Brigada: *43*

Semestre: *Primer semestre*

Fecha de entrega*: 7 de enero del 2021*

Observaciones:

CALIFICACIÓN:

Arreglos unidimensionales y multidimensionales.

Objetivo:

Reconocer la importancia y utilidad de los arreglos, en la elaboración de programas que resuelvan problemas que requieran agrupar datos del mismo tipo, así como trabajar con arreglos tanto unidimensionales como multidimensionales.

Objetivos del alumno:

Aprender y comprender el funcionamiento de los arreglos, además como se aplican dentro del lenguaje C, esto para conocer todo los que implica este lenguaje de programación.

Introducción.

Un arreglo es un conjunto de datos contiguos del mismo tipo con un tamaño fijo definido al momento de crearse.

A cada elemento (dato) del arreglo se le asocia una posición particular, el cual se requiere indicar para acceder a un elemento en específico. Esto se logra a través del uso de índices.

Los arreglos pueden ser unidimensionales o multidimensionales. Los arreglos se utilizan para hacer más eficiente el código de un programa.

Actividad.

Comentar los códigos dentro de la práctica.

Arreglo unidimensional while.

```c
/*
    Este programa genera un arreglo unidimensional de 5 elementos y los
    accede a cada elemento del arreglo a través de un ciclo while.
*/

int main (){
    #define TAMANO 5
    int lista[TAMANO] = {10, 8, 5, 8, 7};

    int indice = 0;

    printf("\tLista\n");
    while (indice < 5 ){
        printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
        indice += 1;        // análogo a indice = indice + 1;
    }

    printf("\n");

    return 0;
}
```

Aquí podemos observar el arreglo donde está la definición de una variable donde está el tipo de dato y con el tamaño definido, donde define el número máximo de elementos que puede contener el arreglo.

Código (arreglo unidimensional for)

```c
#include <stdio.h>

/*
    Este programa genera un arreglo unidimensional de 5 elementos y
    accede a cada elemento del arreglo a través de un ciclo for.
*/

int main (){
    #define TAMANO 5
    int lista[TAMANO] = {10, 8, 5, 8, 7};

    printf("\tLista\n");
    for (int indice = 0 ; indice < 5 ; indice++){
        printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
    }

    printf("\n");

    return 0;
}
```

Parecido como el otro código está el nombre de la variable adicionando los valores del tamaño en números enteros que estos son el número máximo de elementos que puede contener el arreglo, para la estructura de repetición.

Código (apuntadores)

```c
#include <stdio.h>

/*
    Este programa crea un apuntador de tipo carácter.
*/

int main () {
    char *ap, c = 'a';
    ap = &c;

    printf("Carácter: %c\n",*ap);
    printf("Código ASCII: %d\n",*ap);
    printf("Dirección de memoria: %d\n",ap);

    return 0;
}
```

En este código podemos observar la variable “ap” que esto es una apuntador, que trabajan para la memoria, ya que a través de ellos se accede con rapidez a un dato. Como vemos posee un ampersand que lo que hace es acceder a la memoria.

Código (apuntadores)

```c
#include<stdio.h>

/*
    Este programa accede a las localidades de memoria de distintas variables a
través de un apuntador.
*/

int main () {
    int a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0};
    int *apEnt;
    apEnt = &a;

    printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n");
    printf("apEnt = &a\n");

    b = *apEnt;
    printf("b = *apEnt \t-> b = %i\n", b);

    b = *apEnt +1;
    printf("b = *apEnt + 1 \t-> b = %i\n", b);

    *apEnt = 0;
    printf("*apEnt = 0 \t-> a = %i\n", a);

    apEnt = &c[0];
    printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt);

    return 0;
}
```

En este código se aplican apuntadores que almacenan la dirección de memoria de la variables a la que apunta; pero también hay un arreglo unidimensional ya que está definida la variable y tiene una tamaño que es el número máximo que puede contener el arreglo.

Código (apuntadores)

```c
#include <stdio.h>

/*
    Este programa trabaja con aritmética de apuntadores para acceder a los
valores de un arreglo.
*/

int main () {
    int arr[] = {5, 4, 3, 2, 1};
    int *apArr;
    apArr = arr;

    printf("int arr[] = {5, 4, 3, 2, 1};\n");
    printf("apArr = &arr[0]\n");

    int x = *apArr;
    printf("x = *apArr \t -> x = %d\n", x);

    x = *(apArr+1);
    printf("x = *(apArr+1) \t -> x = %d\n", x);

    x = *(apArr+2);
    printf("x = *(apArr+1) \t -> x = %d\n", x);

    return 0;
}
```

Como podemos ver el código posee apuntadores y arreglo unidimensional; pero en este caso parece que el apuntador se está sumando con uno de los números máximos del arreglo para la eficiencia del código.

Código (apuntadores en cadena)

```c
#include <stdio.h>

/*
    Este programa muestra el manejo de cadenas en lenguaje C.
*/

int main(){
    char palabra[20];
    int i=0;

    printf("Ingrese una palabra: ");
    scanf("%s", palabra);
    printf("La palabra ingresada es: %s\n", palabra);

    for (i = 0 ; i < 20 ; i++){
        printf("%c\n", palabra[i]);
    }

    return 0;
}
```

En este código el apuntador no está escrito explícitamente, además hay un arreglo unidimensional, donde ambos están participando en la estructura de repetición for para que el funcionamiento de este de eficiente y rapido.

Código (arreglos multidimensionales)

```c
#include<stdio.h>

/*    Este programa genera un arreglo de dos dimensiones (arreglo
multidimensional) y accede a sus elementos a través de dos ciclos
for, uno anidado dentro de otro.
*/

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");
    }
```

```c
    return 0;
}
```

En este código se observa una arreglo multidimensional, ya que está dado por varias dimensiones la cual este caso posee tres dimensiones que está relacionado con los renglones,columnas y al plano, esto para contener el número máximo de elementos que puede contener el arreglo por dimensión, donde este puede tolerar una arreglo: Entero, real de carácter.

Código (arreglos multidimensional con apuntadores)

```c
#include<stdio.h>

/*   Este programa genera un arreglo de dos dimensiones (arreglo
multidimensional) y accede a sus elementos a través de un apuntador utilizando
un ciclo for.
*/

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;
}
```

En este código se tiene arreglo multidimensional específicamente tres dimensiones; pero no solo tiene esto sino que tiene apuntadores que almacenan memoria en la variable, donde se unen en el ciclo for, para un mejor funcionamiento del código.

Conclusión.

En esta práctica analizamos los códigos con arreglos y apuntadores, donde me doy cuenta lo complejas que son, además de lo mucho que abarcan dentro del código, con esto puedo decir que se obtuvo el conocimiento sobre los arreglos.

Referencias.

UNAM. (s. f.). *Laboratorio Salas A y B*. Manual de Prácticas. Recuperado 8 de enero de 2021, de http://lcp02.fi-b.unam.mx/