	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Karina García Morales

Asignatura: Fundamentos de programación

Grupo: 22

No. de práctica(s): Práctica No.9

Integrante(s): Freddy Beckham Cedillo Arias

No. de lista o brigada: No.11

Semestre: 1er Semestre

Fecha de entrega: 15 de octubre del 2024

Observaciones:

CALIFICACIÓN: _____

“Practica 9”

Arreglos unidimensionales

Objetivo:

El alumno utilizará arreglos de una dimensión en la elaboración de programas que resuelvan problemas que requieran agrupar datos del mismo tipo, alineados en un vector o lista.

Desarrollo:

Arreglos:
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. La dimensión del arreglo va de acuerdo con el número de índices que se requiere emplear para acceder a un elemento del arreglo. Así, si se requiere ubicar a un elemento en un arreglo de una dimensión (unidimensional), se requiere de un índice, para un arreglo de dos dimensiones se requieren dos índices y así sucesivamente. Los arreglos se utilizan para hacer más eficiente el código de un programa, así como manipular datos del mismo tipo
con un significado común.

Dar un ejemplo de arreglo unidimensional:

```c
#include <stdio.h>
int main ()
{
    int lista[5] = {10, 8, 5, 8, 7};
    // Se declara e inicializa el arreglo unidimensional
    int indice = 0;
    printf("\tlista\n");
    while (indice < 5 )
    // Acceso a cada elemento del arreglo unidimensional usando while
    {
        printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
        indice += 1; // Sentencia análoga a indice = indice + 1;
    }

    printf("\n");
    return 0;
}
```

Sintaxis del arreglo:
tipoDeDato nombre|tamaño]

¿Qué estructura requieres para manipular un arreglo por medio de sus índices?
La estructura principal que se necesita es un ciclo que use una variable índice que permita acceder a los elementos del arreglo a través de esa variable.

Apuntadores (Dar un ejemplo de declaración y asignación):

Declaración=

```c
#include <stdio.h>
int main ()
{
    char *ap, c = 'a'; // Se declara el apuntador ap de tipo alfanumérico
    ap = &c; //Se le asigna al apuntador la dirección de memoria de la variable c
    printf("Carácter: %c\n",*ap); /* Se imprime el contenido de la variable a la
                                     que apunta el apuntador ap */
    printf("Código ASCII: %d\n",*ap); /*Se imprime el código ASCII del carácter
                                        'a' */
    printf("Dirección de memoria: %d\n",ap);/*Se imprime la dirección de
                                              memoria que almacena el apuntador*/
    return 0;
}
```

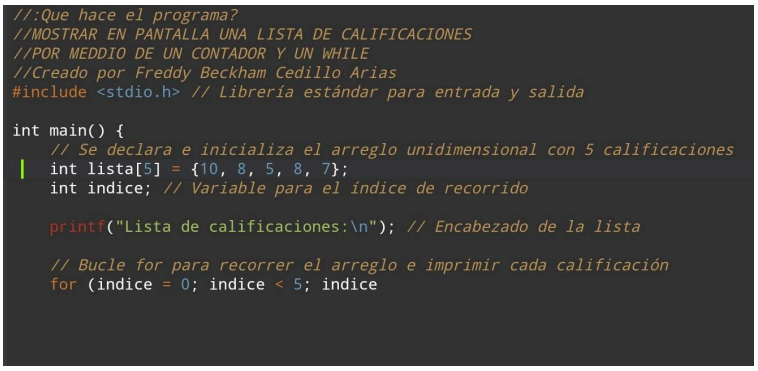
Asignación =

```c
#include <stdio.h>
int main ()
{
    int arr[] = {5, 4, 3, 2, 1};
    int *apArr;//Se declara el apuntador apArr
    int x;
    apArr = arr;
    printf("int arr[] = {5, 4, 3, 2, 1};\n");
    printf("apArr = &arr[0]\n");
    x = *apArr; /*A la variable x se le asigna el contenido del arreglo arr en
                  su elemento 0*/
    printf("x = *apArr \t -> x = %d\n", x);
    x = *(apArr+1); /*A la variable x se le asigna el contenido del arreglo arr
                      en su elemento 1*/
    printf("x = *(apArr+1) \t -> x = %d\n", x);
    x = *(apArr+2); /*A la variable x se le asigna el contenido del arreglo arr
                      en su elemento 2*/
    printf("x = *(apArr+2) \t -> x = %d\n", x);
    x = *(apArr+3); /*A la variable x se le asigna el contenido del arreglo arr
                      en su elemento 3*/
    printf("x = *(apArr+3) \t -> x = %d\n", x);
    x = *(apArr+4); /*A la variable x se le asigna el contenido del arreglo arr
                      en su elemento 4*/
    printf("x = *(apArr+2) \t -> x = %d\n", x);

    return 0;
}
```

Durante la práctica se hicieron los siguientes códigos:
-Este programa crea un arreglo con 5 elementos. Para acceder, recorrer y mostrar cada elemento, se usa una variable índice que va de 0 a 4, utilizando un ciclo while.

```c
// ¿Qué hace el programa?
// Muestra en pantalla una lista de calificaciones usando un contador y un ciclo while.
// Creado por Cedillo Arias Freddy Beckham

#include <stdio.h> // Librería para el uso de entradas y salidas

int main() {
    int lista[5] = {10, 8, 5, 8, 7}; // Se declara e inicializa el arreglo unidimensional
    int indice = 0;

    printf("Lista de calificaciones:\n");

    while (indice < 5) { // Acceso a cada elemento del arreglo unidimensional usando while
        printf("La calificación del alumno %d es %d\n", indice + 1, lista[indice]);
        indice += 1; // Incrementa el índice para continuar el ciclo while
    }

    return 0; // Retorna 0 para indicar que el programa finalizó correctamente
}
```

-Se presenta un código que crea un arreglo de 5 elementos. Para poder recorrer y mostrar cada uno de estos elementos, se utiliza una variable llamada índice, que va desde 0 hasta 4, dentro de un ciclo for. El código demuestra cómo funciona este proceso. Sin embargo, en lugar de usar for, el programa usa un ciclo while para hacer lo mismo.



```
//:Que hace el programa?
//MOSTRAR EN PANTALLA UNA LISTA DE CALIFICACIONES
//POR MEDDIO DE UN CONTADOR Y UN WHILE
//Creado por Freddy Beckham Cedillo Arias
#include <stdio.h> // Libreria estándar para entrada y salida

int main() {
    // Se declara e inicializa el arreglo unidimensional con 5 calificaciones
    int lista[5] = {10, 8, 5, 8, 7};
    int indice; // Variable para el índice de recorrido

    printf("Lista de calificaciones:\n"); // Encabezado de la lista

    // Bucle for para recorrer el arreglo e imprimir cada calificación
    for (indice = 0; indice < 5; indice
```

Tarea:

1.-Indica que realiza el siguiente programa:

```
#include <stdio.h>
int arreglo[] = {16,32,42,67,25,20,73,28,17,45};
int i, j, n, aux;
main() {
n = 10;
for(i=1; i< i++) {
            j = i;
            aux = arreglo[i];
           while(j>0 && aux<arreglo[j-1]){
                        arreglo[j] = arreglo[j-1];
                        j=j-1;
            }
            arreglo[j] = aux;
}
printf("\n\nLos elementos obtenidos del arreglo son: \n");
for(i=0; i< i++) {
            printf("Elemento [%d]: %d\n", i, arreglo[i]);
}
return 0;
}
```

R= Descripción del algoritmo:
El programa implementa el algoritmo de ordenamiento por inserción.
El algoritmo comienza desde el segundo elemento (índice 1) y lo compara con los elementos anteriores para colocarlo en su posición correcta. Este proceso se repite para todos los elementos del arreglo.
Después de la ordenación, el programa imprime los elementos del arreglo en orden ascendente.

2.- Genera un programa que lea solicite una cadena de letras y números al usuario(emplear arreglo) e imprima solo letras. Este arreglo debe ser de caracteres, recuerda que un caracter puede almacenar números pero un arreglo de números no puede almacenar caracteres.

Análisis:
DE: Imprimir solo las letras de un arreglo con una combinación de letras y números.
RE: Imprimir solo letras sin números, excluir(0,1,2,3,4,5,6,7,8,9).
DS: Obtener la impresión de las letras.
Ejemplo:
Entrada de usuario= Ej3mpl0c4d3n4
Salida en pantalla= Ejmplcdn

```c
#include <stdio.h>
#include <ctype.h> // Para usar la función isalpha()

int main() {
    char cadena[100];
    int i;

    // Solicitar entrada del usuario
    printf("Ingresa una cadena de letras y números: ");
    scanf("%s", cadena);

    printf("Solo las letras en la cadena son: ");

    // Recorrer la cadena y verificar si el carácter es una letra
    for (i = 0; cadena[i] != '\0'; i++) {
        if (isalpha(cadena[i])) {
            printf("%c", cadena[i]);
        }
    }

    printf("\n");
    return 0;
}
```

```
Ingresa una cadena de letras y números: xhfdtuf6ige56gyud76to7g7v
Solo las letras en la cadena son: xhfdtufigegyudtogv


...Program finished with exit code 0
Press ENTER to exit console.
```

3.- Escribe un ejemplo de un arreglo tipo estructura

```c
#include <stdio.h>
#include <string.h>

struct Persona {
    char nombre[50];
    int edad;
};

int main() {
    struct Persona personas[3];

    // Asignación de valores a las estructuras
    strcpy(personas[0].nombre, "Juan");
    personas[0].edad = 25;

    strcpy(personas[1].nombre, "Ana");
    personas[1].edad = 30;

    strcpy(personas[2].nombre, "Luis");
    personas[2].edad = 28;

    // Imprimir información de las personas
    for (int i = 0; i < 3; i++) {
        printf("Nombre: %s, Edad: %d\n", personas[i].nombre, personas[i].edad);
    }

    return 0;
}
```

```
Nombre: Juan, Edad: 25
Nombre: Ana, Edad: 30
Nombre: Luis, Edad: 28


...Program finished with exit code 0
Press ENTER to exit console.
```

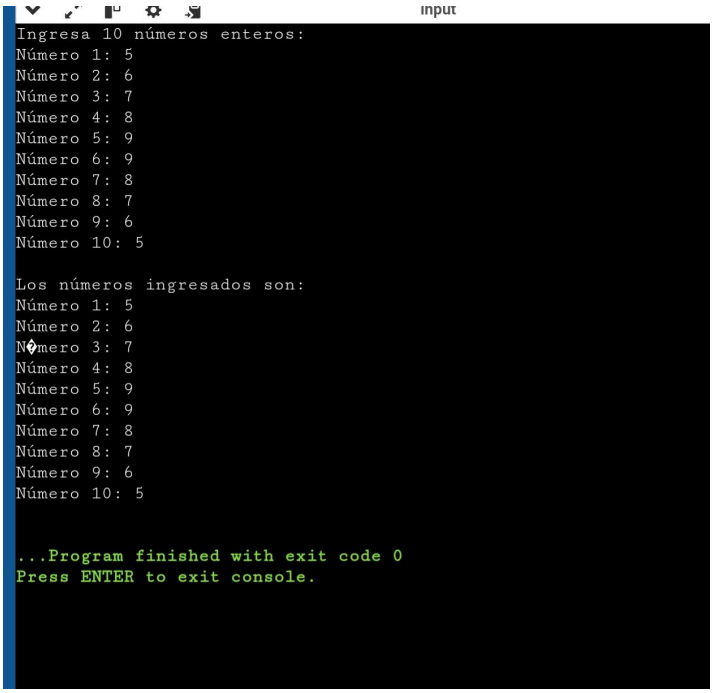
4.- Genera un programa que solicite al usuario un vector de 10 enteros haciendo uso de un arreglo y la estructura iterativa para recorrer por medio de sus índices e imprimir en pantalla.

```c
#include <stdio.h>

int main() {
    int numeros[10];
    int i;

    // Solicitar los 10 números al usuario
    printf("Ingresa 10 números enteros:\n");
    for (i = 0; i < 10; i++) {
        printf("Número %d: ", i + 1);
        scanf("%d", &numeros[i]);
    }

    // Imprimir los valores ingresados
    printf("\nLos números ingresados son:\n");
    for (i = 0; i < 10; i++) {
        printf("Número %d: %d\n", i + 1, numeros[i]);
    }

    return 0;
}
```

```
Ingresa 10 números enteros:
Número 1: 6
Número 2: 7
Número 3: 8
Número 4: 9
Número 5: 4
Número 6: 7
Número 7: 8
Número 8: 5
Número 9: 3
Número 10: 9

Los números ingresados son:
Número 1: 6
Número 2: 7
Número 3: 8
Número 4: 9
Número 5: 4
Número 6: 7
Número 7: 8
Número 8: 5
Número 9: 3
Número 10: 9


...Program finished with exit code 0
Press ENTER to exit console.
```

5.- Modifica el programa del ejercicio 4 y aplica el uso de apuntadores.

```c
#include <stdio.h>

int main() {
    int numeros[10];
    int *ptr;
    int i;

    // Solicitar los 10 números al usuario
    printf("Ingresa 10 números enteros:\n");
    for (i = 0; i < 10; i++) {
        printf("Número %d: ", i + 1);
        scanf("%d", &numeros[i]);
    }

    // Usamos el apuntador para recorrer el arreglo
    ptr = numeros;  // Apuntar al primer elemento del arreglo

    printf("\nLos números ingresados son:\n");
    for (i = 0; i < 10; i++) {
        printf("Número %d: %d\n", i + 1, *(ptr + i));
    }

    return 0;
}
```



Conclusión:

Los apuntadores son variables que nos guardan direcciones de memoria mientras que los arreglos unidimensionales son grupos de elementos organizados en una secuencia, y su nombre actúa como un apuntador al primer elemento. Esto nos permitirá trabajar en un futuro con arreglos de manera eficaz y con mayorfluidez, pero hay que tener cuidado, ya que un uso incorrecto de los apuntadores puede causar grandes errores que podrían ser altamente difíciles de corregir.

Bibliografía:

Para esta Práctica no se uso ninguna información web