	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): Ramírez Pérez Daniela Itzel

No. de Equipo de cómputo empleado: No aplica

No. de Lista o Brigada: 36

Semestre: 1er

Fecha de entrega: Viernes 8 de Enero

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.

Introducción

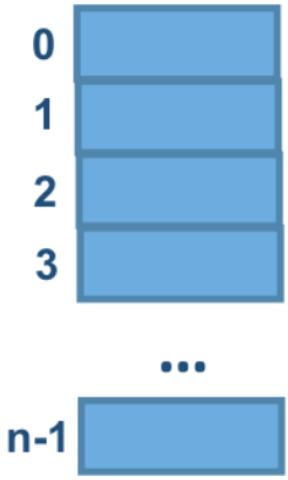
Un arreglo es un conjunto de datos contiguos del mismo tipo con un tamaño fijo definido al momento de crearse. Podemos acceder a ellos usando un índice.

Los arreglos pueden ser:

- Unidimensionales
- Multidimensionales

Arreglos unidimensionales

Un arreglo unidimensional de n elementos en la memoria se almacena de la siguiente manera:



La primera localidad del arreglo corresponde al índice 0 y la última corresponde al índice n-1, donde n es el tamaño del arreglo.

```
tipoDeDato nombre[tamaño]
```

- **nombre** se refiere al identificador del arreglo
- **tamaño** es un número entero y define el número máximo de elementos que puede contener el arreglo.
- Un **arreglo** puede ser de los tipos de dato entero, real, carácter o estructura.

Código (arreglo unidimensional while)

```c
#include <stdio.h>
/*
 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;
}
```

```
        Lista

Calificación del alumno 1 es 10
Calificación del alumno 2 es 8
Calificación del alumno 3 es 5
Calificación del alumno 4 es 8
Calificación del alumno 5 es 7
```

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;
}
```

```
        Lista

Calificación del alumno 1 es 10
Calificación del alumno 2 es 8
Calificación del alumno 3 es 5
Calificación del alumno 4 es 8
Calificación del alumno 5 es 7
```

Apuntadores

Un apuntador es una variable que contiene la localización de la variable pedida

La sintaxis para declarar un apuntador y para asignarle la dirección de memoria de otra variable es, respectivamente:

```
TipoDeDato *apuntador, variable;

apuntador = &variable;
```

La declaración de una variable apuntador inicia con el carácter *. Cuando a una variable le antecede un ampersand, lo que se hace es acceder a la dirección de memoria de la misma (es lo que pasa cuando se lee un dato con scanf). con el que fueron declarados; para acceder al contenido de dicha dirección, a la variable apuntador se le antepone *.

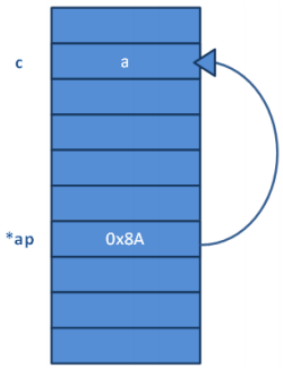
Código (apuntadores)

```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;
}
```

```
Carácter: a
Código ASCII: 97
Dirección de memoria: 6422299
```



Un apuntador almacena la dirección de memoria de la variable a la que apunta

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;
}
```

```
a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}
apEnt = &a
b = *apEnt      -> b = 5
b = *apEnt + 1  -> b = 6
*apEnt = 0      -> a = 0
apEnt = &c[0]   -> apEnt = 5
```

Cabe mencionar que el nombre de un arreglo es un apuntador fijo al primero de sus elementos; por lo que las siguientes instrucciones, para el código de arriba, son equivalentes:

```
apEnt = &c[0];
apEnt = c;
```

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;
}
```

```
int arr[] = {5, 4, 3, 2, 1};
apArr = &arr[0]
x = *apArr          -> x = 5
x = *(apArr+1)      -> x = 4
x = *(apArr+1)      -> x = 3
```

Código (apuntadores en ciclo 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 apuntador
utilizando un ciclo for.
*/
int main (){
 #define TAMANO 5
 int lista[TAMANO] = {10, 8, 5, 8, 7};
 int *ap = lista; for (int indice = 0 ; indice < 5 ; indice++){
 printf("\nCalificaci%cn del alumno %d es %d",162, indice+1, *(ap+indice));
 }

 printf("\n");

 return 0;
}
```

```
Calificación del alumno 1 es 10
Calificación del alumno 2 es 8
Calificación del alumno 3 es 5
Calificación del alumno 4 es 8
Calificación del alumno 5 es 7
```

Código (apuntadores en cadenas)

```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;
}
```

```
Ingrese una palabra: Hola
La palabra ingresada es: Hola
H
o
l
a

↓
@

P
↓
@









P
>
```

Arreglos multidimensionales

Lenguaje C permite crear arreglos de varias dimensiones con la siguiente sintaxis:

```
tipoDato nombre[ tamaño ][ tamaño ]…[tamaño];
```

La diferencia que se encuentra es en el aumento en el número de corchetes (sus dimensiones)

De manera práctica se puede considerar que la primera dimensión corresponde a los renglones, la segunda a las columnas, la tercera al plano, y así sucesivamente. Sin embargo, en la memoria cada elemento del arreglo se guarda de forma contigua, por lo tanto, se puede recorrer un arreglo multidimensional con apuntadores.

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");
 }
return 0;
}
```

```
Imprimir Matriz
1, 2, 3,
4, 5, 6,
7, 8, 9,
```

Código (arreglos multidimensionales con apuntadores)

```
#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;
}
```

Conclusión

Gracias a los arreglos es mucho más fácil poder acceder y utilizar las cantidades caracteres que ingresamos al saber que estas mismas son guardadas en un lugar especifico sin tener la angustia que se pierda, aunque se empieza a complicar estos arreglos la practica logro su objetivo de darnos paso a paso el funcionamiento de estos mismos.

Referencias

- Nakayama Cervantes, A., Castañeda Perdomo, M., Solano Gálvez, J. A., García Cano, E. E., Sandoval Montaño, L., & Arteaga Ricci, T. I. (2018, 6 abril). Manual de prácticas del laboratorio de Fundamentos de programación (p.179-190). Laboratorio de Computación Salas A y B. http://odin.fib.unam.mx/salac/practicasFP/MADO17_FP.pdf