	Carátula para entrega de prácticas
Facultad de Ingeniería	Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor:	Marco Antonio Martinez Quintana
Asignatura:	Fundamentos de Programación
Grupo:	03
No de Práctica(s):	Practica N°11
Integrante(s):	Torres Gracian Christian Ivan
No. de Equipo de cómputo empleado:	No Aplica
No. de Lista o Brigada:	55
Semestre:	Semestre N°1
Fecha de entrega:	
Observaciones:	

CALIFICACIÓN: _____

Guía práctica de estudio 11: 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.

Actividades:

- Elaborar un programa en lenguaje C que emplee arreglos de una dimensión.
- Resolver un problema que requiera el uso de un arreglo de dos dimensiones, a través de un programa en lenguaje C.
- Manipular arreglos a través de índices y apuntadores.

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.

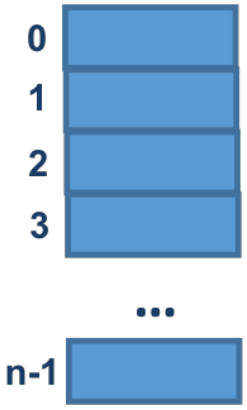
Licencia GPL de GNU

El software presente en esta práctica es libre bajo la licencia GPL de GNU, es decir, se puede modificar y distribuir mientras se mantenga la licencia GPL.

```
/*
 *
 * This program is free software: you can redistribute it and/or modify
 * it under the terms of the GNU General Public License as published by
 * the Free Software Foundation, either version 3 of the License, or
 * (at your option) any later version.
 *
 * This program is distributed in the hope that it will be useful,
 * but WITHOUT ANY WARRANTY; without even the implied warranty of
 * MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the
 * GNU General Public License for more details.
 *
 * You should have received a copy of the GNU General Public License
 * along with this program. If not, see <http://www.gnu.org/licenses/>.
 *
 * Author: Jorge A. Solano
 *
*/
```

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.

La sintaxis para definir un arreglo en lenguaje C es la siguiente:

tipoDeDato nombre[tamaño]

Donde 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.

NOTA: Los tipos de datos estructuras no se abordarán en esta práctica.

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;
}
```

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;
}
```

Apuntadores

Un apuntador es una variable que contiene la dirección de una variable, es decir, hace referencia a la localidad de memoria de otra variable. Debido a que los apuntadores trabajan directamente con la memoria, a través de ellos se accede con rapidez a un dato.

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).

Los apuntadores solo pueden apuntar a direcciones de memoria del mismo tipo de dato con el que fueron declarados; para acceder al contenido de dicha dirección, a la variable apuntador se le antepone *.

Código (apuntadores)

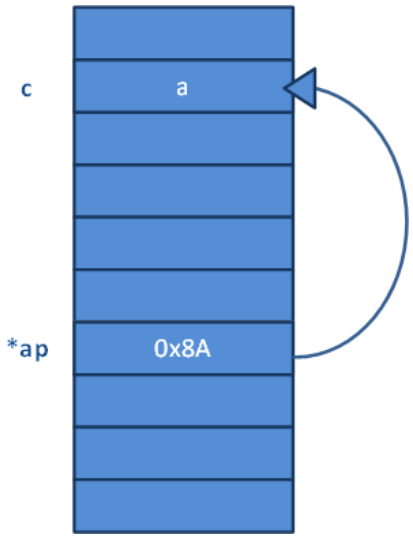
```
#include <stdio.h>

/*
    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;
}
```



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

Código (apuntadores)

```
#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[O] \t-> apEnt = %i\n", *apEnt);

     return 0;
}
```

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)

```
#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[O]\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;
}
```

Código (apuntadores en ciclo for)

```
#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;
```

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

    printf("\n");

    return 0;
}
```

Código (apuntadores en cadenas)

```
#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;
}
```

Arreglos multidimensionales

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

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

Donde nombre se refiere al identificador del arreglo, tamaño es un número entero y define el número máximo de elementos que puede contener el arreglo por dimensión (el número de dimensiones está determinado por el número de corchetes). Los tipos de dato que puede tolerar un arreglo multidimensional son: entero, real, carácter o estructura.

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;
}
```

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;
}
```

Resultados

Bibliografía

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