



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: ALEJANDRO ESTEBAN PIMENTEL ALARCÓN

Asignatura: FUNDAMENTOS DE PROGRAMACIÓN

Grupo: 135

No de Práctica(s): 11

Integrante(s): ALITZEL ANAID GUTIÉRREZ RAMOS
LAURA MUÑOZ REYES

*No. de Equipo de
cómputo empleado:* SOMALIA 42

No. de Lista o 9370
2823

Semestre: 2020 - 1

Fecha de entrega: 28/10/2019

Observaciones: Muy bien

CALIFICACIÓN: 10

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. 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 y se utilizan para hacer más eficiente el código de un programa C.

Arreglos unidimensionales



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] 0.

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.

Arreglos unidimensionales

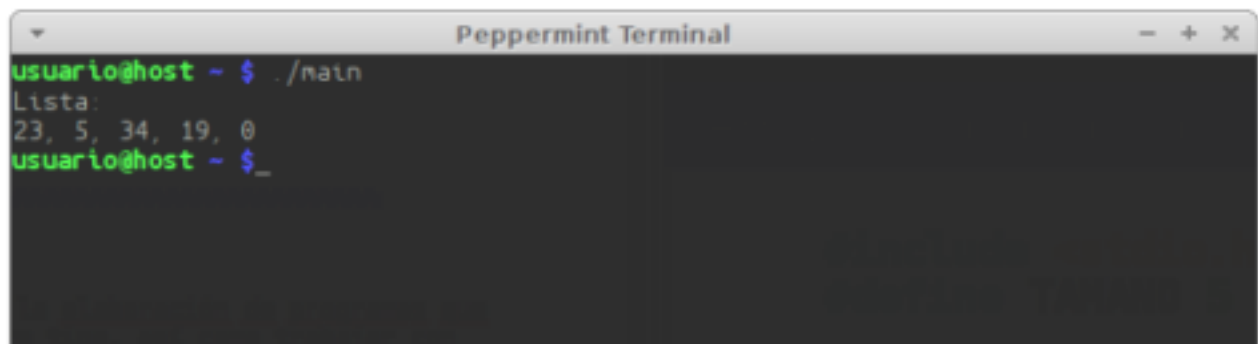
```
#include <stdio.h>
#define TAMANO 5

int main(int argc, char *argv[])
{
    int lista[TAMANO] = {23, 5, 34, 19, 0};

    printf("Lista:\n");
    for(int i=0; i< TAMANO-1; i++){
        printf("%i, ", lista[i]);
    }

    printf("%i\n", lista[TAMANO-1]);

    return 0;
}
```



The screenshot shows a terminal window titled "Peppermint Terminal". The user runs the command `./main` at the prompt `usuario@host ~`. The program outputs "Lista:" followed by the array elements "23, 5, 34, 19, 0" on the next line. The prompt returns to `usuario@host ~` with a trailing underscore.

```
Peppermint Terminal
usuario@host ~ $ ./main
Lista:
23, 5, 34, 19, 0
usuario@host ~ $ _
```

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.

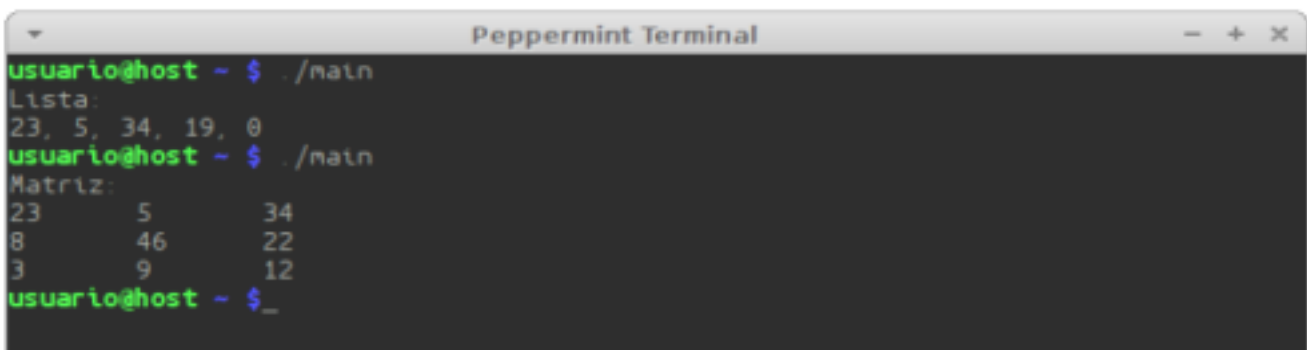
Arreglos multidimensionales

```
#include <stdio.h>
#define DIM 3

int main(int argc, char *argv[])
{
    int matriz[DIM][DIM] = {{23, 5, 34},
                             { 8, 46, 22},
                             { 3, 9, 12}};

    printf("Matriz:\n");
    for(int i=0; i < DIM; i++){
        for(int j=0; j < DIM ; j++){
            printf("%i\t",matriz[i][j]);
        }
        printf("\n");
    }

    return 0;
}
```



Peppermint Terminal

```
usuario@host ~ $ ./matn
Lista:
23, 5, 34, 19, 0
usuario@host ~ $ ./matn
Matriz:
23      5      34
8       46     22
3       9      12
usuario@host ~ $ _
```

Actividad 1

Hacer un programa que:

- ✓ Pida al usuario un numero.
- ✓ Genere un arreglo de esa longitud.
- ✓ Pida al usuario numeros suficientes para llenar el arreglo.
- ✓ Muestre al usuario el numero menor y el mayor de dicho arreglo.

En este programa con el primer for llenamos la lista. Luego declaramos nuestra variable menor. Por consiguiente, le asignamos un valor a menor, suponiendo que el primer elemento de la lista es el numero menor. El segundo for es para obtener el numero menor.

Y ahora con el if vamos a comparar el siguiente elemento de la lista (porque $i=1$) con el numero menor relativo que fue el primero (que la primera vez es `lista[0]`). Una vez que ya realizamos eso si el siguiente elemento es menor éste pasará a ser el nuevo número menor relativo. Y seguirá así hasta que se comparen todos los elementos de la lista. Por ultimo mostramos en pantalla el resultado del numero mayor y menor obtenidos.

Ya que realizamos todo lo anterior y comprobamos que funcionara, éstos fueron nuestros resultados:

File Edit Selection Find View Goto Tools Project Preferences



tablas.c

promedio.c

1.c

```
1  #include<stdio.h>
2
3  int main(){
4      int t;
5      printf("Ingresa el tamano de la lista\n");
6      scanf("%i",&t);
7
8      int lista[t];
9
10     for(int a=0;a<t;a++){
11
12         printf("lista[%i]=\n",a+1);gcc
13         scanf("%i",&lista[a]);
14     }
15
16     int menor;
17     menor=lista[0];
18
19     for(int a=0;a<t;a++){
20         if (lista[a]<menor){
21             menor=lista[a];
22         }
23     }
24
25     int mayor;
26     mayor=lista[0];
27
28     for(int a=0;a<t;a++){
29         if (lista[a]>mayor){
30             mayor=lista[a];
31         }
32     }
33     printf("El numero menor es %i\n",menor);
34     printf("El numero mayor es %i\n",mayor);
35
36     return 0;
```

```
vanessa@Titan:~/Escritorio$ gcc 1.c -o 1
```

```
vanessa@Titan:~/Escritorio$ ./1
```

```
Ingresa el tamaño de la lista
```

```
3
```

```
lista[1]=
```

```
1
```

```
lista[2]=
```

```
2
```

```
lista[3]=
```

```
3
```

```
El número menor es 1
```

```
El número mayor es 3
```

```
vanessa@Titan:~/Escritorio$ ./1
```

```
$ Ingresa el tamaño de la lista
```

```
6
```

```
lista[1]=
```

```
5
```

```
lista[2]=
```

```
9
```

```
lista[3]=
```

```
7
```

```
lista[4]=
```

```
3
```

```
lista[5]=
```

```
1
```

```
lista[6]=
```

```
4
```

```
El número menor es 1
```

```
El número mayor es 9
```

```
vanessa@Titan:~/Escritorio$ 
```

Actividad 2

Hacer un programa que:

- ✓ Pida al usuario dos números N y M.
- ✓ Genere dos matrices de N x M.
- ✓ Pida al usuario números suficientes para llenar ambas matrices.
- ✓ Muestre al usuario la matriz resultado de sumar las dos de entrada.
- ✓

En este programa lo que hacemos de inicio, es declarar las variables de matrices: 1, 2 y resultado, luego le indicamos al usuario que ingrese los números para rellenar el arreglo numero 1 y eso es posible gracias al primer for el cual tiene dentro otro for que de igual manera nos ayudará a rellenar el segundo arreglo y el tercer for tiene como función sumar e imprimir el resultado de la suma.

Una vez que realizamos los detalles para que el programa corriera, estos fueron nuestros resultados:


```
#include<stdio.h>

int main(){
    int x,y;
    printf("Ingrese columnas de la matriz\n");
    scanf("%i",&x);
    printf("Ingrese filas de la matriz\n");
    scanf("%i",&y);

    int m1[x][y];
    int m2[x][y];
    int m3[x][y];
    printf("Llenar la Matriz 1\n");
    printf("\n");
    for(int a=0;a<x;a++){
        for (int b=0;b<y;b++){
            printf("lugar[%i][%i]\n",a+1,b+1);
            scanf("%i",&m1[a][b]);
        }
    }
    printf("\n");

    printf("Llenar la Matriz 2\n");

    for (int a=0;a<x;a++){
        for (int b=0;b<y;b++){
            printf("lugar[%i][%i]\n",a+1,b+1);
            scanf("%i",&m2[a][b]);
        }
    }
```

```
34
35
36     }
37     printf("\n");
38
39
40     printf("La suma de las dos matrices es\n");
41
42
43
44     for (int a=0;a<x;a++){
45
46         for (int b=0;b<y;b++){
47             m3[a][b]= m1[a][b]+ m2[a][b];
48             printf("%i\t",m3[a][b]);
49         }
50
51         printf("\n");
52
53     }
54
55     return 0;
56 }
```

```
Archivo  Editor  Ver  Buscar  Terminal  Ayuda
-----
vanessa@Titan:~/Escritorio$ ./a2
Ingrese columnas de la matriz
2
Ingrese filas de la matriz
3
Llenar la Matriz 1
lugar[1][1]
1
lugar[1][2]
2
lugar[1][3]
3
lugar[2][1]
6
lugar[2][2]
5
lugar[2][3]
48

Llenar la Matriz 2
lugar[1][1]
7
lugar[1][2]
9
lugar[1][3]
8
lugar[2][1]
68
lugar[2][2]
8
lugar[2][3]
8

La suma de las dos matrices es
8      11      11
74     13      56
```

Conclusión

Los arreglos unidimensionales son importantes para el almacenamiento de datos, e incluso de grandes listas, nos permiten realizar operaciones dentro de ellas lo cual es muy útil, desde una lista de calificaciones hacer un promedio hasta grandes bases de datos, el arreglo unidimensional es exclusivo de listas y es muy útil.

Los arreglos multidimensionales nos permiten hacer operaciones complejas como lo podrían ser las matrices en las matemáticas, lo increíble de estos programas es que se pueden editar para n caso, es decir, son flexibles dependiendo del problema que necesite resolver el usuario, tanto para los arreglos unidimensionales como para los multidimensionales.