



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): Emiliano Guevara Chávez

No. de Equipo de

Cómputo empleado: No aplica

No. De Lista o Brigada: 20

Semestre: Ciclo 2021-1

Fecha de entrega: 04/01/2021

Observaciones: _____

CALIFICACIÓN: _____

Práctica #11:

Arreglos unidimensionales y multidimensionales

Objetivo:

Reconocer la importancia y utilidad de los arreglos, en la elaboración de programas que resuelven problemas que requieren agrupar datos del mismo tiempo, 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. Los arreglos se utilizan para hacer más eficiente el código de un programa.

Desarrollo de las actividades:

Arreglos unidimensionales

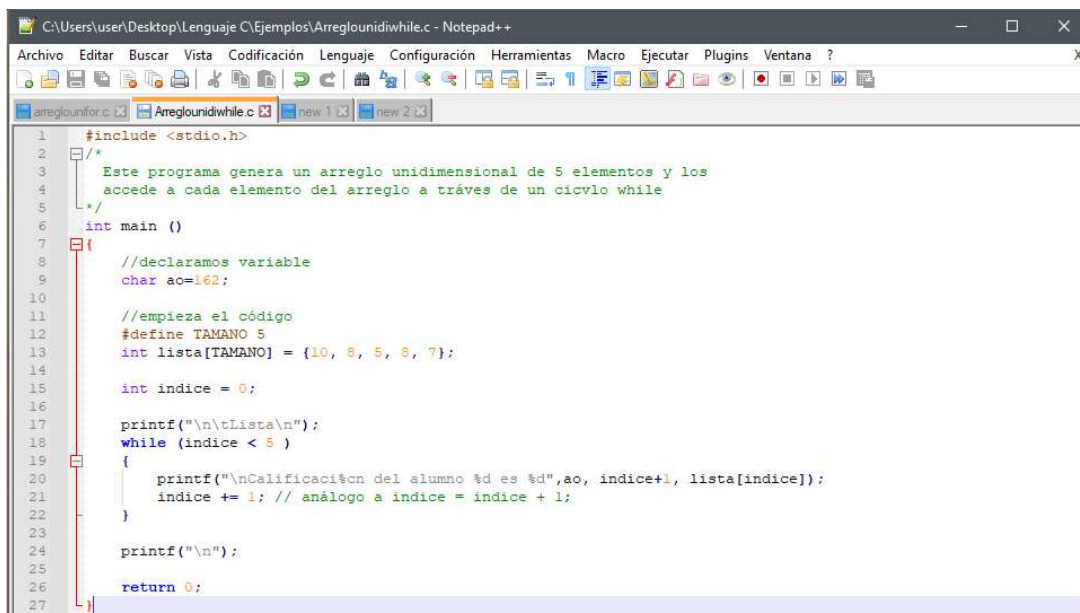
Un arreglo unidimensional de n elementos en la memoria se almacena de la siguiente manera:

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]

Código (arreglo unidimensional while)



```
1 #include <stdio.h>
2
3 /*
4  Este programa genera un arreglo unidimensional de 5 elementos y los
5  accede a cada elemento del arreglo a través de un ciclo while
6 */
7
8 int main ()
9 {
10     //declaramos variable
11     char ao=162;
12
13     //empieza el código
14     #define TAMANO 5
15     int lista[TAMANO] = {10, 8, 5, 8, 7};
16
17     int indice = 0;
18
19     printf("\n\tLista\n");
20     while (indice < 5 )
21     {
22         printf("\nCalificación del alumno %d es %d",ao, indice+1, lista[indice]);
23         indice += 1; // análogo a indice = indice + 1;
24     }
25
26     printf("\n");
27
28     return 0;
29 }
```

```
C:\WINDOWS\system32\cmd.exe

C:\Users\user\Desktop\Lenguaje C\Ejemplos>gcc Arreglounidiwhile.c -o Arreglounidiwhile.exe

C:\Users\user\Desktop\Lenguaje C\Ejemplos>Arreglounidiwhile.exe

    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

C:\Users\user\Desktop\Lenguaje C\Ejemplos>
```

Código (arreglo unidimensional for)

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos\arreglounifor.c - Notepad++

Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar  Plugins  Ventana  ?

arreglounifor.c  Arreglounidiwhile.c  new.1  new.2

1  #include <stdio.h>
2  /*
3   Este programa genera un arreglo unidimensional de 5 elementos y
4   accede a cada elemento del arreglo a través de un ciclo for.
5  */
6  int main ()
7  {
8   //declaramos variable
9   char ao=162;
10
11   //empieza el código
12   #define TAMANO 5
13   int lista[TAMANO] = {10, 8, 5, 8, 7};
14
15   printf("\tLista\n");
16   for (int indice = 0 ; indice < 5 ; indice++)
17   {
18     printf("\nCalificación del alumno %d es %d", ao,indice+1, lista[indice]);
19   }
20
21   printf("\n");
22
23   return 0;
24 }
25
```

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos>gcc arreglounifor.c -o Arreglounifor.exe

C:\Users\user\Desktop\Lenguaje C\Ejemplos>arreglounifor.exe

    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

C:\Users\user\Desktop\Lenguaje C\Ejemplos>
```

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;

Cuando a una variable le antecede un &, 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)

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos\apuntadores.c - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar  Plugins  Ventana  ?
arreglounfor.c  arreglounidwhile.c  apuntadores.c  new 2

1  #include <stdio.h>
2  /*
3   * Este programa crea un apuntador de tipo carácter.
4   */
5  int main ()
6  {
7      //Se declaran las variables
8      char ao=162;
9      char aa=160;
10
11     //Empieza el código
12     char *ap, c = 'a';
13     ap = &c;
14
15     printf("\nCarácter: %c\n", aa,*ap);
16     printf("Código ASCII: %d\n", ao,*ap);
17     printf("Dirección de memoria: %d\n", ao,ap);
18
19     return 0;
20 }
21
```

```
C:\WINDOWS\system32\cmd.exe

C:\Users\user\Desktop\Lenguaje C\Ejemplos>gcc apuntadores.c -o apuntadores.exe

C:\Users\user\Desktop\Lenguaje C\Ejemplos>apuntadores.exe

Carácter: a
Código ASCII: 97
Dirección de memoria: 6422295

C:\Users\user\Desktop\Lenguaje C\Ejemplos>
```

Código (apuntadores)

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos\apuntadores2.c - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar  Plugins  Ventana  ?
apuntadores2.c
1  #include<stdio.h>
2  /*
3   Este programa accede a las localidades de memoria de distintas variables a
4   través de un apuntador.
5  */
6  int main ()
7  {
8      int a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0};
9      int *apEnt;
10     apEnt = &a;
11
12     printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n");
13     printf("apEnt = %a\n");
14
15     b = *apEnt;
16     printf("b = *apEnt \t-> b = %i\n", b);
17
18     b = *apEnt + 1;
19     printf("b = *apEnt + 1 \t-> b = %i\n", b);
20
21     *apEnt = 0;
22     printf("*apEnt = 0 \t-> a = %i\n", a);
23
24     apEnt = &c[0];
25     printf("apEnt = %c[0] \t-> apEnt = %i\n", *apEnt);
26
27     return 0;
28 }
29
```

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos>gcc apuntadores2.c -o apuntadores2.exe
```

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos>apuntadores2.exe
```

```
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
```

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos>
```

Códigos (apuntadores)

```
1  #include <stdio.h>
2  /*
3   Este programa trabaja con aritmética de punteros para acceder a los
4   valores de un arreglo.
5  */
6  int main ()
7  {
8      int arr[] = {5, 4, 3, 2, 1};
9      int *apArr;
10     apArr = arr;
11
12     printf("\nint arr[] = {5, 4, 3, 2, 1};\n");
13     printf("apArr = &arr[0]\n");
14
15     int x = *apArr;
16     printf("x = *apArr \t -> x = %d\n", x);
17
18     x = *(apArr+1);
19     printf("x = *(apArr+1) \t -> x = %d\n", x);
20
21     x = *(apArr+2);
22     printf("x = *(apArr+1) \t -> x = %d\n", x);
23
24     return 0;
25 }
```

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos>gcc apuntador3.c -o apuntador3.exe
```

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos>apuntador3.exe
```

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

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos>
```


Código (apuntadores en ciclo for)

```
1 #include <stdio.h>
2 /*
3  Este programa genera un arreglo unidimensional de 5 elementos y
4  accede a cada elemento del arreglo a través de un apuntador
5  utilizando un ciclo for.
6  */
7 int main ()
8 {
9     //definir variables
10    char ao=162;
11
12    //empieza el apuntador
13    #define TAMANO 5
14    int lista[TAMANO] = {10, 8, 5, 8, 7};
15    int *ap = lista;
16    printf("\tLista\n");
17    for (int indice = 0 ; indice < 5 ; indice++)
18    {
19        printf("\nCalificaci%cn del alumno %d es %d", ao, indice+1, *(ap+indice));
20    }
21
22    printf("\n");
23
24    return 0;
25 }
```

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos>gcc apuntadorfor.c -o apuntadorfor.exe
```

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos>apuntadorfor.exe
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
```

```
C:\Users\user\Desktop\Lenguaje C\Ejemplos>
```

Código (apuntadores en cadenas)

```
1 #include <stdio.h>
2 /*
3  Este programa muestra el manejo de cadenas en lenguaje C.
4  */
5 int main()
6 {
7     char palabra[20];
8     int i=0;
9
10    printf("\nIngrese una palabra: ");
11    scanf("%s", palabra);
12    printf("\nLa palabra ingresada es: %s\n", palabra);
13
14    for (i = 0 ; i < 20 ; i++)
15    {
16        printf("\n%c\n", palabra[i]);
17    }
18
19    return 0;
20 }
```

```
C:\WINDOWS\system32\cmd.exe

C:\Users\user\Desktop\Lenguaje C\Ejemplos>apuntadorcadena.exe

Ingrese una palabra: emiliano

La palabra ingresada es: emiliano

e
m
i
l
i
a
n
o
↓
@
```

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)

```
apuntadorfor.c | apuntadorcadena.c | arreglomulti.c
1  #include<stdio.h>
2  /* Este programa genera un arreglo de dos dimensiones (arreglo
3     multidimensional) y accede a sus elementos a través de dos ciclos
4     for, uno anidado dentro de otro.
5  */
6  int main()
7  {
8     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
9     int i, j;
10    printf("\nImprimir Matriz\n");
11    for (i=0 ; i<3 ; i++)
12    {
13        for (j=0 ; j<3 ; j++)
14        {
15            printf("%d, ",matriz[i][j]);
16        }
17        printf("\n");
18    }
19    return 0;
20 }
21
```



```

C:\Users\user\Desktop\Lenguaje C\Ejemplos>gcc arreglomulti.c -o arreglomulti.exe

C:\Users\user\Desktop\Lenguaje C\Ejemplos>arreglomulti.exe

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

C:\Users\user\Desktop\Lenguaje C\Ejemplos>

```

Código (arreglos multidimensionales con apuntadores)

```

C:\Users\user\Desktop\Lenguaje C\Ejemplos\multiapu.c - Notepad++
Archivo  Editar  Buscar  Vista  Codificación  Lenguaje  Configuración  Herramientas  Macro  Ejecutar  Plugins  Ventana  ?
multiapu.c
1  #include<stdio.h>
2  /* Este programa genera un arreglo de dos dimensiones (arreglo
3  multidimensional) y accede a sus elementos a través de un apuntador utilizando
4  un ciclo for.
5  */
6  int main()
7  {
8      int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
9      int i, cont=0, *ap;
10     ap = matriz;
11
12     printf("\nImprimir Matriz\n");
13     for (i=0 ; i<9 ; i++)
14     {
15         if (cont == 3)
16         {
17             printf("\n");
18             cont = 0;
19         }
20         printf("%d\t",*(ap+i));
21         cont++;
22     }
23     printf("\n");
24     return 0;
25 }

```

```

C:\WINDOWS\system32\cmd.exe
C:\Users\user\Desktop\Lenguaje C\Ejemplos>gcc multiapu.c -o multiapu.exe
multiapu.c: In function 'main':
multiapu.c:10:5: warning: assignment to 'int *' from incompatible pointer type 'int (*)[3]' [-Wincompatible-pointer-types]
   10 |     ap = matriz;
      |         ^
C:\Users\user\Desktop\Lenguaje C\Ejemplos>multiapu.exe

Imprimir Matriz
1      2      3
4      5      6
7      8      9

C:\Users\user\Desktop\Lenguaje C\Ejemplos>_

```

Conclusiones

Aprendimos como ocupar los arreglos y los apuntadores, en la hora de programar, su escritura y su uso de manera correcta.