



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: M.I. Marco Antonio Martínez Quintana

Asignatura: Fundamentos de Programación

Grupo: 3

No de Práctica(s): 11

Integrante(s): Ramirez Garcia Diego Andres

*No. de Equipo de
cómputo empleado:* No Aplica

No. de Lista o Brigada: 35

Semestre: Primero

Fecha de entrega: 04/01/2020

Observaciones: Perfecta complementación para el desarrollo de software dentro
del Lenguaje C.

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.

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

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:

```
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.

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:

```
TipoDeDato *apuntador, variable;  
apuntador = &variable;
```

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 `*`.

Arreglos multidimensionales

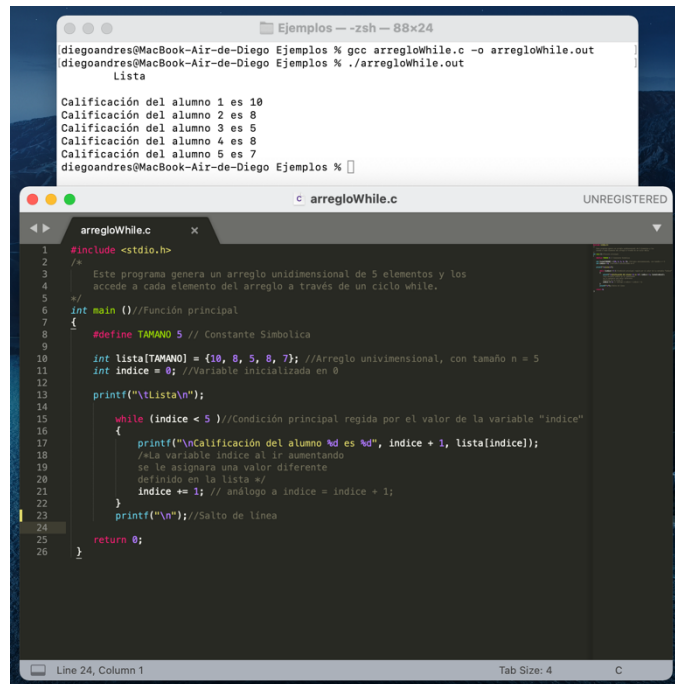
Lenguaje C permite crear arreglos de varias dimensiones con la siguiente sintaxis:

```
tipoDato nombre[tamaño][tamaño]...[tamaño];
```

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.

Desarrollo

Código (arreglo unidimensional while)

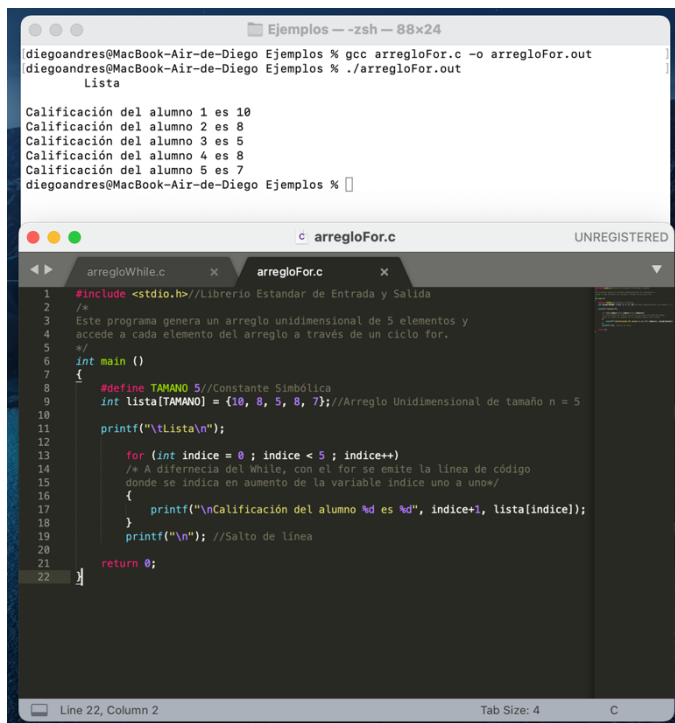


The screenshot shows a terminal window at the top and a code editor window below it. The terminal displays the output of a C program that uses a while loop to iterate through an array of student scores. The code editor shows the source code for `arregloWhile.c`, which includes a header file, a comment describing the program, and the `main` function. The `main` function defines a constant `TAMANO` of 5, initializes an array `lista` with the values {10, 8, 5, 8, 7}, and uses a while loop to print each element and its index.

```
Ejemplos -- zsh -- 88x24
diegoandres@MacBook-Air-de-Diego Ejemplos % gcc arregloWhile.c -o arregloWhile.out
diegoandres@MacBook-Air-de-Diego Ejemplos % ./arregloWhile.out
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
diegoandres@MacBook-Air-de-Diego Ejemplos %

arregloWhile.c
1 #include <stdio.h>
2 /*
3  * Este programa genera un arreglo unidimensional de 5 elementos y los
4  * accede a cada elemento del arreglo a través de un ciclo while.
5  */
6 int main () //Función principal
7 {
8     #define TAMANO 5 // Constante Simbolica
9
10    int lista[TAMANO] = {10, 8, 5, 8, 7}; //Arreglo univimensional, con tamaño n = 5
11    int indice = 0; //Variable inicializada en 0
12
13    printf("\n\tLista\n");
14
15    while (indice < 5) //Condición principal regida por el valor de la variable "indice"
16    {
17        printf("\nCalificación del alumno %d es %d", indice + 1, lista[indice]);
18        /*La variable indice al ir aumentando
19        se le asignara una valor diferente
20        definido en la lista */
21        indice ++ 1; // análogo a indice = indice + 1;
22    }
23    printf("\n"); //Salto de línea
24
25    return 0;
26 }
```

Código (arreglo unidimensional for)



The screenshot shows a terminal window at the top and a code editor window below it. The terminal displays the output of a C program that uses a for loop to iterate through an array of student scores. The code editor shows the source code for `arregloFor.c`, which includes a header file, a comment describing the program, and the `main` function. The `main` function defines a constant `TAMANO` of 5, initializes an array `lista` with the values {10, 8, 5, 8, 7}, and uses a for loop to print each element and its index.

```
Ejemplos -- zsh -- 88x24
diegoandres@MacBook-Air-de-Diego Ejemplos % gcc arregloFor.c -o arregloFor.out
diegoandres@MacBook-Air-de-Diego Ejemplos % ./arregloFor.out
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
diegoandres@MacBook-Air-de-Diego Ejemplos %

arregloFor.c
1 #include <stdio.h> //Libreria Estandar de Entrada y Salida
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     #define TAMANO 5 //Constante Simbólica
9
10    int lista[TAMANO] = {10, 8, 5, 8, 7}; //Arreglo Unidimensional de tamaño n = 5
11
12    printf("\n\tLista\n");
13
14    for (int indice = 0 ; indice < 5 ; indice++)
15    /* A diferencia del while, con el for se emite la línea de código
16    donde se indica en aumento de la variable indice uno a uno */
17    {
18        printf("\nCalificación del alumno %d es %d", indice+1, lista[indice]);
19    }
20    printf("\n"); //Salto de línea
21
22    return 0;
23 }
```

Código (apuntadores)

```
Ejemplos --zsh-- 88x24
diegoandres@MacBook-Air-de-Diego Ejemplos % gcc apuntador.c -o apuntador.out
diegoandres@MacBook-Air-de-Diego Ejemplos % ./apuntador.out
Carácter: a
Código ASCII: 97
Dirección de memoria: a???.??
diegoandres@MacBook-Air-de-Diego Ejemplos %
```

```
apuntador.c
1 #include <stdio.h> //Librería Estandar de Entrada y Salida
2 /*
3  * Este programa crea un apuntador de tipo carácter.
4  */
5 int main ()
6 {
7     char *ap, c = 'a'; //La variable apuntador "ap" apuntara hacia la variable "c"
8     ap = &c; //Accede al contenido de la dirección de memoria de la variable "c"
9
10    printf("Carácter: %c\n", *ap); //Especificador de formato para imprimir un caracter
11    printf("Código ASCII: %d\n", *ap);
12    /*
13     * Especificador de formato para imprimir el
14     * valor del código ASCII del caracter en base 10
15     */
16    printf("Dirección de memoria: %s\n", ap);
17    /*
18     * Tuve que cambiar el especificador de dato de decimal
19     * a cadena de caracter debido a las especificaciones de
20     * la computadora
21     */
22    return 0;
23 }
24
```

Código (apuntadores)

```
Ejemplos --zsh-- 97x24
diegoandres@MacBook-Air-de-Diego Ejemplos % gcc apuntadoryarreglo.c -o apuntadoryarreglo.out
diegoandres@MacBook-Air-de-Diego Ejemplos % ./apuntadoryarreglo.out
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
diegoandres@MacBook-Air-de-Diego Ejemplos %
```

```
apuntadoryarreglo.c
1 #include <stdio.h> //Librería Estandar de Entrada y Salida
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     /*
10     * Tres variables de tipo entero,
11     * una de ellas es un arreglo de tamaño n = 10
12     */
13     int *apEnt; //Apuntador "apEnt"
14     apEnt = &a; //Accede al contenido de la dirección de memoria de la variable "a"
15
16     printf("a = 5, b = 10, c[10] = {5, 4, 3, 2, 1, 9, 8, 7, 6, 0}\n");
17     printf("apEnt = %s\n", apEnt);
18     b = *apEnt; // "b" adquiere el valor del apuntador, que es igual a "a"
19     printf("b = *apEnt \t-> b = %i\n", b); //b=5
20     b = *apEnt + 1; // "b" se incrementa en una unidad, es decir, que "a" se incrementa en una unidad
21     printf("b = *apEnt + 1 \t-> b = %i\n", b); //b=6
22     *apEnt = 0; // Se asigna a "apEnt" un valor de 0, que se le asigna a la variable "a"
23     printf("apEnt = 0 \t-> a = %i\n", a); // a=0
24     apEnt = &c[0]; // "apEnt" accede al contenido de la dirección de la memoria de la variable c con tamaño n = 0
25     printf("apEnt = &c[0] \t-> apEnt = %i\n", *apEnt); //apEnt = 5
26     return 0;
27 }
```

Código (apuntadores)

```
diegoandres@MacBook-Air-de-Diego Ejemplos % gcc apuntadorX.c -o apuntadorX.out
diegoandres@MacBook-Air-de-Diego Ejemplos % ./apuntadorX.out
int arr[] = {5, 4, 3, 2, 1};
apArr = &arr[0]
x = *apArr      -> x = 5
x = *(apArr+1)  -> x = 4
x = *(apArr+1)  -> x = 3
diegoandres@MacBook-Air-de-Diego Ejemplos %
```

```
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}; //Arreglo sin tamaño
9
10    int *apArr; //Puntero
11    apArr = arr; //apArr adquiere el valor de "arr", toma el primer valor definido para "arr"
12
13    printf("int arr[] = {5, 4, 3, 2, 1};\n");
14    printf("apArr = %p\n", apArr);
15
16    int x = *apArr; //x adquiere el valor de "apArr", es decir de "arr[0]"
17    printf("x = *apArr \t -> x = %d\n", x); //Como "apArr = 5", "x = 5"
18
19    x = *(apArr+1); //arr incrementa en una unidad, por lo que "arr[1] = 4" y x = 4
20    printf("x = *(apArr+1) \t -> x = %d\n", x);
21
22    x = *(apArr+2); //arr incrementa en una unidad, por lo que "arr[2] = 3" y x = 3
23    printf("x = *(apArr+1) \t -> x = %d\n", x);
24
25    return 0;
26 }
```

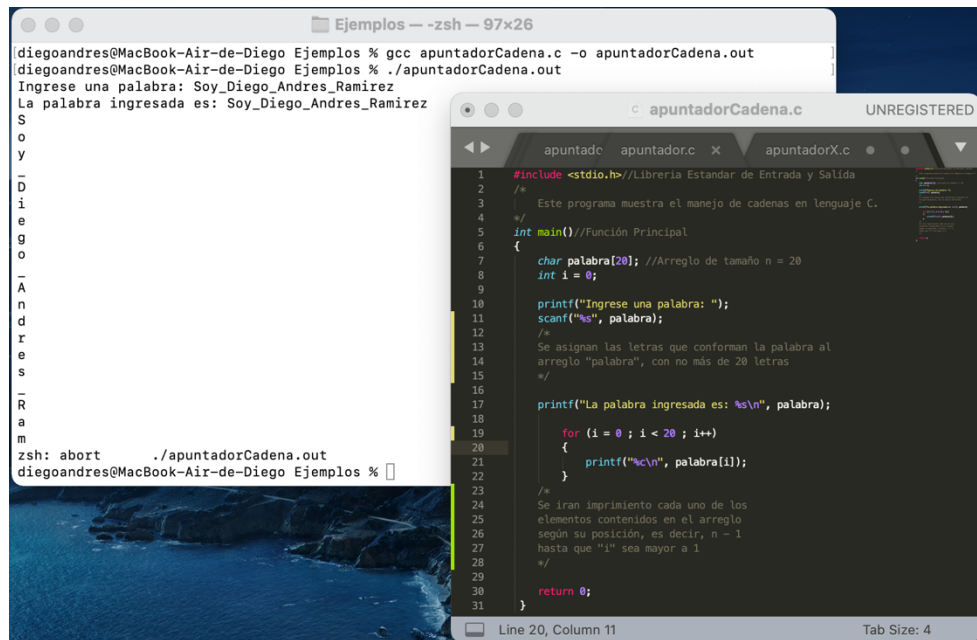
Código (apuntadores en ciclo for)

```
diegoandres@MacBook-Air-de-Diego Ejemplos % gcc apuntadorFor.c -o apuntadorFor.out
diegoandres@MacBook-Air-de-Diego Ejemplos % ./apuntadorFor.out
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
diegoandres@MacBook-Air-de-Diego Ejemplos %
```

```
1 #include <stdio.h>
2 /*
3  Este programa genera un arreglo unidimensional de 5 elementos y accede a
4  cada elemento del arreglo a través de un puntero utilizando un ciclo for.
5  */
6 int main ()
7 {
8     #define TAMANO 5 //Constante Simbólica
9     int lista[TAMANO] = {10, 8, 5, 8, 7}; //Arreglo con tamaño n=5
10
11    int *ap = lista; //Puntero "ap" es igual a arreglo "lista"
12
13    printf("\nLista\n");
14
15    for (int indice = 0; indice < 5; indice++)
16    /*
17     Índice ira incrementando hasta que no se compla la condición
18     */
19    {
20        printf("\nCalificación del alumno %d es %d", indice+1, *(ap+indice));
21        /*
22         Lista[0] incrementa en una unidad a "n"
23         por lo que ahora es lista[1] que es igual a: ap = 8
24         */
25    }
26    printf("\n");
27    return 0;
28 }
```

Código (apuntadores en cadenas)

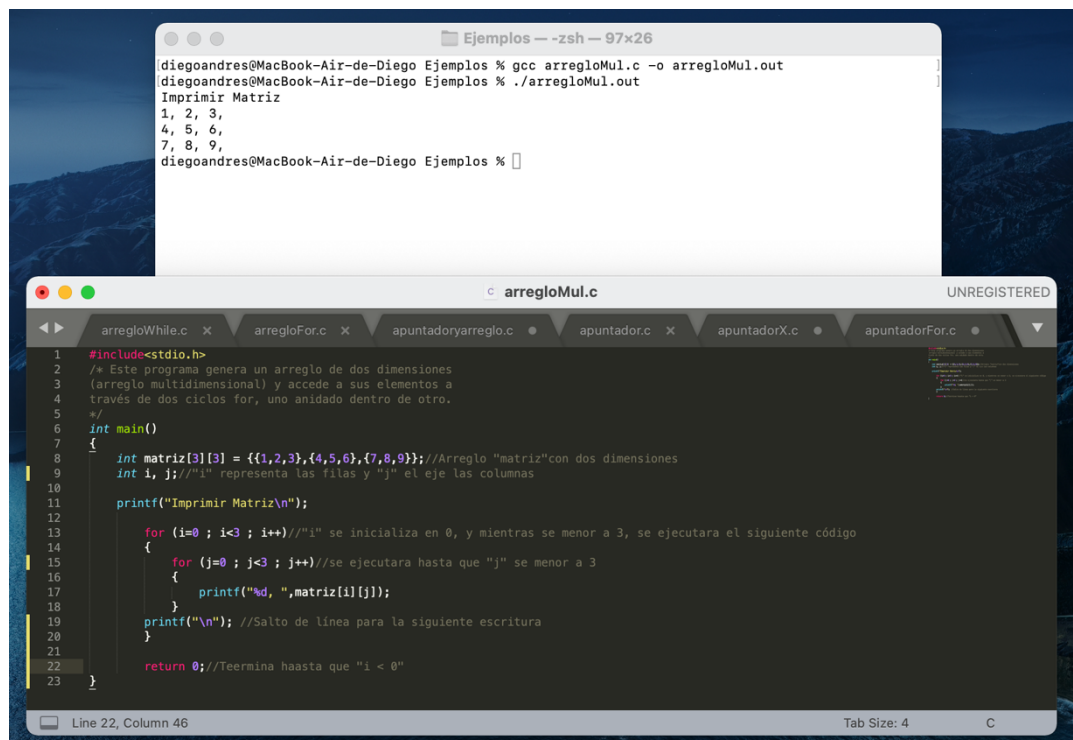


The image shows a terminal window and a code editor. The terminal window, titled "Ejemplos - zsh - 97x26", displays the compilation and execution of a C program. The code editor, titled "apuntadorCadena.c", shows the source code of the program.

```
diegoandres@MacBook-Air-de-Diego Ejemplos % gcc apuntadorCadena.c -o apuntadorCadena.out
diegoandres@MacBook-Air-de-Diego Ejemplos % ./apuntadorCadena.out
Ingrese una palabra: Soy_Diego_Andres_Ramirez
La palabra ingresada es: Soy_Diego_Andres_Ramirez
S
o
y
-
D
i
e
g
o
-
A
n
d
r
e
s
-
R
a
m
zsh: abort ./apuntadorCadena.out
diegoandres@MacBook-Air-de-Diego Ejemplos %
```

```
1 #include <stdio.h> //Libreria Estandar de Entrada y Salida
2 /*
3  Este programa muestra el manejo de cadenas en lenguaje C.
4  */
5 int main() //Función Principal
6 {
7     char palabra[20]; //Arreglo de tamaño n = 20
8     int i = 0;
9
10    printf("Ingrese una palabra: ");
11    scanf("%s", palabra);
12    /*
13     Se asignan las letras que conforman la palabra al
14     arreglo "palabra", con no más de 20 letras
15     */
16    printf("La palabra ingresada es: %s\n", palabra);
17
18    for (i = 0 ; i < 20 ; i++)
19    {
20        printf("%c\n", palabra[i]);
21    }
22    /*
23     Se iran imprimiendo cada uno de los
24     elementos contenidos en el arreglo
25     según su posición, es decir, n - 1
26     hasta que "i" sea mayor a 1
27     */
28    return 0;
29 }
30
31
```

Código (arreglos multidimensionales)

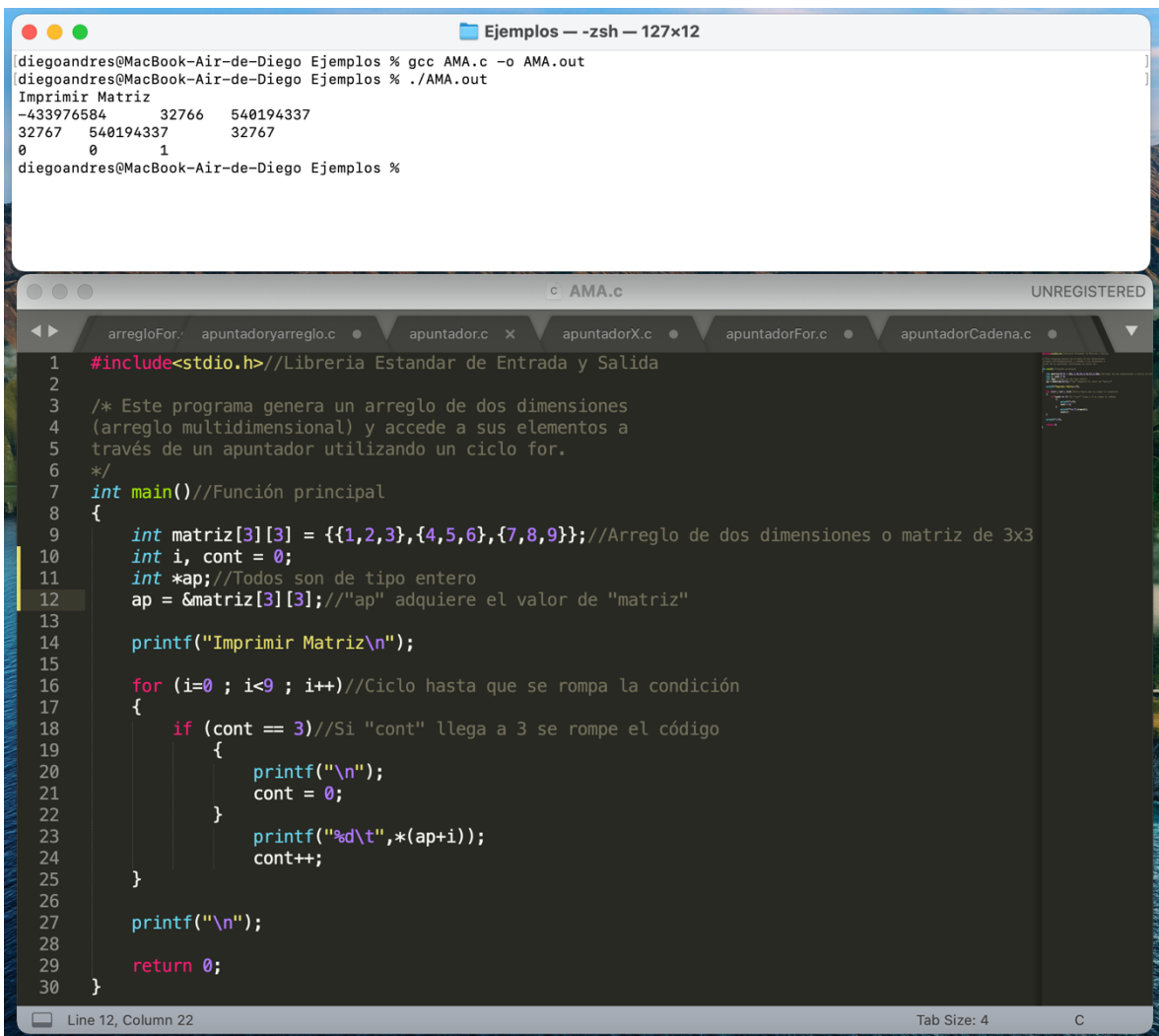


The image shows a terminal window and a code editor. The terminal window, titled "Ejemplos - zsh - 97x26", displays the compilation and execution of a C program. The code editor, titled "arregloMul.c", shows the source code of the program.

```
diegoandres@MacBook-Air-de-Diego Ejemplos % gcc arregloMul.c -o arregloMul.out
diegoandres@MacBook-Air-de-Diego Ejemplos % ./arregloMul.out
Imprimir Matriz
1, 2, 3,
4, 5, 6,
7, 8, 9,
diegoandres@MacBook-Air-de-Diego Ejemplos %
```

```
1 #include <stdio.h>
2 /* Este programa genera un arreglo de dos dimensiones
3  (arreglo multidimensional) y accede a sus elementos a
4  través de dos ciclos 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}}; //Arreglo "matriz" con dos dimensiones
9     int i, j; // "i" representa las filas y "j" el eje las columnas
10
11    printf("Imprimir Matriz\n");
12
13    for (i=0 ; i<3 ; i++) // "i" se inicializa en 0, y mientras se menor a 3, se ejecutara el siguiente código
14    {
15        for (j=0 ; j<3 ; j++) // se ejecutara hasta que "j" se menor a 3
16        {
17            printf("%d, ",matriz[i][j]);
18        }
19        printf("\n"); //Salto de línea para la siguiente escritura
20    }
21    return 0; //Termina hasta que "i < 0"
22 }
23
```

Código (arreglos multidimensionales con apuntadores)



The image shows a terminal window and a code editor. The terminal window, titled "Ejemplos -- zsh -- 127x12", displays the output of a C program. The code editor, titled "c AMA.c", shows the source code of the program. The code defines a 3x3 matrix and uses a pointer to access its elements.

```
diegoandres@MacBook-Air-de-Diego Ejemplos % gcc AMA.c -o AMA.out
diegoandres@MacBook-Air-de-Diego Ejemplos % ./AMA.out
Imprimir Matriz
-433976584      32766      540194337
32767      540194337      32767
0              1
diegoandres@MacBook-Air-de-Diego Ejemplos %
```

```
1  #include<stdio.h>//Libreria Estandar de Entrada y Salida
2
3  /* Este programa genera un arreglo de dos dimensiones
4  (arreglo multidimensional) y accede a sus elementos a
5  través de un apuntador utilizando un ciclo for.
6  */
7  int main()//Función principal
8  {
9      int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};//Arreglo de dos dimensiones o matriz de 3x3
10     int i, cont = 0;
11     int *ap;//Todos son de tipo entero
12     ap = &matriz[3][3];/*"ap" adquiere el valor de "matriz"
13
14     printf("Imprimir Matriz\n");
15
16     for (i=0 ; i<9 ; i++)//Ciclo hasta que se rompa la condición
17     {
18         if (cont == 3)//Si "cont" llega a 3 se rompe el código
19         {
20             printf("\n");
21             cont = 0;
22         }
23         printf("%d\t",*(ap+i));
24         cont++;
25     }
26     printf("\n");
27
28     return 0;
29 }
30
```

Line 12, Column 22 Tab Size: 4 C

Conclusiones

El manejo y conocimiento de los arreglos existentes en el Lenguaje C nos permite desarrollar programas cuya optimización en memoria y arranque sea mayor. Los arreglos unidimensionales o multidimensionales, acortan de igual forma las líneas dentro del código, sin embargo, su manejo debe ser considerable ya que en caso de no comprenderlos y emplearlos, se tiende a cometer un mayor número de errores, que únicamente se puede ver mediante el uso de la depuración dentro de un IDE donde se observe el comportamiento de cada una de las variables involucradas.

Por su parte los apuntadores al ser más complejos se debe tener mucho cuidado con su uso, sin embargo son de gran utilidad en compañía de los arreglos unidimensionales y multidimensionales.

Referencias

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

“Ingeniería en Computación (2018) Guía de estudio práctica 06: Entorno de C [editores, compilación y ejecución], Manual de Prácticas del Laboratorio de Fundamentos de Programación”