



Carátula para entrega de prácticas

Facultad de Ingeniería

Laboratorio de docencia

Laboratorios de computación salas A y B

Profesor: Karina García Morales

Asignatura: Fundamentos de programación

Grupo: 20

No. de práctica(s): 10

Integrante(s): Santiago Maya Luis Angel

No. de lista o brigada: 45

Semestre: 2023-1

Fecha de entrega: 6-Dic-22

Observaciones:

CALIFICACIÓN: _____

Objetivo

El alumno utilizará arreglos de dos dimensiones en la elaboración de programas que resuelvan problemas que requieran agrupar datos del mismo tipo, en estructuras que utilicen dos índices.

Introducción

Un arreglo de dos dimensiones es un conjunto de datos contiguos del mismo tipo con un tamaño fijo definido al momento de crearse. Para acceder a un elemento en este tipo de arreglos se requiere el uso de dos índices. Los arreglos se utilizan para hacer más eficiente el código de un programa, ya que la manipulación de datos del mismo tipo que son agrupados en un arreglo por tener un significado común se realiza de una forma más clara y eficaz.

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). El tipoDeDato es el tipo de dato de los elementos del arreglo, el cual puede ser tipo 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.

Esta práctica estará enfocada exclusivamente al manejo de arreglos de dos dimensiones (bidimensionales).

Como ya lo hemos leído, la practica nos va a enseñar a crear y comprender los arreglos multidimensionales, por ahora enfocándonos en específicamente los bidimensionales. A continuación, compilaremos y ejecutaremos un programa que genera un arreglo de dos dimensiones (arreglo multidimensional) y accede a cada uno de sus elementos por la posición que indica el renglón y la columna a través de dos ciclos for, uno anidado dentro de otro.

Este programa que hemos realizado es de tamaño 3x3 y de salida nos ha dado una matriz de 3 filas y 3 columnas con números que previamente ya estaban declarados eh inicializados en el arreglo.

```
5 // Created by Santiago Maya Luis Angel on 11/30/22.
6 // Copyright © 2022 Santiago Maya Luis Angel. All rights reserved.
7 //
8
9 #include<stdio.h>
10
11 int main() {
12     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
13     int i, j;
14     printf("Imprimir Matriz\n");
15     for (i=0 ; i<3 ; i++) //Representa al renglón del arreglo
16     {
17         for (j=0 ; j<3 ; j++)//Representa a la columna del arreglo
18         {
19             printf("%d, ",matriz[i][j]);
20         }
21         printf("\n");
22     }
23
24 }
25 return 0;
26
27 }
28
```

Imprimir Matriz

```
1, 2, 3,
4, 5, 6,
7, 8, 9,
Program ended with exit code: 0
```

Ahora compilamos y corremos nuestro segundo programa que genera un arreglo de dos dimensiones y accede a sus elementos por la posición que indica el renglón y la columna a través de dos ciclos for, uno anidado dentro de otro, el contenido de cada elemento de este arreglo es la suma de sus índices.

Este programa es casi lo mismo al otro sin decir que son los mismos, ya que en este lo que cambia es el tamaño del arreglo mas que aparte los valores del arreglo se van guardando de los ciclos for que se están realizando, para finalizar imprimiendo una matriz de 5x5

```
5 // Created by Santiago Maya Luis Angel on 11/30/22.
6 // Copyright © 2022 Santiago Maya Luis Angel. All rights reserved.
7 //
8
9 #include<stdio.h>
10 int main() {
11     int i,j,a[5][5];
12     for (i=0 ; i<5 ; i++)//Representa al renglón del arreglo
13     {
14         for (j=0 ; j<5 ; j++)//Representa a la columna del arreglo
15         {
16             a[i][j]=i+j;
17             printf("\t%d, ",a[i][j]);
18         }
19         printf("\n");
20     }
21     return 0; }
22
23
24 |
```

0,	1,	2,	3,	4,
1,	2,	3,	4,	5,
2,	3,	4,	5,	6,
3,	4,	5,	6,	7,
4,	5,	6,	7,	8,

Program ended with exit code: 0

A este mismo programa lo modificamos para que ahora fuera de tamaño 6x6, ahora imprimiéndonos hasta el numero 10

```
4 //
5 // Created by Santiago Maya Luis Angel on 11/30/22.
6 // Copyright © 2022 Santiago Maya Luis Angel. All rights reserved.
7 //
8
9 #include<stdio.h>
10 int main() {
11     int i,j,a[10][10];
12     for (i=0 ; i<6 ; i++)//Representa al renglón del arreglo
13     {
14         for (j=0 ; j<6 ; j++)//Representa a la columna del arreglo
15         {
16             a[i][j]=i+j;
17             printf("\t%d, ",a[i][j]);
18         }
19         printf("\n");
20     }
21     return 0; }
22
23
24
```

0,	1,	2,	3,	4,	5,
1,	2,	3,	4,	5,	6,
2,	3,	4,	5,	6,	7,
3,	4,	5,	6,	7,	8,
4,	5,	6,	7,	8,	9,
5,	6,	7,	8,	9,	10,

Program ended with exit code: 0

Códigos (arreglos multidimensionales usando while)

Al seguir con la práctica, ahora tenemos que realizar la compilación y ejecución del siguiente programa que se observa a continuación, este genera un arreglo de dos dimensiones (arreglo multidimensional) y accede a sus elementos por la posición que indica el renglón y la columna a través de dos ciclos while, uno anidado dentro de otro.

Este programa es muy similar a los anteriores solo que ahora usamos el ciclo while para su función, aparte que su forma de imprimir los números cambio a la de los anteriores y en una matriz de 4x4.

```
5 // Created by Santiago Maya Luis Angel on 11/30/22.
6 // Copyright © 2022 Santiago Maya Luis Angel. All rights reserved.
7 //
8
9 #include<stdio.h>
10 int main() {
11     int matriz[4][4] = {{1,2,3,4},{4,5,6,7},{7,8,9,10},{10,11,12,13}}; int i, j;
12     printf("Imprimir Matriz\n");
13     i=0;
14     while(i<4) //Representa al renglón del arreglo
15     {
16         j=0;
17         while (j<4)//Representa a la columna del arreglo
18         {
19             printf("%d, ",matriz[i][j]);
20             j++; }
21             printf("\n");
22             i++; }
23     return 0; }
```

Imprimir Matriz

1, 2, 3, 4,
4, 5, 6, 7,
7, 8, 9, 10,
10, 11, 12, 13,

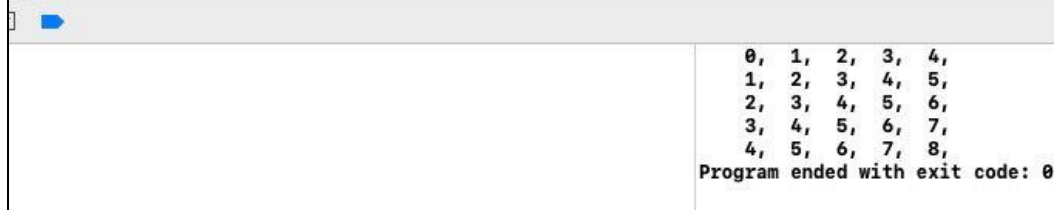
Ahora realizamos el programa 2b, que es lo mismo que el 2ª, solo que ahora en vez de usar los ciclos for anidados. Hacemos uso del ciclo while

```
4 //
5 // Created by Santiago Maya Luis Angel on 11/30/22.
6 // Copyright © 2022 Santiago Maya Luis Angel. All rights reserved.
7 //
8
9 #include<stdio.h>
10 int main() {
11     int i,j,a[5][5];
12     i=0;
13     while (i<5) //Representa al renglón del arreglo
14     {
15         j=0;
16         while (j<5) //Representa a la columna del arreglo
17         {
18             a[i][j]=i+j;
19             printf("\t%d, ",a[i][j]);
20             j++;
21         }
22         printf("\n");
23         i++;
24     }
25     return 0;
26 }
27 }
```

0, 1, 2, 3, 4,
1, 2, 3, 4, 5,
2, 3, 4, 5, 6,
3, 4, 5, 6, 7,
4, 5, 6, 7, 8,
Program ended with exit code: 0

Y a continuación, nuestro programa 1c, que como antes dicho es lo mismo solo que ahora utilizamos la estructura do while

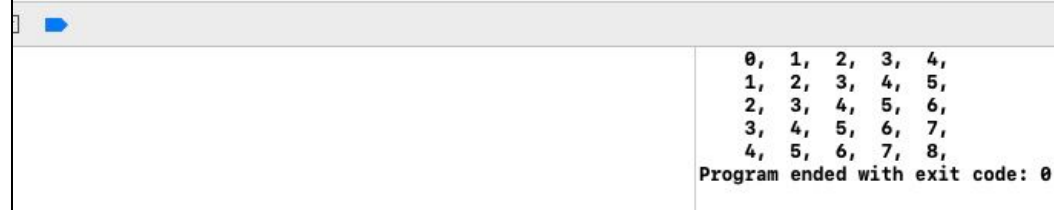
```
1 //
2 // main.c
3 // Arreglos multidimensionales
4 //
5 // Created by Santiago Maya Luis Angel on 11/30/22.
6 // Copyright © 2022 Santiago Maya Luis Angel. All rights reserved.
7 //
8
9 #include<stdio.h>
10 int main() {
11     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}}; int i, j;
12     printf("Imprimir Matriz\n");
13     i=0;
14     do //Representa al renglón del arreglo
15     {
16         j=0;
17         do //Representa a la columna del arreglo
18         {
19             printf("%d, ",matriz[i][j]);
20             j++; }
21         while (j<3); printf("\n");
22         i++; }
23     while(i<3);
24     return 0; }
25
```



0, 1, 2, 3, 4,
1, 2, 3, 4, 5,
2, 3, 4, 5, 6,
3, 4, 5, 6, 7,
4, 5, 6, 7, 8,
Program ended with exit code: 0

Así como hemos realizado el cambio de estructura en el código anterior, lo realizaremos en el 2c de igual manera utilizando el ciclo do while, y esto imprime los mismos resultados que los de los anteriores códigos descendientes el 2ª

```
1 //
2 // main.c
3 // Arreglos multidimensionales
4 //
5 // Created by Santiago Maya Luis Angel on 11/30/22.
6 // Copyright © 2022 Santiago Maya Luis Angel. All rights reserved.
7 //
8
9 #include<stdio.h>
10 int main() {
11     int i,j,a[5][5];
12     i=0;
13     do //Representa al renglón del arreglo
14     {
15         j=0;
16         do //Representa a la columna del arreglo
17         {
18             a[i][j]=i+j; printf("\t%d, ",a[i][j]); j++;
19         }
20         while (j<5); printf("\n"); i++;
21     }
22     while (i<5); return 0;
23 }
24
```



0, 1, 2, 3, 4,
1, 2, 3, 4, 5,
2, 3, 4, 5, 6,
3, 4, 5, 6, 7,
4, 5, 6, 7, 8,
Program ended with exit code: 0

Previamente realizados los anteriores 6 programas utilizando diferentes estructuras, realizaremos el programa3, este programa genera un arreglo multidimensional de máximo 10 renglones y 10 columnas, para poder almacenar datos en cada elemento y posteriormente mostrar el contenido de esos elementos se hace uso de ciclos for anidados respectivamente.

Para este programa decidimos reducir el tamaño del arreglo, de 10 a 5 para que no ingresáramos tantos valores, así mismo, este programa funciona a base del uso de for anidados, que, al ejecutarlo, nos pide 5 calores los cuales imprimió en una matriz de 3x3

```

5 // Created by Santiago Maya Luis Angel on 11/30/22.
6 // Copyright © 2022 Santiago Maya Luis Angel. All rights reserved.
7 //
8
9 #include <stdio.h>
10 int main () {
11     int lista[5][5]; // Se declara el arreglo multidimensional
12     int i,j;
13     int renglon,columna;
14     printf("\nDa el número de renglones y columnas separados con coma\n");
15     scanf("%d,%d",&renglon,&columna);
16     if(((renglon>=1) && (renglon<=5))&&((columna>=1) && (columna<=5)))
17     {
18         // Acceso a cada elemento del arreglo multidimensional usando for
19         for (i= 0 ; i <= renglon-1 ; i++) {
20             for(j= 0 ; j <= columna-1 ; j++) {
21                 printf("\nNúmero para el elemento %d,%d del arreglo", i,j );
22                 scanf("%d",&lista[i][j]);
23             }
24             printf("\nLos valores dados son: \n");
25             // Acceso a cada elemento del arreglo multidimensional usando for
26             for (i= 0 ; i <= renglon-1 ; i++) {
27                 for(j= 0 ; j <= columna-1 ; j++) {
28                     printf("%d ", lista[i][j]);
29                 }
30                 printf("\n");
31             }
32         }
33     else printf("Los valores dados no es válido"); printf("\n");
34     return 0;
35 }

```

Número para el elemento 1,1 del arreglo	4
Número para el elemento 1,2 del arreglo	7
Número para el elemento 2,0 del arreglo	5
Número para el elemento 2,1 del arreglo	3
Número para el elemento 2,2 del arreglo	2
Los valores dados son:	
13	4 4
7	4 7
5	3 2

Apuntadores y su relación con arreglos de dos dimensiones

Recordemos que 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 referirse a la dirección de memoria donde se ubica el valor de dicha variable (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 *. Como se mencionó al inicio de esta práctica, en la memoria cada elemento del arreglo se guarda de forma contigua, por lo tanto, se puede recorrer un arreglo multidimensional con apuntadores. Por ejemplo, los elementos de un arreglo de 3 renglones y 2 columnas de nombre arr se almacenan en memoria como lo muestra la siguiente figura.

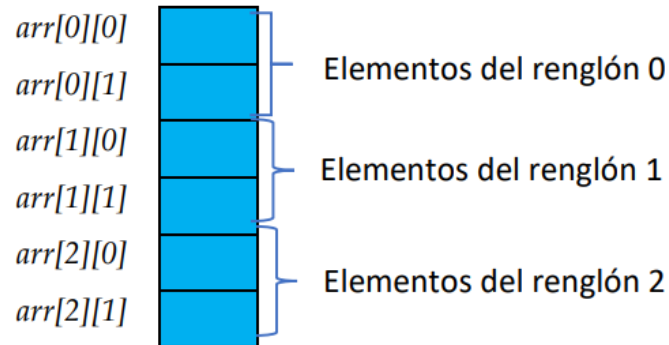


Figura 1. Almacenamiento en memoria de un arreglo de 3X2

Como podemos observar en la Figura 1, los elementos de un arreglo bidimensional se almacenan por renglones; esto se debe considerar al momento de acceder a los elementos del arreglo usando apuntadores.

Códigos (arreglos multidimensionales con apuntadores)

Ahora tenemos el programa 4a, este programa genera un arreglo de dos dimensiones (arreglo multidimensional) y accede a sus elementos a través de un apuntador utilizando un ciclo for.

```

1 // Created by Santiago Maya Luis Angel on 11/30/22.
2 // Copyright © 2022 Santiago Maya Luis Angel. All rights reserved.
3 //
4
5 #include<stdio.h>
6 int main() {
7     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
8     int i, cont=0, *ap;
9     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0]; printf("Imprimir Matriz\n");
10    for (i=0 ; i<9 ; i++)
11    {
12        if (cont == 3) //Este if sirve para que cada 3 numeros impresos haga un salto de linea y esto le da la forma de 3x3
13            a la matriz
14        {
15            printf("\n");
16            cont = 0; //Inicia conteo de elementos del siguiente renglón
17        }
18        printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz
19        cont++;
20    } printf("\n"); return 0;
21 }

```

1	2	3
4	5	6
7	8	9

Este programa nos devuelve una matriz de 3x3 a diferencia de los otros programas realizados previamente, este hace el uso de apuntadores, aparte que para saber cuáles valores guardados utilizara hace uso de un ciclo for y una estructura condicional if.

Ahora del mismo programa anterior realizaremos la misma sintaxis, solo que ahora, cambiaremos nuestro tamaño del arreglo, aparte de que en este se utilizara el ciclo while. Para realizar la ejecución y compilación de este mismo.

Lo que obtenemos ahora es una matriz de 4x4 y un programa corriendo con base de ciclo while.

```
4 //
5 // Created by Santiago Maya Luis Angel on 11/30/22.
6 // Copyright © 2022 Santiago Maya Luis Angel. All rights reserved.
7 //
8
9 #include<stdio.h>
10 int main() {
11     int matriz[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
12     int i, cont=0, *ap;
13     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0]; printf("Imprimir Matriz\n");
14     i=0;
15     while (i<16)
16     {
17         if (cont == 4) //Se imprimió un renglón y se hace un salto de línea
18         {
19             printf("\n");
20             cont = 0; //Inicia conteo de elementos del siguiente renglón
21         }
22         printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz
23         cont++;
24         i++; }
25     printf("\n");
26     return 0; }
27
28
```

1	2	3	4
5	6	7	8
9	10	11	12
13	14	15	16

Program ended with exit code: 0

Y como hemos realizado con los anteriores, ahora este mismo programa se correrá, pero con base utilizando al ciclo do while, y su salida es exactamente igual a los anteriores.

```
5 // Created by Santiago Maya Luis Angel on 11/30/22.
6 // Copyright © 2022 Santiago Maya Luis Angel. All rights reserved.
7 //
8
9 #include<stdio.h>
10 int main() {
11     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
12     int i, cont=0, *ap;
13     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0]; printf("Imprimir Matriz\n");
14     i=0;
15     do
16     {
17         if (cont == 3) //Este if sirve para los saltos de línea y dependiendo el tamaño de nuestro arr
18             valor de este if
19         {
20             printf("\n");
21             cont = 0; //Inicia conteo de elementos del siguiente renglón
22         }
23         printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz
24         cont++;
25         i++;
26     }
27     while (i<9); printf("\n");
28     return 0; }
```

1	2	3
4	5	6
7	8	9

Program ended with exit code: 0

Tabla comparativa para el programa 2

Programa2a.c

```

8
9 #include <stdio.h> // aquí esta la librería
10 int main() {
11     int i,j,a[5][5];
12     for (i=0 ; i<5 ; i++)//Representa al renglón del arreglo
13     {
14         for (j=0 ; j<5 ; j++)//Representa a la columna del arreglo
15         {
16             a[i][j]=i+j;
17             printf("\t%d", a[i][j]); }
18         printf("\n"); }
19
20 }
21

```

Éste programa usa la estructura(*for*), si recordamos esta estructura inicializa la variable, esta la restricción y tiene el incremento.

Programa2b.c

```

8
9 #include <stdio.h> // aquí esta la librería
10 int main() {
11     int i,j,a[5][5];
12     i=0;
13     while (i<5) //Representa al renglón del arreglo
14     {
15         j=0;
16         while (j<5) //Representa a la columna del arreglo
17         {
18             a[i][j]=i+j;
19             printf("\t%d", a[i][j]); j++;
20         } printf("\n"); i++;
21     }
22     return 0;
23
24 }
25

```

En la estructura mientras inicializa la variable al principio del programa, después del while sigue la restricción y hasta el ultimo antes de terminar esta el incremento

Programa2c.c

```

8
9 #include <stdio.h> // aquí esta la librería
10 int main() {
11     int i,j,a[5][5];
12     i=0;
13     do //Representa al renglón del arreglo
14     {
15         j=0;
16         do //Representa a la columna del arreglo
17         {
18             a[i][j]=i+j; printf("\t%d", a[i][j]); j++;
19         }
20         while (j<5); printf("\n"); i++;
21     }
22     while (i<5);
23     return 0;
24 }
25

```

El hacer mientras (do-while) es algo parecido a while pero este hace el incremento y después la restricción y se declaran las variables al principio.

Tabla comparativa para el programa 4

Programa4a.c

```

9 #include <stdio.h> // aqui esta la libreria
10 int main() {
11     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
12     int i, cont=0, *ap;
13     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0];
14     printf("Imprimir Matriz\n");
15     for (i=0 ; i<9 ; i++)
16     {
17         if (cont == 3) //Se imprimió un renglón y se hace un salto de línea, sir
            ponga los numeros en una sola fila, si no que aga salto de línea |
18         {
19             printf("\n");
20             cont = 0; //Inicia conteo de elementos del siguiente renglón
21         }
22         printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz
23         cont++;
24     } printf("\n");
25     return 0;
26 }
27

```

El programa esta en la estructura para (*for*), y siempre en la estructura for en el mismo espacio se declara la variable, se hace la restricción y esta el incremento, todo en uno.

Programa4a.c

```

9 #include <stdio.h> // aqui esta la libreria
10 // programa 4b.c
11 int main() {
12     int matriz[4][4] = {{1,2,3,4},{5,6,7,8},{9,10,11,12},{13,14,15,16}};
13     int i, cont=0, *ap;
14     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0]; printf("Imprimir
        Matriz\n");
15     i=0;
16     while (i<16)
17     {
18         if (cont == 4) //Se imprimió un renglón y se hace un salto de línea
19         {
20             printf("\n");
21             cont = 0; //Inicia conteo de elementos del siguiente renglón
22         }
23         printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz
24         cont++;
25         i++; }
26     printf("\n");
27     return 0;
28 }
29

```

El while la variable “i” se declara al principio con las demás, y en el while le antesede la restricción y por ultimo esta el incremento esto casi al terminar el código.

Programa4c.c

```

9 #include <stdio.h> // aqui esta la libreria
10 // programa 4c.c
11 int main() {
12     int matriz[3][3] = {{1,2,3},{4,5,6},{7,8,9}};
13     int i, cont=0, *ap;
14     ap = *matriz; //Esta sentencia es análoga a: ap = &matriz[0][0];
15     printf("Imprimir Matriz\n");
16     i=0;
17     do //inicia el do
18     {
19         if (cont == 3) //Se imprimió un renglón y se hace un salto de línea, da salto de línea
            para no tener los numeros en una sola fila
20         {
21             printf("\n");
22             cont = 0; //Inicia conteo de elementos del siguiente renglón
23         }
24         printf("%d\t",*(ap+i)); //Se imprime el siguiente elemento de la matriz
25         cont++;
26         i++;
27     }
28     while (i<9); printf("\n"); // termina el do while
29     return 0;
30 }
31

```

La estructura do-while primer va a realizar las instrucciones donde se va encontrar el i++, después le antesedara la restricción, las variables siempre se declaran al inicio del programa

Tarea

1.-Realiza un programa que muestre tu nombre y número de cuenta haciendo uso de 2 [arreglos](#), emplear while y for

```
1  #include <stdio.h>
2  int main ()
3  {
4      char nombre[15]="Angel Santiago";
5      int num[] = {3,2,0,3,0,2,1,4,9};
6      int *pa, cont=0;
7      pa=&num[0];
8      char *pun;
9      pun=&nombre[0];
10     do{
11         printf("%c",*(pun+cont));
12         cont=cont+1;
13     }
14     while (cont!=15);
15     printf("\n ");
16
17     for(int i=0; i<=8; i++){
18
19         printf("%d",*(pa+i));
20
21     }
22     return 0;
23 }
```

C:\Users\luisr\Documents\ejers1.exe

Angel Santiago
320302149

Process exited after 0.01662 seconds with return value 0
Presione una tecla para continuar . . .

2.-Modifica el programa del ejercicio 1 utilizando apuntadores

ejers1.cpp [*] prog5ser.cpp matriz4ser.cpp progser3.cpp porgser2.cpp progser1.cpp matrz1.cpp mastrs2.cpp program3.cpp

```
1  #include <stdio.h>
2  int main ()
3  {
4      char nombre[15]="Angel Santiago";
5      int num[] = {3,2,0,3,0,2,1,4,9};
6      int cont=0;
7
8      do{
9          printf("%c",nombre[cont]);
10         cont=cont+1;
11     }
12     while (cont!=15);
13     printf("\n ");
14     for(int i=0; i<=9; i++){
15
16         printf("%d",num[i]);
17
18     }
19     return 0;
20 }
```

C:\Users\luisr\Documents\ejers1.exe

Angel Santiago
3203021490

Process exited after 0.01662 seconds with return value 0
Presione una tecla para continuar . . .

El tres no lo pude hacer

Conclusiones

Esta practica ya va subiendo de nivel mas y mas conforme pasamos el semestre y así creo que hemos aprendido biern lo ¿Qué se debía de estas mismas por lo que puedo decir que se ha cumplido el objetivo de la practica asui mismo creo que ya estamos preparados para dar el ultimo salto y así terminar el semestre con todos los connocimientos básicos de los fundamentos de progrfamacion.

Referencias • Online C++ Compiler - online editor. (s. f.). GDB online Debugger.
https://www.onlinegdb.com/online_c++_compiler • Laboratorios Salas A y B (unam.mx)