



Introducción a la Programación Practica 5

Medina Martinez Jonathan Jason 2023640061

15 de abril del 2023

Índice

1. Objetivo	3
2. Introducción	3
3. Desarrollo	4
3.1. Funciones	4
3.1.1. Programa1.c	4
3.1.2. Funciones.h	5
3.1.3. Ejecución	6
3.1.4. Programa2.c	7
3.1.5. Funciones.h	7
3.1.6. Ejecución	9
3.1.7. Programa3.c	11
3.1.8. Funciones.h	12
3.1.9. Ejecución	13
3.1.10. Programa32.c	14
3.1.11. Funciones.h	14
3.1.12. Ejecución	16
3.2. Recursión	17
3.2.1. Programa1.c	17
3.2.2. Ejecución	18
3.2.3. Programa2.c	19
3.2.4. Ejecución	20
3.2.5. Programa3.c	21
3.2.6. Ejecución	23
3.2.7. Programa4.c	24
3.2.8. Ejecución	24
3.2.9. Programa5.c	25
3.2.10. Ejecución	25
3.2.11. Programa6.c	26
3.2.12. Ejecución	27
3.2.13. Programa7.c	28
3.2.14. Ejecución	28
3.2.15. Programa.8	29
3.2.16. Ejecución	30
3.2.17. Programa9.c	31
3.2.18. Ejecución	31
3.2.19. Programa10.c	32
3.2.20. Ejecución	33
4. Conclusión	34

1. Objetivo

Desarrollar los programas hechos en las practicas anteriores aplicando funciones y desarrollo de aplicaciones utilizando funciones recursivas.

2. Introducción

En esta práctica se busca aplicar el concepto de funciones y desarrollo de aplicaciones utilizando funciones recursivas en la programación.

3. Desarrollo

3.1. Funciones

Tomando como base los programas realizados en la practica anterior, haga una función por cada operación que se realiza en cada uno, de tal forma que en la función principal unicamente se llamen las funciones creadas. Las funciones deberán estar en su propio archivo `.c` y se deberá incluir su archivo de encabezado `.h`.

3.1.1. Program1.c

```
1  /**
2  * @file program1.c
3  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
4  * @brief
5  * @version 0.1
6  * @date 2023-03-23
7  *
8  * @copyrigth GPLv3
9  *
10 */
11
12 #include <stdio.h>
13 #include <stdlib.h>
14 #include "funciones.h"
15
16 int main()
17 {
18     int n = 0;
19
20     char text1[] = "Ingrese el valor de n";
21     imprimir_texto(text1);
22     scanf("%d",&n);
23
24     int arr[n];
25     generar_arreglo_aleatorio(arr, n);
26
27     char text2[] = "Arreglo generado:";
28     imprimir_texto(text2);
29     imprimir_arreglo(arr, n);
30
31     ordenar_arreglo(arr, n);
32     char text3[] = "Arreglo ordenado de mayor a menor:";
33     imprimir_texto(text3);
34
35     imprimir_arreglo(arr, n);
36
37     return 0;
38 }
```

3.1.2. Funciones.h

```
1  /**
2  * @file funciones.h
3  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
4  * @brief
5  * @version 0.1
6  * @date 2023-04-17
7  *
8  * @copyrigth GPLv3
9  *
10 */
11
12 #include <stdio.h>
13 #include <stdlib.h>
14 #include <time.h>
15
16 /// @brief Genera un arreglo aleatorio de una dimension dado
17 /// @param arr El arreglo generada
18 /// @param n La dimension del arreglo
19 void generar_arreglo_aleatorio(int arr[], int n) {
20     srand(time(NULL));
21     for (int i = 0; i < n; i++)
22     {
23         arr[i] = (rand() % 500) + 1;
24     }
25 }
26
27
28 /// @brief Ordena un arreglo de mayor a menor
29 /// @param arr El arreglo a ordenar
30 /// @param n La dimension del arreglo
31 void ordenar_arreglo(int arr[], int n) {
32     int aux = 0, i = 0, j = 0;
33     for (i = 0; i < n; i++)
34     {
35         for (j = 0; j < n - 1 - i; j++)
36         {
37             if (arr[j] > arr[j + 1])
38             {
39                 aux = arr[j];
40                 arr[j] = arr[j + 1];
41                 arr[j + 1] = aux;
42             }
43         }
44     }
45 }
46
47
48 /// @brief Imprime un arreglo
49 /// @param arr El arreglo a imprimir
50 /// @param n La dimension del arreglo
51 void imprimir_arreglo(int arr[], int n) {
52     for (int i = 0; i < n; i++) {
53         printf("%d ", arr[i]);
54     }
55     printf("\n\n");
56 }
57
```

```
58  
59 /// @brief Imprime un texto  
60 /// @param text el texto a imprimir  
61 void imprimir_texto(char text[]) {  
62     printf("\n\n%s\n\n", text);  
63 }
```

3.1.3. Ejecución

```
1  
2  
3     Ingrese el valor de n  
4  
5     15  
6  
7  
8     Arreglo generado:  
9  
10    45 76 278 439 69 442 129 393 372 220 228 85 264 126 227  
11  
12  
13  
14    Arreglo ordenado de mayor a menor:  
15  
16    45 69 76 85 126 129 220 227 228 264 278 372 393 439 442
```

3.1.4. Programa2.c

```
1  /**
2  * @file programa2.c
3  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
4  * @brief
5  * @version 0.1
6  * @date 2023-03-24
7  *
8  * @copyright GPLv3
9  *
10 */
11
12 #include <stdio.h>
13 #include "funciones.h"
14
15 int main() {
16     int A[3][3], B[3][3], C[3][3];
17
18     char text1[] = "Ingrese los elementos de la matriz A:";
19     imprimir_texto(text1);
20     pedir_elementos(A);
21
22     char text2[] = "Ingrese los elementos de la matriz B:";
23     imprimir_texto(text2);
24     pedir_elementos(B);
25
26     char text3[] = "La suma de A + B es:";
27     imprimir_texto(text3);
28     suma_matrices(A, B, C);
29     imprimir_matriz(C);
30
31     char text4[] = "La resta de A - B es:";
32     imprimir_texto(text4);
33     resta_matrices(A, B, C);
34     imprimir_matriz(C);
35
36     char text5[] = "La multiplicacion de A x B es:";
37     imprimir_texto(text5);
38     multiplicacion_matrices(A, B, C);
39     imprimir_matriz(C);
40
41     return 0;
42 }
```

3.1.5. Funciones.h

```
1  /**
2  *
3  * @file funciones.h
4  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
5  * @brief
6  * @version 0.1
7  * @date 2023-04-19
8  *
9  * @copyright GPLv3
10 *
11 */
```

```

12
13 #include <stdio.h>
14
15
16 /// @brief Pide los elementos de la matriz
17 /// @param matriz La matriz en la que se guardan los elementos
18 void pedir_elementos(int matriz[3][3]) {
19     for (int i = 0; i < 3; i++) {
20         for (int j = 0; j < 3; j++) {
21             scanf("%d", &matriz[i][j]);
22         }
23     }
24 }
25
26
27 /// @brief Suma 2 matrices
28 /// @param matrizA Una de las matrices a sumar
29 /// @param matrizB Una de las matrices a sumar
30 /// @param matrizC La matriz resultante de la suma
31 void suma_matrices(int matrizA[3][3], int matrizB[3][3], int
matrizC[3][3]) {
32     for (int i = 0; i < 3; i++) {
33         for (int j = 0; j < 3; j++) {
34             matrizC[i][j] = matrizA[i][j] + matrizB[i][j];
35         }
36     }
37 }
38
39
40 /// @brief Resta 2 matrices
41 /// @param matrizA Una de las matrices a restar
42 /// @param matrizB Una de las matrices a restar
43 /// @param matrizC La matriz resultante de la resta
44 void resta_matrices(int matrizA[3][3], int matrizB[3][3], int
matrizC[3][3]) {
45     for (int i = 0; i < 3; i++) {
46         for (int j = 0; j < 3; j++) {
47             matrizC[i][j] = matrizA[i][j] - matrizB[i][j];
48         }
49     }
50 }
51
52
53 /// @brief Multiplica 2 matrices
54 /// @param matrizA Una de las matrices a multiplicar
55 /// @param matrizB Una de las matrices a multiplicar
56 /// @param matrizC La matriz resultante de la multiplicacion
57 void multiplicacion_matrices(int matrizA[3][3], int matrizB[3][3], int
matrizC[3][3]) {
58     for (int i = 0; i < 3; i++) {
59         for (int j = 0; j < 3; j++) {
60             matrizC[i][j] = 0;
61             for (int k = 0; k < 3; k++) {
62                 matrizC[i][j] += matrizA[i][k] * matrizB[k][j];
63             }
64         }
65     }
66 }
67

```



```

68
69 /// @brief Imprime una matriz
70 /// @param matrizC
71 void imprimir_matriz(int matrizC[3][3]) {
72     for (int i = 0; i < 3; i++) {
73         for (int j = 0; j < 3; j++) {
74             printf("%4d ", matrizC[i][j]);
75         }
76         printf("\n");
77     }
78     printf("\n");
79 }
80
81
82 /// @brief Imprime un texto
83 /// @param text el texto a imprimir
84 void imprimir_texto(char text[]) {
85     printf("\n\n\n%s\n\n", text);
86 }

```

3.1.6. Ejecución

```

1
2     Ingrese los elementos de la matriz A:
3
4     15
5     12
6     22
7     26
8     336
9     12
10    25
11    23
12    63
13
14
15
16    Ingrese los elementos de la matriz B:
17
18    12
19    32
20    15
21    25
22    36
23    21
24    25
25    15
26    26
27
28
29
30    La suma de A + B es:
31
32    27    44    37
33    51   372    33
34    50    38    89
35
36

```

```
37
38   La resta de A - B es:
39
40   3  -20   7
41   1  300  -9
42   0   8   37
43
44
45
46   La multiplicacion de A x B es:
47
48   1030 1242 1049
49   9012 13108 7758
50   2450 2573 2496
```

3.1.7. Programa3.c

```
1  /**
2  * @file programa3.c
3  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
4  * @brief
5  * @version 0.1
6  * @date 2023-03-24
7  *
8  * @copyrigth GPLv3
9  *
10 */
11
12 #include <stdio.h>
13 #include "funciones.h"
14
15 int main() {
16
17     float A[3][3] = {{2, -4, -3}, {1, 5, -5}, {4, 2, 67}};
18
19     float B[3] = {15, 5, 20};
20
21     float invA[3][3];
22
23     float X[3];
24
25     invertirMatrizA(A, invA);
26
27     generarvectorx(X, invA, B);
28
29     imprimirresultado(X);
30
31     return 0;
32 }
```

3.1.8. Funciones.h

```
1  /**
2  * @file funciones.h
3  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
4  * @brief
5  * @version 0.1
6  * @date 2023-04-19
7  *
8  * @copyrigth GPLv3
9  *
10 */
11
12 #include <stdio.h>
13
14 /// @brief Invierte la Matriz A
15 /// @param A Matriz a invertir
16 /// @param invA Matriz invertida
17 void invertirMatrizA(float A[3][3], float invA[3][3]){
18
19     float detA = A[0][0] * (A[1][1] * A[2][2] - A[2][1] * A[1][2])
20               - A[0][1] * (A[1][0] * A[2][2] - A[2][0] * A[1][2])
21               + A[0][2] * (A[1][0] * A[2][1] - A[2][0] * A[1][1]);
22
23     invA[0][0] = (A[1][1] * A[2][2] - A[2][1] * A[1][2]) / detA;
24     invA[0][1] = (A[0][2] * A[2][1] - A[2][2] * A[0][1]) / detA;
25     invA[0][2] = (A[0][1] * A[1][2] - A[1][1] * A[0][2]) / detA;
26
27     invA[1][0] = (A[1][2] * A[2][0] - A[2][2] * A[1][0]) / detA;
28     invA[1][1] = (A[0][0] * A[2][2] - A[2][0] * A[0][2]) / detA;
29     invA[1][2] = (A[0][2] * A[1][0] - A[1][2] * A[0][0]) / detA;
30
31     invA[2][0] = (A[1][0] * A[2][1] - A[2][0] * A[1][1]) / detA;
32     invA[2][1] = (A[0][1] * A[2][0] - A[2][1] * A[0][0]) / detA;
33     invA[2][2] = (A[0][0] * A[1][1] - A[1][0] * A[0][1]) / detA;
34
35 }
36
37 /// @brief Genera el vector de resultados X
38 /// @param X Vector de resultados
39 /// @param invA Matriz inversa de A
40 /// @param B Vector B
41 void generarvectorx(float X[3], float invA[3][3], float B[3]){
42
43     X[0] = invA[0][0] * B[0] + invA[0][1] * B[1] + invA[0][2] * B[2];
44     X[1] = invA[1][0] * B[0] + invA[1][1] * B[1] + invA[1][2] * B[2];
45     X[2] = invA[2][0] * B[0] + invA[2][1] * B[1] + invA[2][2] * B[2];
46
47 }
48
49 /// @brief Imprime el resultado en pantalla
50 /// @param X Vector de resultados
51 void imprimirresultado(float X[3]){
52
53     printf("La solucion es: \n");
54     printf("x = %f \n", X[0]);
55     printf("y = %f \n", X[1]);
56     printf("z = %f \n", X[2]);
57 }
```

3.1.9. Ejecución

```
1  La solucion es:  
2  x = 6.579670  
3  y = -0.398352  
4  z = -0.082418
```

3.1.10. Programa32.c

```
1  /**
2  * @file programa3.c
3  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
4  * @brief
5  * @version 0.1
6  * @date 2023-03-24
7  *
8  * @copyrigth GPLv3
9  *
10 */
11
12 #include <stdio.h>
13 #include "funciones.h"
14
15 int main() {
16
17     float A[3][3], B[3], invA[3][3], X[3];
18
19     printf("Calculadora de sistemas de ecuaciones lineales 3x3\n\n");
20     printf("Ejemplo de sistema de ecuaciones lineales 3x3:\n");
21     printf("A[0][0]x + A[0][1]y + A[0][2]z = B[1]\n");
22     printf("A[1][0]x + A[1][1]y + A[1][2]z = B[2]\n");
23     printf("A[2][0]x + A[2][1]y + A[2][2]z = B[3]\n\n");
24
25     matriz3x3(A);
26
27     vector(B);
28
29     matrizinversa(A, invA);
30
31     resultados(X, invA, B);
32
33     printf("\nLa solucion es: \n");
34     printf("x = %f \n", X[0]);
35     printf("y = %f \n", X[1]);
36     printf("z = %f \n", X[2]);
37     getc(stdin);
38
39     return 0;
40 }
```

3.1.11. Funciones.h

```
1  /**
2  *
3  * @file funciones.h
4  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
5  * @brief
6  * @version 0.1
7  * @date 2023-04-19
8  *
9  * @copyrigth GPLv3
10 *
11 */
12
13
```

```

14 #include <stdio.h>
15
16 /// @brief Crea una matriz 3x3 con los datos dados por el usuario
17 /// @param A La matriz generada
18 void matriz3x3(float A[3][3]){
19
20     printf("Ingrese los elementos de la matriz A: \n");
21     for(int i=0; i<3; i++) {
22         for(int j=0; j<3; j++) {
23             printf("A[%d][%d] = ", i, j);
24             scanf("%d", &A[i][j]);
25         }
26     }
27 }
28
29
30
31
32 /// @brief Crea un vector a partir de los datos dados por el usuario
33 /// @param B El vector generado
34 void vector(float B[3]){
35
36     printf("\nIngrese los elementos del vector B: \n");
37     for(int i=0; i<3; i++) {
38         printf("B[%d] = ", i);
39         scanf("%d", &B[i]);
40     }
41 }
42
43
44
45
46 /// @brief Saca la inversa de una matriz 3x3
47 /// @param A la matriz a invertir
48 /// @param invA La matriz invertida
49 /// @param detA El determinante de A solo se requiere inicializar en 0
50 void matrizinversa(float A[3][3], float invA[3][3]){
51
52     int detA = A[0][0] * (A[1][1] * A[2][2] - A[2][1] * A[1][2])
53     - A[0][1] * (A[1][0] * A[2][2] - A[2][0] * A[1][2])
54     + A[0][2] * (A[1][0] * A[2][1] - A[2][0] * A[1][1]);
55
56     invA[0][0] = (A[1][1] * A[2][2] - A[2][1] * A[1][2]) / detA;
57     invA[0][1] = (A[0][2] * A[2][1] - A[2][2] * A[0][1]) / detA;
58     invA[0][2] = (A[0][1] * A[1][2] - A[1][1] * A[0][2]) / detA;
59
60     invA[1][0] = (A[1][2] * A[2][0] - A[2][2] * A[1][0]) / detA;
61     invA[1][1] = (A[0][0] * A[2][2] - A[2][0] * A[0][2]) / detA;
62     invA[1][2] = (A[0][2] * A[1][0] - A[1][2] * A[0][0]) / detA;
63
64     invA[2][0] = (A[1][0] * A[2][1] - A[2][0] * A[1][1]) / detA;
65     invA[2][1] = (A[0][1] * A[2][0] - A[2][1] * A[0][0]) / detA;
66     invA[2][2] = (A[0][0] * A[1][1] - A[1][0] * A[0][1]) / detA;
67 }
68
69
70
71

```

```

72 /// @brief Calcula los valores de x1, x2 y x3 y los almacena en un
    vector
73 /// @param X El vector de resultados
74 /// @param invA la inversa de la matriz 3x3
75 /// @param B el vector B
76 void resultados(float X[3], float invA[3][3], float B[3]){
77
78     X[0] = invA[0][0] * B[0] + invA[0][1] * B[1] + invA[0][2] * B[2];
79     X[1] = invA[1][0] * B[0] + invA[1][1] * B[1] + invA[1][2] * B[2];
80     X[2] = invA[2][0] * B[0] + invA[2][1] * B[1] + invA[2][2] * B[2];
81
82 }

```

3.1.12. Ejecución

```

1  Calculadora de sistemas de ecuaciones lineales 3x3
2
3  Ejemplo de sistema de ecuaciones lineales 3x3:
4  A[0][0]x + A[0][1]y + A[0][2]z = B[1]
5  A[1][0]x + A[1][1]y + A[1][2]z = B[2]
6  A[2][0]x + A[2][1]y + A[2][2]z = B[3]
7
8  Ingrese los elementos de la matriz A:
9  A[0][0] = 12
10 A[0][1] = 65
11 A[0][2] = 23
12 A[1][0] = 23
13 A[1][1] = 14
14 A[1][2] = 26
15 A[2][0] = 23
16 A[2][1] = 36
17 A[2][2] = 14
18
19 Ingrese los elementos del vector B:
20 B[0] = 25
21 B[1] = 12
22 B[2] = 23
23
24 La solucion es:
25 x = 0.578800
26 y = 0.365204
27 z = -0.247125

```


3.2. Recursión

Realice los siguientes programas utilizando recursión.

3.2.1. Programa1.c

Solicite al usuario un numero entero n e imprima los números de 1 hasta n en incrementos de 3.
Ejemplo: n = 10, Salida: 1, 4, 7, 10

```
1  /**
2
3  * @file Programa1.c
4  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
5  * @brief
6  * @version 0.1
7  * @date 2023-04-17
8  *
9  * @copyrigth GPLv3
10 *
11 */
12
13 #include <stdio.h>
14
15
16 void imprimir(int m, int n);
17
18 int main() {
19     int n = 0;
20
21     printf("Ingresa un numero entero: ");
22
23     scanf("%d", &n);
24
25     printf("Los numeros en incrementos de 3 son: ");
26
27     imprimir(1, n);
28
29     printf("\n");
30
31     return 0;
32 }
33
34
35
36
37
38 /// @brief Funcion que imprime una secuencia de numeros
39 /// en incrementos de 3 desde uno hasta un numero dado
40 /// @param n El valor de final
41 /// @param m El inicio de la secuencia.
42 void imprimir(int m, int n) {
43
44     if (m > n) {
45
46         return;
47
48     }
49
50     printf("%d", m);
```

```
51
52     if (m != n) {
53
54         printf(", ");
55
56     }
57
58     imprimir(m + 3, n);
59
60 }
```

3.2.2. Ejecución

```
1     Ingresa un numero entero: 10
2     Los numeros en incrementos de 3 son: 1, 4, 7, 10
```

3.2.3. Programa2.c

Generar un arreglo de 20 números pseudoaleatorios entre 10 y 100, e imprimir sus elementos.

```
1  /**
2  *
3  * @file Programa2.c
4  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
5  * @brief
6  * @version 0.1
7  * @date 2023-04-22
8  *
9  * @copyrigth GPLv3
10 *
11 */
12
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <time.h>
16
17 #define MAX 20
18
19 void generar(int arr[], int n);
20
21 void imprimir(int arr[]);
22
23 int main() {
24     int arr[MAX];
25
26     srand(time(NULL));
27
28     generar(arr, 0);
29
30     imprimir(arr);
31
32     return 0;
33 }
34
35
36
37 /// @brief Funcion que genera un arreglo de 20 elementos
38 /// pseudoaleatorios de 10 a 100
39 /// @param arr el arreglo generado
40 /// @param n variable para almacenar la pocicion dentro del arreglo
41 void generar(int arr[], int n) {
42
43     if (n < MAX) {
44
45         arr[n] = rand() % 91 + 10;
46         generar(arr, n + 1);
47
48     }
49
50 }
51
52
53 /// @brief Programa que imprime un arreglo
54 /// @param arr el arreglo a imprimir
55 void imprimir(int arr[]) {
56     printf("Los elementos del arreglo son:\n");
```

```
57
58     for (int i = 0; i < MAX; i++) {
59
60         printf("%d ", arr[i]);
61
62     }
63
64     printf("\n");
65
66 }
```

3.2.4. Ejecución

```
1     Los elementos del arreglo son:
2     46 96 74 37 63 14 94 94 16 77 38 56 88 99 23 86 47 58 56 63
```

3.2.5. Programa3.c

Dado un arreglo de 20 números pseudoaleatorios entre 50 y 500, encontrar el elemento más grande y el más pequeño.

```
1  /**
2  * @file Programa3.c
3  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
4  * @brief
5  * @version 0.1
6  * @date 2023-04-22
7  *
8  * @copyrigth GPLv3
9  *
10 */
11
12 #include <stdio.h>
13 #include <stdlib.h>
14 #include <time.h>
15
16 #define ARR 20
17
18 #define MIN 50
19
20 #define MAX 500
21
22 void valor_maximo(int arr[], int n, int *maximo);
23
24 void valor_minimo(int arr[], int n, int *minimo);
25
26 int main() {
27
28     int arr[ARR];
29
30     int maximo = 0;
31     int minimo = 0;
32
33     srand(time(NULL));
34
35     for (int i = 0; i < ARR; i++) {
36
37         arr[i] = rand() % (MAX - MIN + 1) + MIN;
38
39     }
40
41     printf("Arreglo generado:\n");
42
43     for (int i = 0; i < ARR; i++) {
44
45         printf("%d ", arr[i]);
46
47     }
48     printf("\n");
49
50     valor_maximo(arr, ARR, &maximo);
51
52     valor_minimo(arr, ARR, &minimo);
53
54     printf("Elemento mas grande: %d\n", maximo);
```

```

55     printf("Elemento mas pequeno: %d\n", minimo);
56
57     return 0;
58
59 }
60
61
62 /// @brief funcion que obtiene el valor maximo de un arreglo
63 /// @param arr el arreglo
64 /// @param n el lugar a evaluar
65 /// @param maximo el valor maximo
66 void valor_maximo(int arr[], int n, int *maximo) {
67
68     if (n == 1) {
69
70         *maximo = arr[0];
71
72     } else {
73
74         valor_maximo(arr, n - 1, maximo);
75
76         if (arr[n - 1] > *maximo) {
77
78             *maximo = arr[n - 1];
79
80         }
81
82     }
83
84 }
85
86 /// @brief funcion que obtiene el valor minimo de un arreglo
87 /// @param arr el arreglo
88 /// @param n el lugar a evaluar
89 /// @param minimo el valor minimo
90 void valor_minimo(int arr[], int n, int *minimo) {
91
92     if (n == 1) {
93
94         *minimo = arr[0];
95
96     } else {
97
98         valor_minimo(arr, n - 1, minimo);
99
100        if (arr[n - 1] < *minimo) {
101
102            *minimo = arr[n - 1];
103
104        }
105
106    }
107
108 }

```

3.2.6. Ejecución

```
1  Arreglo generado:  
2  252 53 248 106 160 169 217 77 251 432 109 114 176 148 233 165 350  
   492 314 100  
3  Elemento mas grande: 492  
4  Elemento mas pequeno: 53
```

```
1  Arreglo generado:  
2  425 485 385 216 149 402 435 203 101 460 388 194 203 485 378 227 52  
   303 391 370  
3  Elemento mas grande: 485  
4  Elemento mas pequeno: 52
```

3.2.7. Programa4.c

Contar el número de dígitos de un numero entero proporcionado por el usuario. Ejemplo: Entrada 56790, Salida: 5

```
1  /**
2
3  * @file Programa4.c
4  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
5  * @brief
6  * @version 0.1
7  * @date 2023-04-22
8  *
9  * @copyright GPLv3
10 *
11 */
12
13 #include <stdio.h>
14
15 void contar(int n, int *suma);
16
17 int main() {
18     int n = 0, suma = 0;
19
20     printf("Ingrese un numero entero: ");
21     scanf("%d", &n);
22
23     contar(n, &suma);
24
25     printf("El numero %d tiene %d digitos\n", n, suma);
26
27     return 0;
28 }
29
30
31
32 /// @brief Funcion que cuenta los digitos de un numero entero
33 /// @param n El numero entero
34 /// @param suma variable para almacenar la cuenta
35 void contar(int n, int *suma) {
36
37     if (n != 0) {
38
39         (*suma)++;
40
41         contar(n / 10, suma);
42
43     }
44
45 }
```

3.2.8. Ejecución

```
1  Ingrese un numero entero: 56790
2  El numero 56790 tiene 5 digitos
```

```
1  Ingrese un numero entero: 15152151512
2  El numero -2027717672 tiene 10 digitos
```


3.2.9. Programa5.c

Sumar los dígitos de un numero entero proporcionado por el usuario. Ejemplo: $256 = 2 + 5 + 6$

```
1  /**
2  * @file Programa5.c
3  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
4  * @brief
5  * @version 0.1
6  * @date 2023-04-22
7  *
8  * @copyrigth GPLv3
9  *
10 */
11
12 #include <stdio.h>
13
14 void suma(int n, int *sum);
15
16 int main() {
17     int n = 0, sum = 0;
18
19     printf("Ingrese un numero entero positivo: ");
20     scanf("%d", &n);
21     suma(n, &sum);
22     printf("La suma de los digitos de %d es %d.\n", n, sum);
23
24     return 0;
25 }
26
27
28
29 /// @brief suma digito a digito un numero entero
30 /// @param n el numero entero
31 /// @param sum la suma
32 void suma(int n, int *sum) {
33
34     if (n == 0) {
35
36         return;
37
38     } else {
39
40         *sum += n % 10;
41
42         suma(n / 10, sum);
43
44     }
45 }
46 }
```

3.2.10. Ejecución

```
1  Ingrese un numero entero positivo: 256
2  La suma de los digitos de 256 es 13.
```

```
1  Ingrese un numero entero positivo: 1455823
2  La suma de los digitos de 1455823 es 28.
```

3.2.11. Programa6.c

Genere un arreglo con 10,000 números pseudoaleatorios, cada uno entre 1 y 100,000. Posteriormente, cree una función recursiva que ordene el arreglo anterior en orden descendente, aplicando el algoritmo de la burbuja.

```
1  /**
2  * @file Programa6.c
3  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
4  * @brief
5  * @version 0.1
6  * @date 2023-04-22
7  *
8  *
9  * @copyrigth GPLv3
10 *
11 */
12
13 #include <stdio.h>
14 #include <stdlib.h>
15 #include <time.h>
16
17 #define TAM 10000
18
19 void burbuja(int arr[], int tam);
20
21 void imprimir(int arr[]);
22
23 int main() {
24     int arr[TAM];
25
26     srand(time(NULL));
27
28     for (int i = 0; i < TAM; i++) {
29         arr[i] = rand() % 100000 + 1;
30     }
31
32     printf("Arreglo generado:\n");
33
34     imprimir(arr);
35
36     burbuja(arr, TAM);
37
38     printf("Arreglo ordenado:\n");
39
40     imprimir(arr);
41
42     return 0;
43 }
44
45
46
47
48 /// @brief Ordena un arreglo en orden decendente
49 /// @param arr el arreglo
50 /// @param tam dimension del arreglo
51 void burbuja(int arr[], int tam) {
52     if (tam == 1) {
53
```

```

54         return;
55     }
56
57     for (int i = 0; i < tam - 1; i++) {
58         if (arr[i] < arr[i+1]) {
59             int temp = arr[i];
60             arr[i] = arr[i+1];
61             arr[i+1] = temp;
62         }
63     }
64     burbuja(arr, tam - 1);
65 }
66
67 /// @brief Imprime un arreglo en pantalla
68 /// @param arr El arreglo
69 void imprimir(int arr[]){
70     for (int i = 0; i < TAM; i++) {
71         printf("%d ", arr[i]);
72     }
73     printf("\n");
74 }

```

3.2.12. Ejecución

Por insuficiencia de espacio reducimos el tamaño del arreglo a 50

```

1  Arreglo generado:
2  14052 8497 24192 2531 5625 24169 14514 25087 1231 10919 3830 7795
   16618 5169 8717 31315 24864 15951 4268 16333 21491 13906 26737
   25939 6174 6613 17178 1533 25454 29251 17401 14632 11219 27033
   2467 3428 18103 15469 26601 10893 4750 3376 32265 24375 24239
   31920 32037 7503 1378 11232
3  Arreglo ordenado:
4  32265 32037 31920 31315 29251 27033 26737 26601 25939 25454 25087
   24864 24375 24239 24192 24169 21491 18103 17401 17178 16618
   16333 15951 15469 14632 14514 14052 13906 11232 11219 10919
   10893 8717 8497 7795 7503 6613 6174 5625 5169 4750 4268 3830
   3428 3376 2531 2467 1533 1378 1231

```

3.2.13. Programa7.c

```
1  /**
2  * @file Programa7.c
3  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
4  * @brief
5  * @version 0.1
6  * @date 2023-04-22
7  *
8  * @copyrigth GPLv3
9  *
10 */
11
12 #include <stdio.h>
13
14 void hailstone(int n);
15
16 int main() {
17     int n;
18     printf("Ingrese un numero: ");
19     scanf("%d", &n);
20     hailstone(n);
21     return 0;
22 }
23
24 /// @brief Funcion que genera la serie de hailstone a partir de n
25 /// @param n Un numero dado
26 void hailstone(int n) {
27
28     printf("%d ", n);
29
30     if (n == 1) {
31
32         return;
33
34     } else if (n % 2 == 0) {
35
36         hailstone(n / 2);
37
38     } else {
39
40         hailstone(3 * n + 1);
41
42     }
43
44 }
```

3.2.14. Ejecución

```
1  Ingrese un numero: 15
2  15 46 23 70 35 106 53 160 80 40 20 10 5 16 8 4 2 1
```

```
1  Ingrese un numero: 251
2  251 754 377 1132 566 283 850 425 1276 638 319 958 479 1438 719
   2158 1079 3238 1619 4858 2429 7288 3644 1822 911 2734 1367 4102
   2051 6154 3077 9232 4616 2308 1154 577 1732 866 433 1300 650
   325 976 488 244 122 61 184 92 46 23 70 35 106 53 160 80 40 20
   10 5 16 8 4 2 1
```

3.2.15. Programa.8

Calcule la suma de los m primeros términos de la serie de Leibniz, y converge a $\pi/4$. Ejecute el programa para $m = 10$ y $m = 500$. Compare posteriormente estos resultados con el valor exacto $\pi/4$.

```
1  /**
2
3  * @file Programa8.c
4  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
5  * @brief
6  * @version 0.1
7  * @date 2023-04-22
8  *
9  * @copyright GPLv3
10 *
11 */
12
13 #include <stdio.h>
14 #include <math.h>
15
16 #define PI 3.14159265358979323846
17
18 double leibniz(int m, double acum);
19
20 int main() {
21     int m = 0;
22
23     printf("Ingrese el valor de m: ");
24     scanf("%d", &m);
25
26     double sum = leibniz(m, 0);
27
28     if (sum < 0) {
29         sum = sum * -1;
30     }
31
32     double picuartos = PI / 4;
33
34     printf("\nLa suma de los primeros %d terminos de la serie de
35     Leibniz es: %f\n", m, sum);
36     printf("El valor de pi/4 es: %f\n", picuartos);
37     return 0;
38 }
39
40 /// @brief calcula la suma de los resultados de la serie de Leibniz
41 /// @param m el numero de repeticiones
42 /// @param acum la suma
43 /// @return regresa el valor de acum
44 double leibniz(int m, double acum) {
45     if (m == 0) {
46         return acum;
47     }
48     double termino = (pow(-1, m)) / (2 * m + 1);
49     return leibniz(m - 1, acum + termino);
50 }
```

3.2.16. Ejecución

```
1   Ingrese el valor de m: 10
2
3   La suma de los primeros 10 terminos de la serie de Leibniz es:
    0.191921
4   El valor de pi/4 es: 0.785398
```

```
1   Ingrese el valor de m: 500
2
3   La suma de los primeros 500 terminos de la serie de Leibniz es:
    0.214103
4   El valor de pi/4 es: 0.785398
```

3.2.17. Programa9.c

Una secuencia de Fibonacci esta compuesta de elementos creados al sumar los dos elementos previos. La secuencia de Fibonacci mas simples comienza con 1, 1 y procede del modo siguiente: 1, 1, 2, 3, 5, 8, 13, . . . Sin embargo, una secuencia de Fibonacci se puede crear con cualesquiera dos números iniciales. Escriba un programa que solicite al usuario ingresar los dos primeros números en una secuencia de Fibonacci y el numero total de elementos solicitados para la secuencia. Encuentre la secuencia utilizando recursión.

```
1  /**
2  * @file Programa9.c
3  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
4  * @brief
5  * @version 0.1
6  * @date 2023-04-22
7  *
8  * @copyrigth GPLv3
9  *
10 */
11
12 #include <stdio.h>
13
14 void fibonacci(int num1, int num2, int total);
15
16 int main() {
17     int num1, num2, total;
18     printf("Ingrese el primer numero: ");
19     scanf("%d", &num1);
20
21     printf("Ingrese el segundo numero: ");
22     scanf("%d", &num2);
23     printf("Ingrese el numero de elementos: ");
24     scanf("%d", &total);
25     printf("%d %d ", num1, num2);
26     fibonacci(num1, num2, total-2);
27     return 0;
28 }
29
30 /// @brief Funcion que genera una secuencia de fibonacci
31 /// @param num1 el primer numero de la secuencia
32 /// @param num2 el segundo numero de la secuencia
33 /// @param total la cantidad de elementos a calcular
34 void fibonacci(int num1, int num2, int total) {
35     if (total == 0) {
36         return;
37     }
38     int next = num1 + num2;
39     printf("%d ", next);
40     fibonacci(num2, next, total-1);
41 }
```

3.2.18. Ejecución

```
1  Ingrese el primer numero: 10
2  Ingrese el segundo numero: 15
3  Ingrese el numero de elementos: 10
4  10 15 25 40 65 105 170 275 445 720
```

3.2.19. Programa10.c

Escriba un programa que acepte los primeros dos números de una secuencia de Fibonacci como entrada de usuario y luego calcule valores adicionales en la secuencia hasta que la diferencia entre las razones de valores adyacentes sea de 0.001.

```
1  /**
2  *
3  * @file Programa10.c
4  * @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
5  * @brief
6  * @version 0.1
7  * @date 2023-04-22
8  *
9  * @copyrigth GPLv3
10 *
11 */
12
13 #include <stdio.h>
14 #include <math.h>
15
16 double aureo(int e1, int e2, double rac, double phi);
17
18 int main() {
19     double phi = (1 + sqrt(5)) / 2;
20     int e1, e2;
21
22     printf("Ingrese el Primer digito: ");
23     scanf("%d", &e1);
24
25     printf("Ingrese el Segundo digito: ");
26     scanf("%d", &e2);
27
28     printf("%d, %d, ", e1, e2);
29
30     double rac = fabs((double)e2 / (double)e1);
31     double result = aureo(e1, e2, rac, phi);
32
33     printf("\nEl valor de Phi es: %f", phi);
34
35     printf("\nEl valor de %d / %d es: %f", e2, e1, result);
36
37     return 0;
38 }
39
40 /// @brief funcion que calcula el numero aure
41 /// @param e1 el primer valor
42 /// @param e2 el segundo valor
43 /// @param rac almacén
44 /// @param phi el valor de phi
45 double aureo(int e1, int e2, double rac, double phi) {
46     int e3 = e1 + e2;
47
48     printf("%d, ", e3);
49
50     double new_rac = fabs((double)e3 / (double)e2);
51
52     if (fabs(rac - new_rac) < 0.001) {
53         return new_rac;
```



```
54     } else {  
55         return aureo(e2, e3, new_rac, phi);  
56     }  
57 }
```

3.2.20. Ejecución

```
1  Ingrese el Primer digito: 10  
2  Ingrese el Segundo digito: 12  
3  10, 12, 22, 34, 56, 90, 146, 236, 382, 618,  
4  El valor de Phi es: 1.618034  
5  El valor de 12 / 10 es: 1.617801
```

4. Conclusión

En conclusión, esta práctica es una continuación de la anterior y busca poner en práctica el uso de funciones y recursión en la programación. Los programas realizados en esta práctica serán de gran utilidad para el desarrollo de habilidades en el manejo de funciones y recursión en la programación.