Introducción a la Programación Practica 5

Medina Martinez Jonathan Jason 2023640061

15 de abril del 2023

Índice

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. Programa1.c

```c
/**
* @file programa1.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-03-23
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>
#include <stdlib.h>
#include "funciones.h"

int main()
{
    int n = 0;

    char text1[] = "Ingrese el valor de n";
    imprimir_texto(text1);
    scanf("%d",&n);

    int arr[n];
    generar_arreglo_aleatorio(arr, n);

    char text2[] = "Arreglo generado:";
    imprimir_texto(text2);
    imprimir_arreglo(arr, n);

    ordenar_arreglo(arr, n);
    char text3[] = "Arreglo ordenado de mayor a menor:";
    imprimir_texto(text3);

    imprimir_arreglo(arr, n);

    return 0;
}
```

3.1.2. Funciones.h

```c
/**
* @file funciones.h
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-17
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

/// @brief Genera un arreglo aleatorio de una dimension dado
/// @param arr El arreglo generada
/// @param n La dimension del arreglo
void generar_arreglo_aleatorio(int arr[], int n) {
    srand(time(NULL));
    for (int i = 0; i < n; i++)
    {
        arr[i] = (rand() % 500) + 1;
    }
}


/// @brief Ordena un arreglo de mayor a menor
/// @param arr El arreglo a ordenar
/// @param n La dimension del arreglo
void ordenar_arreglo(int arr[], int n) {
    int aux = 0, i = 0, j = 0;
    for (i = 0; i < n; i++)
    {
        for (j = 0; j < n - 1 -i; j++)
        {
            if (arr[j] > arr[j + 1])
            {
                aux = arr[j];
                arr[j] = arr[j + 1];
                arr[j + 1] = aux;
            }
        }
    }
}


/// @brief Imprime un arreglo
/// @param arr El arreglo a imprimir
/// @param n La dimension del arreglo
void imprimir_arreglo(int arr[], int n) {
    for (int i = 0; i < n; i++) {
        printf("%d ", arr[i]);
    }
    printf("\n\n");
}

```

```c

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

3.1.3. Ejecución

```
    Ingrese el valor de n

    15


    Arreglo generado:

    45 76 278 439 69 442 129 393 372 220 228 85 264 126 227


    Arreglo ordenado de mayor a menor:

    45 69 76 85 126 129 220 227 228 264 278 372 393 439 442
```

3.1.4. Programa2.c

```c
/**
* @file programa2.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-03-24
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>
#include "funciones.h"

int main() {
    int A[3][3], B[3][3], C[3][3];

    char text1[] = "Ingrese los elementos de la matriz A:";
    imprimir_texto(text1);
    pedir_elementos(A);

    char text2[] = "Ingrese los elementos de la matriz B:";
    imprimir_texto(text2);
    pedir_elementos(B);

    char text3[] = "La suma de A + B es:";
    imprimir_texto(text3);
    suma_matrices(A, B, C);
    imprimir_matriz(C);

    char text4[] = "La resta de A - B es:";
    imprimir_texto(text4);
    resta_matrices(A, B, C);
    imprimir_matriz(C);

    char text5[] = "La multiplicacion de A x B es:";
    imprimir_texto(text5);
    multiplicacion_matrices(A, B, C);
    imprimir_matriz(C);

    return 0;
}
```

3.1.5. Funciones.h

```c

/**
* @file funciones.h
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-19
*
* @copyrigth GPlv3
*
*/
```

```c

#include <stdio.h>


/// @brief Pide los elementos de la matriz
/// @param matriz La matriz en la que se guardan los elementos
void pedir_elementos(int matriz[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            scanf("%d", &matriz[i][j]);
        }
    }
}


/// @brief Suma 2 matrices
/// @param matrizA Una de las matrices a sumar
/// @param matrizB Una de las matrices a sumar
/// @param matrizC La matriz resultante de la suma
void suma_matrices(int matrizA[3][3], int matrizB[3][3], int matrizC[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            matrizC[i][j] = matrizA[i][j] + matrizB[i][j];
        }
    }
}


/// @brief Resta 2 matrices
/// @param matrizA Una de las matrices a restar
/// @param matrizB Una de las matrices a restar
/// @param matrizC La matriz resultante de la resta
void resta_matrices(int matrizA[3][3], int matrizB[3][3], int matrizC[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            matrizC[i][j] = matrizA[i][j] - matrizB[i][j];
        }
    }
}


/// @brief Multiplica 2 matrices
/// @param matrizA Una de las matrices a multiplicar
/// @param matrizB Una de las matrices a multiplicar
/// @param matrizC La matriz resultante de la multiplicacion
void multiplicacion_matrices(int matrizA[3][3], int matrizB[3][3], int matrizC[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            matrizC[i][j] = 0;
            for (int k = 0; k < 3; k++) {
                matrizC[i][j] += matrizA[i][k] * matrizB[k][j];
            }
        }
    }
}

```

```c

/// @brief Imprime una matriz
/// @param matrizC
void imprimir_matriz(int matrizC[3][3]) {
    for (int i = 0; i < 3; i++) {
        for (int j = 0; j < 3; j++) {
            printf("%4d ", matrizC[i][j]);
        }
        printf("\n");
    }
    printf("\n");
}


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

3.1.6. Ejecución

```
    Ingrese los elementos de la matriz A:

    15
    12
    22
    26
    336
    12
    25
    23
    63



    Ingrese los elementos de la matriz B:

    12
    32
    15
    25
    36
    21
    25
    15
    26



    La suma de A + B es:

    27    44    37
    51   372    33
    50    38    89


```

```
La resta de A - B es:

3  -20    7
1  300   -9
0    8   37



La multiplicacion de A x B es:

1030 1242 1049
9012 13108 7758
2450 2573 2496
```

3.1.7. Programa3.c

```c
/**
* @file programa3.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-03-24
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>
#include "funciones.h"

int main() {

    float A[3][3] = {{2, -4, -3}, {1, 5, -5}, {4, 2, 67}};

    float B[3] = {15, 5, 20};

    float invA[3][3];

    float X[3];

    invertirMatrizA(A, invA);

    generarvectorx(X, invA, B);

    imprimirresultado(X);

    return 0;
}
```

3.1.8. Funciones.h

```c
/**
* @file funciones.h
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-19
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>

/// @brief Invierte la Matriz A
/// @param A Matriz a invertir
/// @param invA Matriz invertida
void invertirMatrizA(float A[3][3], float invA[3][3]){

    float detA = A[0][0] * (A[1][1] * A[2][2] - A[2][1] * A[1][2])
               - A[0][1] * (A[1][0] * A[2][2] - A[2][0] * A[1][2])
               + A[0][2] * (A[1][0] * A[2][1] - A[2][0] * A[1][1]);

    invA[0][0] = (A[1][1] * A[2][2] - A[2][1] * A[1][2]) / detA;
    invA[0][1] = (A[0][2] * A[2][1] - A[2][2] * A[0][1]) / detA;
    invA[0][2] = (A[0][1] * A[1][2] - A[1][1] * A[0][2]) / detA;

    invA[1][0] = (A[1][2] * A[2][0] - A[2][2] * A[1][0]) / detA;
    invA[1][1] = (A[0][0] * A[2][2] - A[2][0] * A[0][2]) / detA;
    invA[1][2] = (A[0][2] * A[1][0] - A[1][2] * A[0][0]) / detA;

    invA[2][0] = (A[1][0] * A[2][1] - A[2][0] * A[1][1]) / detA;
    invA[2][1] = (A[0][1] * A[2][0] - A[2][1] * A[0][0]) / detA;
    invA[2][2] = (A[0][0] * A[1][1] - A[1][0] * A[0][1]) / detA;

}

/// @brief Genera el vector de resultados X
/// @param X Vector de resultados
/// @param invA Matriz inversa de A
/// @param B Vector B
void generarvectorx(float X[3], float invA[3][3], float B[3]){

    X[0] = invA[0][0] * B[0] + invA[0][1] * B[1] + invA[0][2] * B[2];
    X[1] = invA[1][0] * B[0] + invA[1][1] * B[1] + invA[1][2] * B[2];
    X[2] = invA[2][0] * B[0] + invA[2][1] * B[1] + invA[2][2] * B[2];

}

/// @brief Imprime el resultado en pantalla
/// @param X Vector de resultados
void imprimirresultado(float X[3]){

    printf("La solucion es: \n");
    printf("x = %f \n", X[0]);
    printf("y = %f \n", X[1]);
    printf("z = %f \n", X[2]);
}
```

3.1.9. Ejecución

```
La solucion es:
x = 6.579670
y = -0.398352
z = -0.082418
```

3.1.10. Programa32.c

```c
/**
* @file programa3.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-03-24
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>
#include "funciones.h"

int main() {

    float A[3][3],  B[3], invA[3][3], X[3];

    printf("Calculadora de sistemas de ecuaciones lineales 3x3\n\n");
    printf("Ejemplo de sistema de ecuaciones lineales 3x3:\n");
    printf("A[0][0]x + A[0][1]y + A[0][2]z = B[1]\n");
    printf("A[1][0]x + A[1][1]y + A[1][2]z = B[2]\n");
    printf("A[2][0]x + A[2][1]y + A[2][2]z = B[3]\n\n");

    matriz3x3(A);

    vector(B);

    matrizinversa(A, invA);

    resultados(X, invA, B);

    printf("\nLa solucion es: \n");
    printf("x = %f \n", X[0]);
    printf("y = %f \n", X[1]);
    printf("z = %f \n", X[2]);
    getc(stdin);

    return 0;
}
```

3.1.11. Funciones.h

```c

/**
* @file funciones.h
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-19
*
* @copyrigth GPlv3
*
*/


```

```c
#include <stdio.h>

/// @brief Crea una matriz 3x3 con los datos dados por el usuario
/// @param A La matriz generada
void matriz3x3(float A[3][3]){

    printf("Ingrese los elementos de la matriz A: \n");
    for(int i=0; i<3; i++) {
        for(int j=0; j<3; j++) {
            printf("A[%d][%d] = ", i, j);
            scanf("%d", &A[i][j]);
        }
    }

}



/// @brief Crea un vector a partir de los datos dados por el usuario
/// @param B El vector generado
void vector(float B[3]){

    printf("\nIngrese los elementos del vector B: \n");
    for(int i=0; i<3; i++) {
        printf("B[%d] = ", i);
        scanf("%d", &B[i]);
    }

}



/// @brief Saca la inversa de una matriz 3x3
/// @param A la matriz a invertir
/// @param invA La matriz invertida
/// @param detA El determinante de A solo se requiere inicializar en 0
void matrizinversa(float A[3][3], float invA[3][3]){

    int detA = A[0][0] * (A[1][1] * A[2][2] - A[2][1] * A[1][2])
    - A[0][1] * (A[1][0] * A[2][2] - A[2][0] * A[1][2])
    + A[0][2] * (A[1][0] * A[2][1] - A[2][0] * A[1][1]);

    invA[0][0] = (A[1][1] * A[2][2] - A[2][1] * A[1][2]) / detA;
    invA[0][1] = (A[0][2] * A[2][1] - A[2][2] * A[0][1]) / detA;
    invA[0][2] = (A[0][1] * A[1][2] - A[1][1] * A[0][2]) / detA;

    invA[1][0] = (A[1][2] * A[2][0] - A[2][2] * A[1][0]) / detA;
    invA[1][1] = (A[0][0] * A[2][2] - A[2][0] * A[0][2]) / detA;
    invA[1][2] = (A[0][2] * A[1][0] - A[1][2] * A[0][0]) / detA;

    invA[2][0] = (A[1][0] * A[2][1] - A[2][0] * A[1][1]) / detA;
    invA[2][1] = (A[0][1] * A[2][0] - A[2][1] * A[0][0]) / detA;
    invA[2][2] = (A[0][0] * A[1][1] - A[1][0] * A[0][1]) / detA;

}
```

```
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

```c

/**
* @file Programa1.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-17
*
* @copyrigth GPlv3
*
*/


#include <stdio.h>


void imprimir(int m, int n);

int main() {

    int n = 0;

    printf("Ingresa un numero entero: ");

    scanf("%d", &n);

    printf("Los numeros en incrementos de 3 son: ");

    imprimir(1, n);

    printf("\n");

    return 0;

}


/// @brief Funcion que imprime una secuencia de numeros
/// en incrementos de 3 desde uno hasta un numero dado
/// @param n El valor de final
/// @param m El inicio de la secuencia.
void imprimir(int m, int n) {

    if (m > n) {

        return;

    }

    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.

```c

/**
* @file Programa2.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-22
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define MAX 20

void generar(int arr[], int n);

void imprimir(int arr[]);

int main() {

    int arr[MAX];

    srand(time(NULL));

    generar(arr, 0);

    imprimir(arr);

    return 0;

}

/// @brief Funcion que genera un arreglo de 20 elementos
/// pseudoaleatorios de 10 a 100
/// @param arr el arreglo generado
/// @param n variable para almacenar la pocicion dentro del arreglo
void generar(int arr[], int n) {

    if (n < MAX) {

        arr[n] = rand() % 91 + 10;
        generar(arr, n + 1);

    }

}

/// @brief Programa que imprime un arreglo
/// @param arr el arreglo a imprimir
void imprimir(int arr[]) {

    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.

```c
/**
* @file Programa3.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-22
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define ARR 20

#define MIN 50

#define MAX 500

void valor_maximo(int arr[], int n, int *maximo);

void valor_minimo(int arr[], int n, int *minimo);

int main() {

    int arr[ARR];

    int maximo = 0;
    int minimo = 0;

    srand(time(NULL));

    for (int i = 0; i < ARR; i++) {

        arr[i] = rand() % (MAX - MIN + 1) + MIN;

    }

    printf("Arreglo generado:\n");

    for (int i = 0; i < ARR; i++) {

        printf("%d ", arr[i]);

    }
    printf("\n");

    valor_maximo(arr, ARR, &maximo);

    valor_minimo(arr, ARR, &minimo);

    printf("Elemento mas grande: %d\n", maximo);
```

```
55
56      printf("Elemento mas pequeno: %d\n", minimo);
57
58      return 0;
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

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

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

```c

/**
* @file Programa4.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-22
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>

void contar(int n, int *suma);

int main() {

    int n = 0, suma = 0;

    printf("Ingrese un numero entero: ");
    scanf("%d", &n);

    contar(n, &suma);

    printf("El numero %d tiene %d digitos\n", n, suma);

    return 0;

}

/// @brief Funcion que cuenta los digitos de un numero entero
/// @param n El numero entero
/// @param suma variable para almacenar la cuenta
void contar(int n, int *suma) {

    if (n != 0) {

        (*suma)++;

        contar(n / 10, suma);

    }

}
```

3.2.8. Ejecución

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

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

3.2.9. Programa5.c

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

```c
/**
* @file Programa5.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-22
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>

void suma(int n, int *sum);

int main() {

    int n = 0, sum = 0;

    printf("Ingrese un numero entero positivo: ");
    scanf("%d", &n);
    suma(n, &sum);
    printf("La suma de los digitos de %d es %d.\n", n, sum);

    return 0;

}

/// @brief suma digito a digito un numero entero
/// @param n el numero entero
/// @param sum la suma
void suma(int n, int *sum) {

    if (n == 0) {

        return;

    } else {

        *sum += n % 10;

        suma(n / 10, sum);

    }

}
```

3.2.10. Ejecución

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

```
    Ingrese un numero entero positivo: 1455823
    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.

```c

/**
* @file Programa6.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-22
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>
#include <stdlib.h>
#include <time.h>

#define TAM 10000

void burbuja(int arr[], int tam);

void imprimir(int arr[]);

int main() {

    int arr[TAM];

    srand(time(NULL));

    for (int i = 0; i < TAM; i++) {

        arr[i] = rand() % 100000 + 1;

    }

    printf("Arreglo generado:\n");

    imprimir(arr);

    burbuja(arr, TAM);

    printf("Arreglo ordenado:\n");

    imprimir(arr);

    return 0;
}

/// @brief Ordena un areglo en orden decendente
/// @param arr el arreglo
/// @param tam dimension del arreglo
void burbuja(int arr[], int tam) {

    if (tam == 1) {
```

```
54
55          return;
56
57      }
58
59      for (int i = 0; i < tam - 1; i++) {
60
61          if (arr[i] < arr[i+1]) {
62
63              int temp = arr[i];
64
65              arr[i] = arr[i+1];
66
67              arr[i+1] = temp;
68
69          }
70
71      }
72
73      burbuja(arr, tam - 1);
74
75  }
76
77  /// @brief Imprime un arreglo en pantalla
78  /// @param arr El arreglo
79  void imprimir(int arr[]){
80
81      for (int i = 0; i < TAM; i++) {
82
83          printf("%d ", arr[i]);
84
85      }
86
87      printf("\n");
88
89  }
```

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

```c
/**
* @file Programa7.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-22
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>

void hailstone(int n);

int main() {
    int n;
    printf("Ingrese un numero: ");
    scanf("%d", &n);
    hailstone(n);
    return 0;
}

/// @brief Funcion que genera la serie de hailstone a partir de n
/// @param n Un numero dado
void hailstone(int n) {

    printf("%d ", n);

    if (n == 1) {

        return;

    } else if (n % 2 == 0) {

        hailstone(n / 2);

    } else {

        hailstone(3 * n + 1);

    }

}
```

3.2.14. Ejecución

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

```
    Ingrese un numero: 251
    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 $\pi/4$.

```c

/**
* @file Programa8.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-22
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>
#include <math.h>

#define PI 3.14159265358979323846

double leibniz(int m, double acum);

int main() {
    int m = 0;

    printf("Ingrese el valor de m: ");
    scanf("%d", &m);

    double sum = leibniz(m, 0);

    if (sum < 0) {
        sum = sum * -1;
    }

    double picuartos = PI / 4;

    printf("\nLa suma de los primeros %d terminos de la serie de
       Leibniz es: %f\n", m, sum);
    printf("El valor de pi/4 es: %f\n", picuartos);
    return 0;
}

/// @brief calcula la suma de los resultados de la serie de Leibniz
/// @param m el numero de repeticiones
/// @param acum la suma
/// @return regresa el valor de acum
double leibniz(int m, double acum) {
    if (m == 0) {
        return acum;
    }
    double termino = (pow(-1, m)) / (2 * m + 1);
    return leibniz(m - 1, acum + termino);
}
```

3.2.16. Ejecución

```
Ingrese el valor de m: 10

La suma de los primeros 10 terminos de la serie de Leibniz es: 0.191921
El valor de pi/4 es: 0.785398
```

```
Ingrese el valor de m: 500

La suma de los primeros 500 terminos de la serie de Leibniz es: 0.214103
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.

```c
/**
* @file Programa9.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-22
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>

void fibonacci(int num1, int num2, int total);

int main() {
    int num1, num2, total;
    printf("Ingrese el primer numero: ");
    scanf("%d", &num1);

    printf("Ingrese el segundo numero: ");
    scanf("%d", &num2);
    printf("Ingrese el numero de elementos: ");
    scanf("%d", &total);
    printf("%d %d ", num1, num2);
    fibonacci(num1, num2, total-2);
    return 0;
}

/// @brief Funcion que genera una secuencia de fibonacci
/// @param num1 el primer numero de la secuencia
/// @param num2 el segundo numero de la secuencia
/// @param total la cantidad de elementos a calcular
void fibonacci(int num1, int num2, int total) {
    if (total == 0) {
        return;
    }
    int next = num1 + num2;
    printf("%d ", next);
    fibonacci(num2, next, total-1);
}
```

3.2.18. Ejecución

```
    Ingrese el primer numero: 10
    Ingrese el segundo numero: 15
    Ingrese el numero de elementos: 10
    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.

```c

/**
* @file Programa10.c
* @author Medina Martinez Jonathan Jason (jmedinam1702@alumno.ipn.mx)
* @brief
* @version 0.1
* @date 2023-04-22
*
* @copyrigth GPlv3
*
*/

#include <stdio.h>
#include <math.h>

double aureo(int e1, int e2, double rac, double phi);

int main() {
    double phi = (1 + sqrt(5)) / 2;
    int e1, e2;

    printf("Ingrese el Primer digito: ");
    scanf("%d", &e1);

    printf("Ingrese el Segundo digito: ");
    scanf("%d", &e2);

    printf("%d, %d, ", e1, e2);

    double rac = fabs((double)e2 / (double)e1);
    double result = aureo(e1, e2, rac, phi);

    printf("\nEl valor de Phi es: %f", phi);

    printf("\nEl valor de %d / %d es: %f", e2, e1, result);

    return 0;
}

/// @brief funcion que calcula el numero aure
/// @param e1 el primer valor
/// @param e2 el segundo valor
/// @param rac almacen
/// @param phi el valor de phi
double aureo(int e1, int e2, double rac, double phi) {
    int e3 = e1 + e2;

    printf("%d, ", e3);

    double new_rac = fabs((double)e3 / (double)e2);

    if (fabs(rac - new_rac) < 0.001) {
        return new_rac;
```

```
    } else {
        return aureo(e2, e3, new_rac, phi);
    }
}
```

3.2.20. Ejecución

```
    Ingrese el Primer digito: 10
    Ingrese el Segundo digito: 12
    10, 12, 22, 34, 56, 90, 146, 236, 382, 618,
    El valor de Phi es: 1.618034
    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.