INVRUNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

Alumno
Córdova Fernández Karla Lilia

Unidad de Aprendizaje: Programación Avanzada

Profesor
M. en C. Niels Henrik Navarrete Manzanilla

Práctica 2
Arreglos y Strings

Ciudad de México; a 28 de octubre de 2020.

Índice

INTRODUCCIÓN

Los arreglos son estructuras de datos que almacenan una cantidad n de elementos de un mismo tipo. Las posiciones de los arreglos empiezan desde cero y terminan en (longitud - 1).

Los strings son un tipo de arreglo de char, que poseen sus propias funciones con la librería <string.h> pero al igual que un arreglo normal, se puede acceder a sus posiciones por medio de fors anidados.

DESARROLLO

PROGRAMA 1. CANTIDAD DE VOCALES

Descripción

Obtener la cantidad de vocales que existe en una palabra.

Análisis

1. **¿Cuáles son la entradas y salidas?**

Entradas:

char palabra[100]; valor entero donde se almacena el dato del que se quiere obtener el factorial.

Salidas:

int cantidad_vocales;

2. **¿Qué es lo que hará el programa?**

Se solicitará al usuario que introduzca una palabra que se guardará en un arreglo. En un array de char, se almacenarán todas las vocales en mayúsculas y minúsculas. Se comparará ambas cadenas y cada vez que una letra en la palabra se encuentre en el arreglo de vocales, se sumará un uno al contador de vocales.

3. **¿Qué espero de salida?**

Imprimir -> cantidad_vocales.

Requerimientos Funcionales

El programa recibe una palabra de longitud máxima de 100 caracteres.

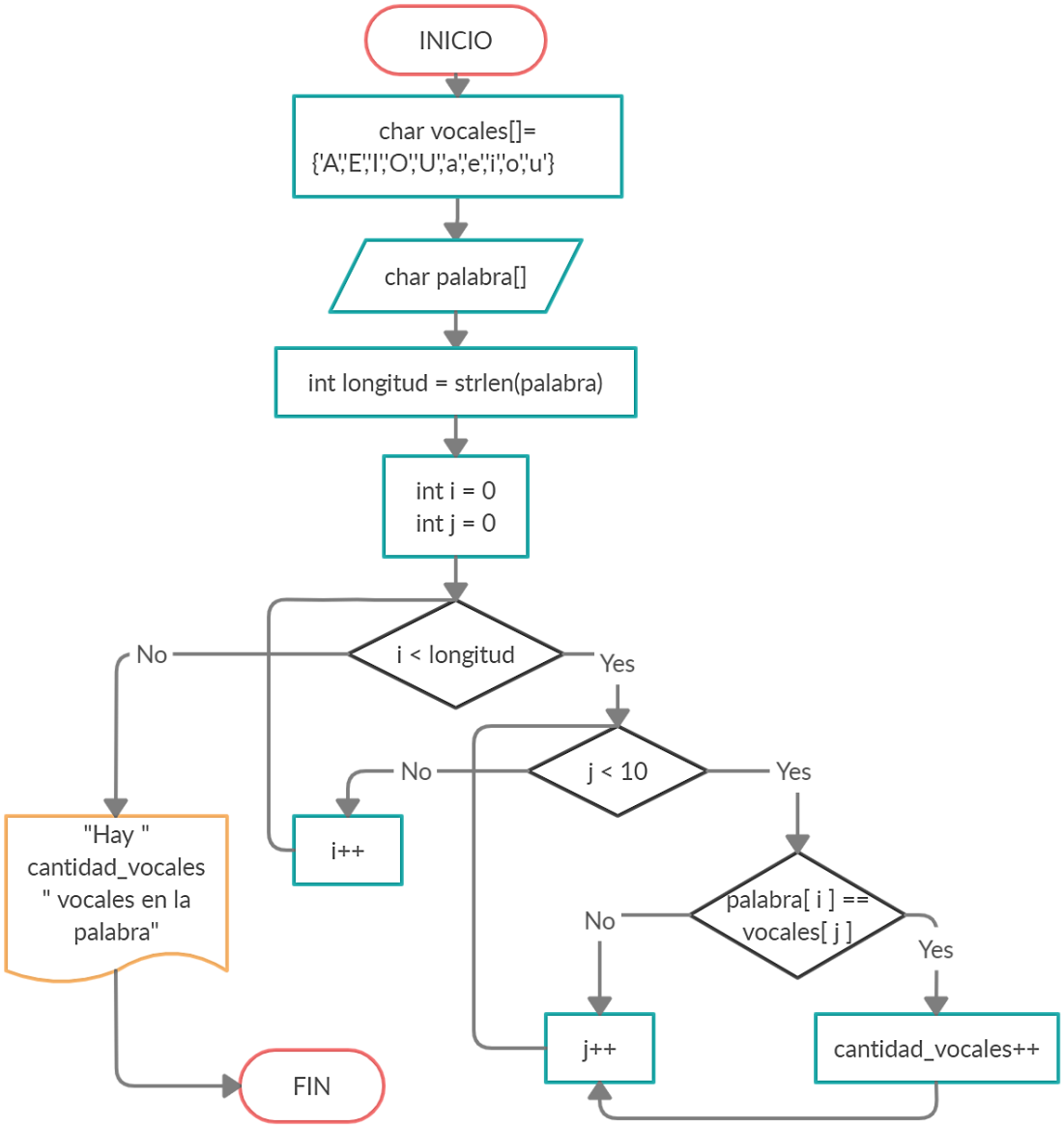
El programa calcula el número de vocales que posee la palabra sin importar si son mayúsculas o minúsculas.

El programa termina hasta que el usuario de la opción de salir.

Requerimientos no Funcionales

El programa no calcula las vocales de más de una palabra, de introducirse más de una, sólo tomará en consideración la primera.

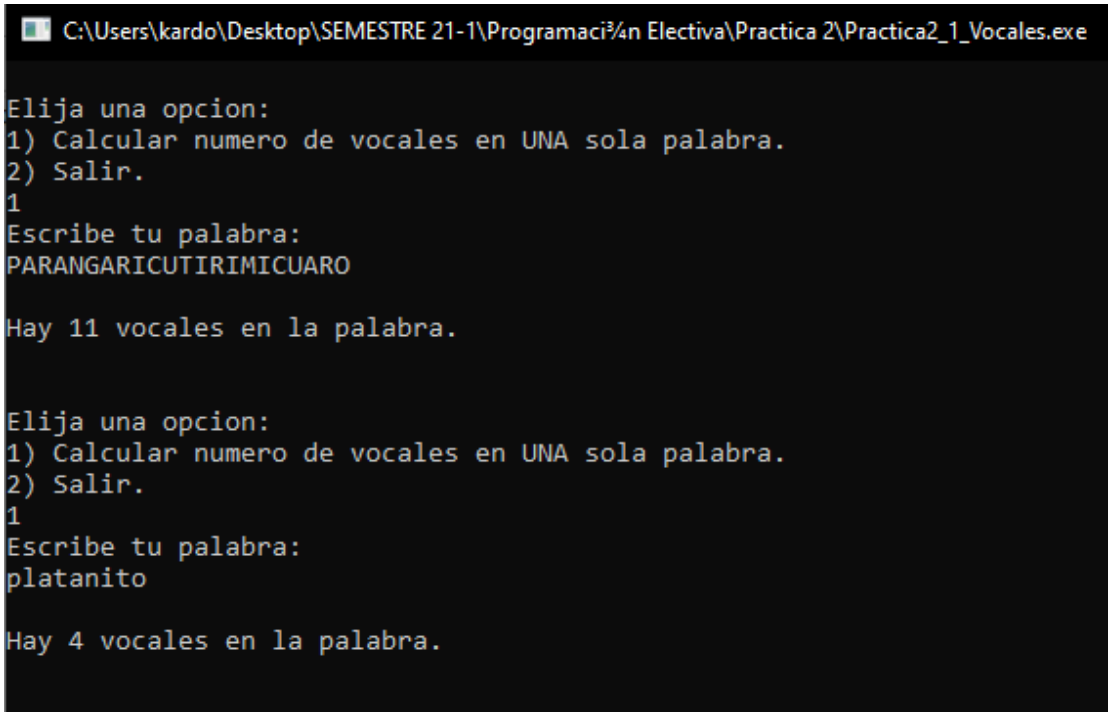
Diagrama de flujo



Código en C

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main(){
    char vocales[10] = {'A','E','I','O','U','a','e','i','o','u'};
    char palabra[100];
    int longitud = 0, cantidad_vocales = 0, indice = 0;

while(indice != 2){
    indice = 0;
    printf("\nElija una opcion:\n1) Calcular numero de vocales en UNA sola palabra.\n2) Salir.\n");
    scanf("%d",&indice);
    fflush(stdin);
    cantidad_vocales = 0;

    if(indice == 1){
        printf("Escribe tu palabra:\n");
        scanf("%s",&palabra);
        fflush(stdin);
        longitud = strlen(palabra);

        for(int i=0; i<longitud; i++){
            for(int j=0; j<10; j++){
                if(palabra[i] == vocales[j])
                cantidad_vocales++;
            }
        }
        printf("\nHay %d vocales en la palabra.\n\n",cantidad_vocales);

    }
}

return 0;
}
```

Resultados (compilación)



Caso 1:

palabra[] =
{‘P’,’A’,’R’,’A’,’N’,’G’,’A’,’R’,’I’,’C’,’U’,’T’,’I’,’R’,’I’,’M’,’I’,’C’,’U’,’A’,’R’,’O’}

cantidad_vocales = 11

Caso 2:

palabra[] = {‘p’,’l’,’a’,’t’,’a’,’n’,’i’,’t’,’o’}

cantidad_vocales = 4

PROGRAMA 2. TRIÁNGULO EQUILÁTERO

Descripción

Crear un programa que dibuje un triángulo equilátero.

Análisis

1. **¿Cuáles son la entradas y salidas?**

Entradas:

Int longitud; valor entero del tamaño de la base del triángulo.

Salidas:

char triangulo[ ][ ]; arreglo donde se va a guardar al triángulo

2. **¿Qué es lo que hará el programa?**

Se solicitará al usuario que introduzca la longitud de los lados del triángulo. En una matriz se dibujará el triángulo con asteriscos y espacios en blanco. Con el uso de ciclos for anidados se irá rellenando la matriz de medidas [longitud][longitud+(longitud-1)], donde una posición tendrá un asterisco y los de su alrededor será vacío.

3. **¿Qué espero de salida?**

Imprimir -> triangulo[ ][ ]

Requerimientos Funcionales

El programa imprime un triángulo equilátero de cualquier longitud de lados entera que el usuario introduzca.

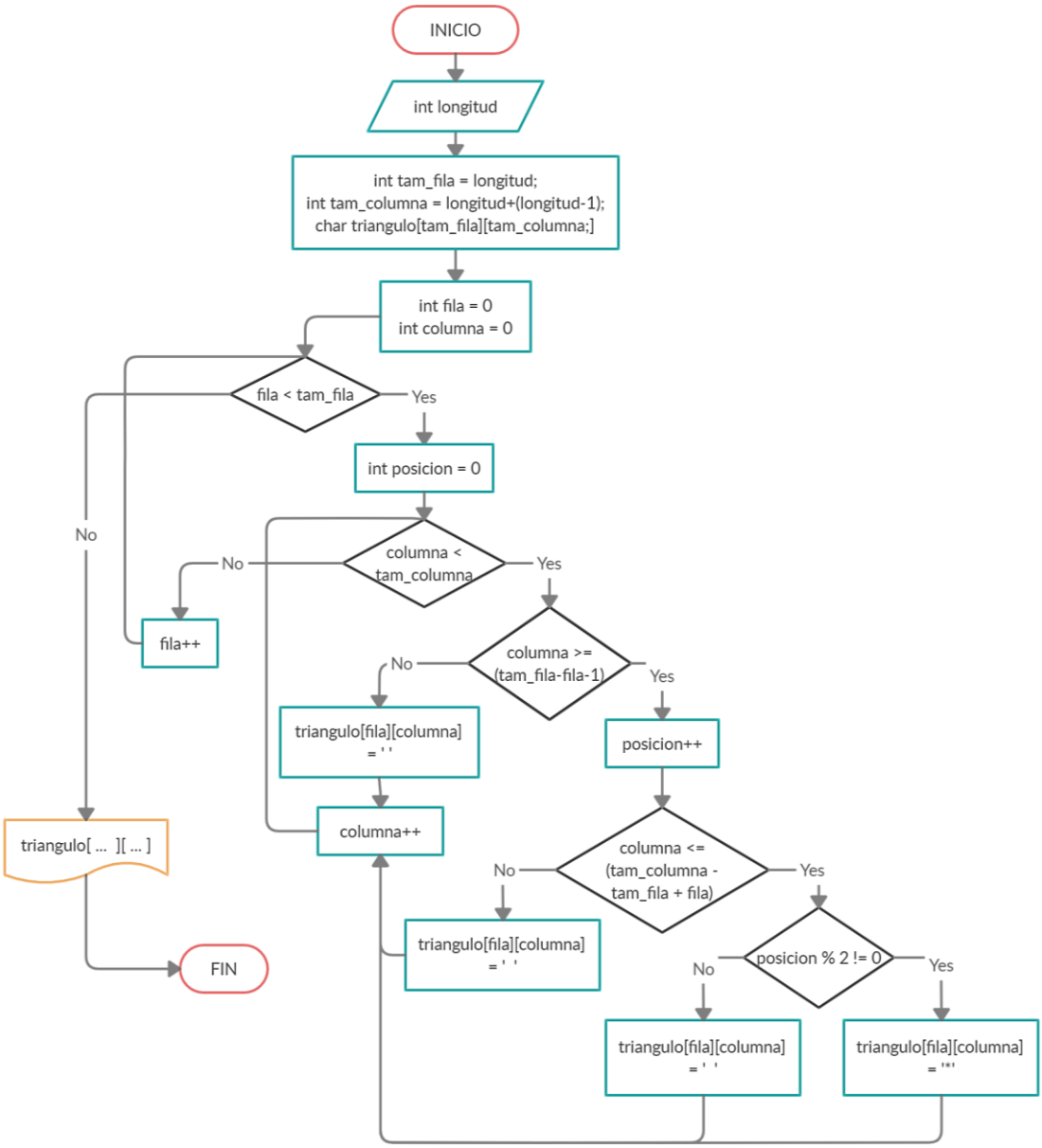
El programa se detiene hasta que el usuario de en la opción salir.

Requerimientos no Funcionales

El programa no acepta longitud de lados flotante o decimal.

El programa no acepta valor negativo ya que es una medida inválida.

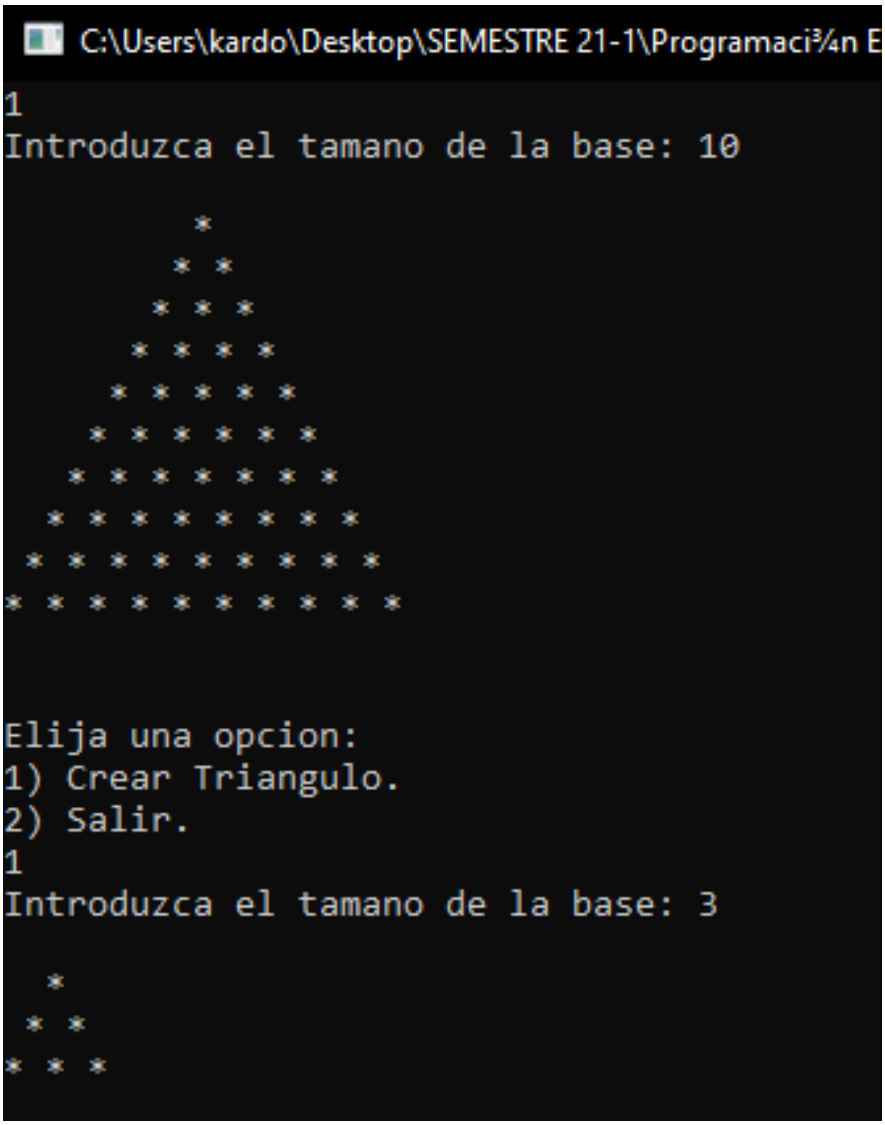
Diagrama de flujo



Código en C

```c
#include <stdlib.h>
#include <stdio.h>

int main(){

int indice, longitud, tam_fila = 0, tam_columna = 0, posicion = 0;

while(indice != 2){
    indice = 0;
    printf("\nElija una opcion:\n1) Crear Triangulo.\n2) Salir.\n");
    scanf("%d",&indice);
    fflush(stdin);

    if(indice == 1){
        longitud = 0;
        printf("Introduzca el tamano de la base: ");
        scanf("%d",&longitud);
        fflush(stdin);

        tam_fila = longitud;
        tam_columna = longitud+(longitud-1);
        char triangulo[tam_fila][tam_columna];

        for(int fila=0; fila<tam_fila; fila++){
        posicion = 0;
            for(int columna=0; columna<tam_columna; columna++){
                if(columna >= (tam_fila-fila-1)){
                    posicion++;
                    triangulo[fila][columna] = (columna<=(tam_columna - tam_fila + fila)) ?
                     ((posicion%2 != 0) ? '*':' '):' ';
                }
                else
                    triangulo[fila][columna] = ' ';
            }
        }

        for(int fila=0; fila<tam_fila; fila++){
            printf("\n");
            for(int columna=0; columna<tam_columna; columna++){
                printf("%c",triangulo[fila][columna]);
            }
        }
        printf("\n\n");
    }
}
}
```

Resultados (compilación)



Caso 1:

longitud = 10

Caso 2:

longitud = 3

PROGRAMA 3. MATRIZ IDENTIDAD

Descripción

Crear una matriz identidad.

Análisis

4. ¿Cuáles son la entradas y salidas?

Entradas:

int longitud; cantidad de filas y columnas de la matriz cuadrada.

Salidas:

int matrizID[longitud][longitud];

5. ¿Qué es lo que hará el programa?

Se solicitará al usuario que introduzca la longitud de la matriz, es decir, la cantidad de filas y columnas (ya que es una matriz cuadrada). El programa con el uso de dos fors anidados, rellenará con ceros todas las posiciones exceptuando aquellas donde el número de fila es igual al número de columna, para formar la diagonal de unos.

6. ¿Qué espero de salida?

Imprimir -> matrizID[longitud][longitud]

Requerimientos Funcionales

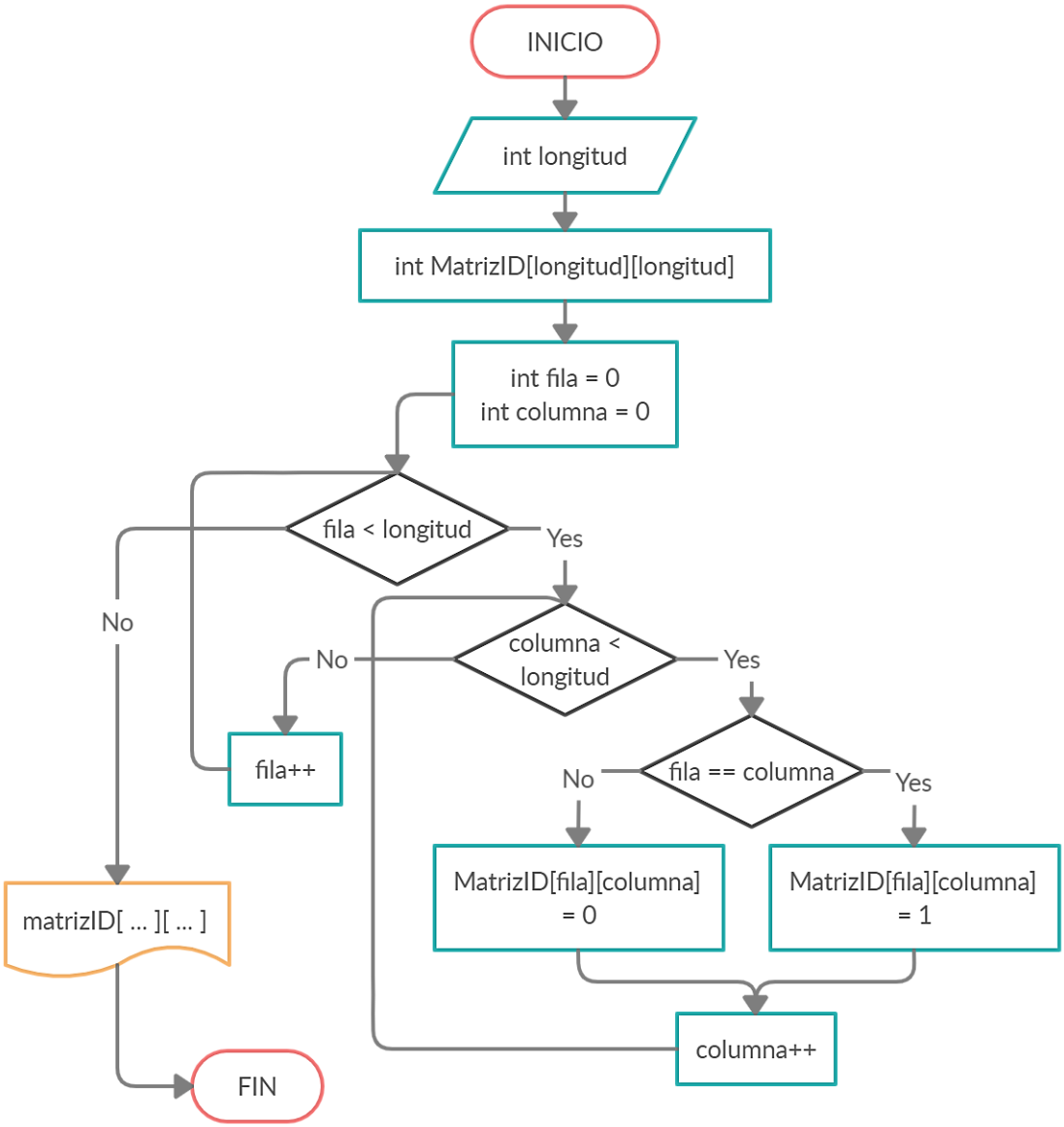
El programa crea y muestra matrices identidad de cualquier longitud entera.

El programa se detiene hasta que el usuario de en la opción salir.

Requerimientos no Funcionales

El programa no admite como dato de longitud un número flotante o decimal, ni negativo, ya que son valores inválidos.

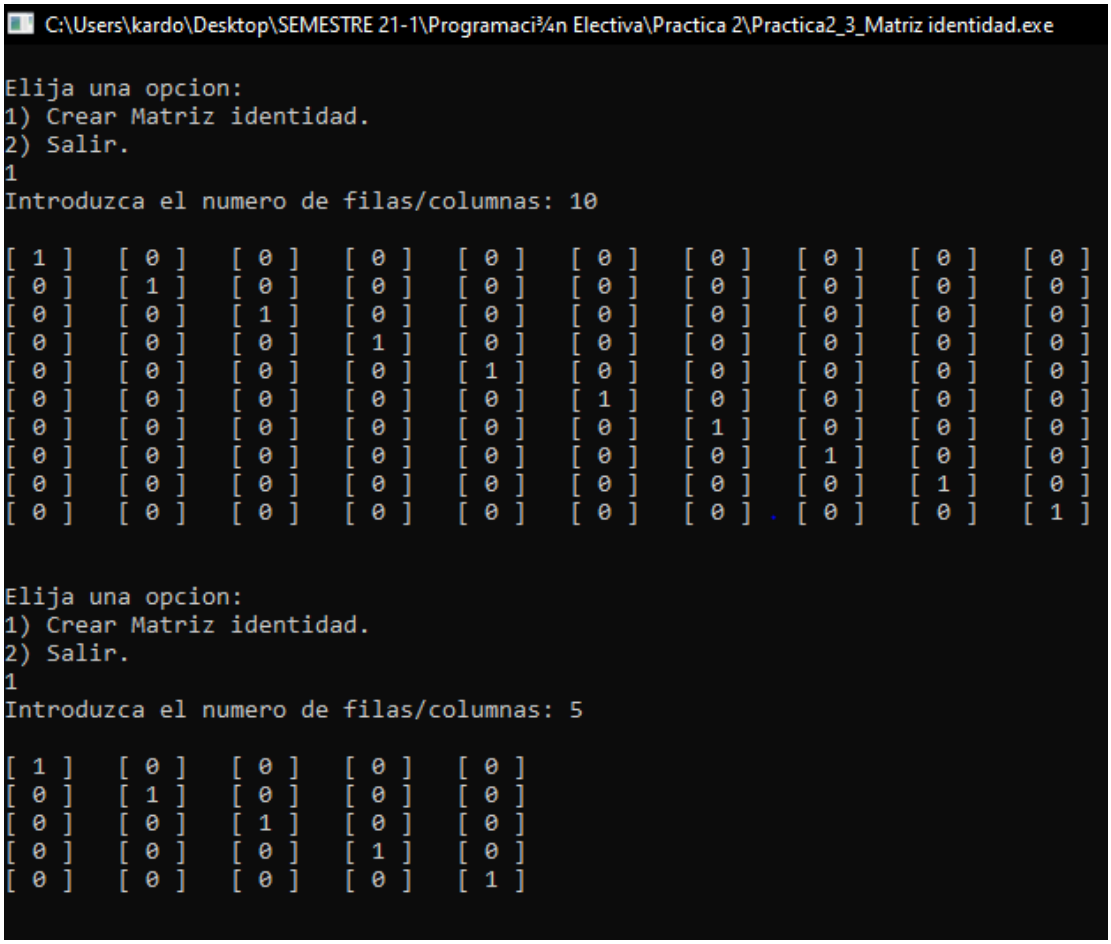
Diagrama de flujo



Código en C

```c
#include <stdlib.h>
#include <stdio.h>

int main(){

int indice, longitud;

while(indice != 2){
    indice = 0;
    printf("\nElija una opcion:\n1) Crear Matriz identidad.\n2) Salir.\n");
    scanf("%d",&indice);
    fflush(stdin);

    if(indice == 1){
        longitud = 0;
        printf("Introduzca el numero de filas/columnas: ");
        scanf("%d",&longitud);
        fflush(stdin);
        int MatrizID[longitud][longitud];

        for(int fila=0; fila<longitud; fila++){
            for(int columna=0; columna<longitud; columna++){
                MatrizID[fila][columna] = (fila == columna) ? 1:0;
            }
        }

        for(int fila=0; fila<longitud; fila++){
            printf("\n");
            for(int columna=0; columna<longitud; columna++){
                printf("[ %d ]\t",MatrizID[fila][columna]);
            }
        }
    printf("\n\n");
    }
}
return 0;
}
```

Resultados (compilación)



Caso 1:

longitud = 10

Caso 2:

longitud = 5

PROGRAMA 4. MATRIZ INVERSA

Descripción

Crear una matriz inversa.

Análisis

7. ¿Cuáles son la entradas y salidas?

Entradas:

int longitud; dimensión de la matriz cuadrada.

float matriz[longitud][longitud]; arreglo donde se almacenan los datos de la matriz.

Salidas:

float inversa[longitud][longitud]; arreglo donde se almacenan los datos de la matriz inversa.

8. ¿Qué es lo que hará el programa?

El programa solicitará las dimensiones de la matriz al usuario (filas y columnas del mismo tamaño porque debe ser una matriz cuadrada). El programa con el método de Montante (usando ciclos for) y la matriz identidad, calculará la matriz inversa.

9. ¿Qué espero de salida?

Imprimir -> inversa[longitud][longitud]

Requerimientos Funcionales

El programa recibe matrices de cualquier longitud.

El programa permite introducir datos enteros y flotantes, positivos y negativos.

El programa se detiene cuando el usuario de la opción de salir.

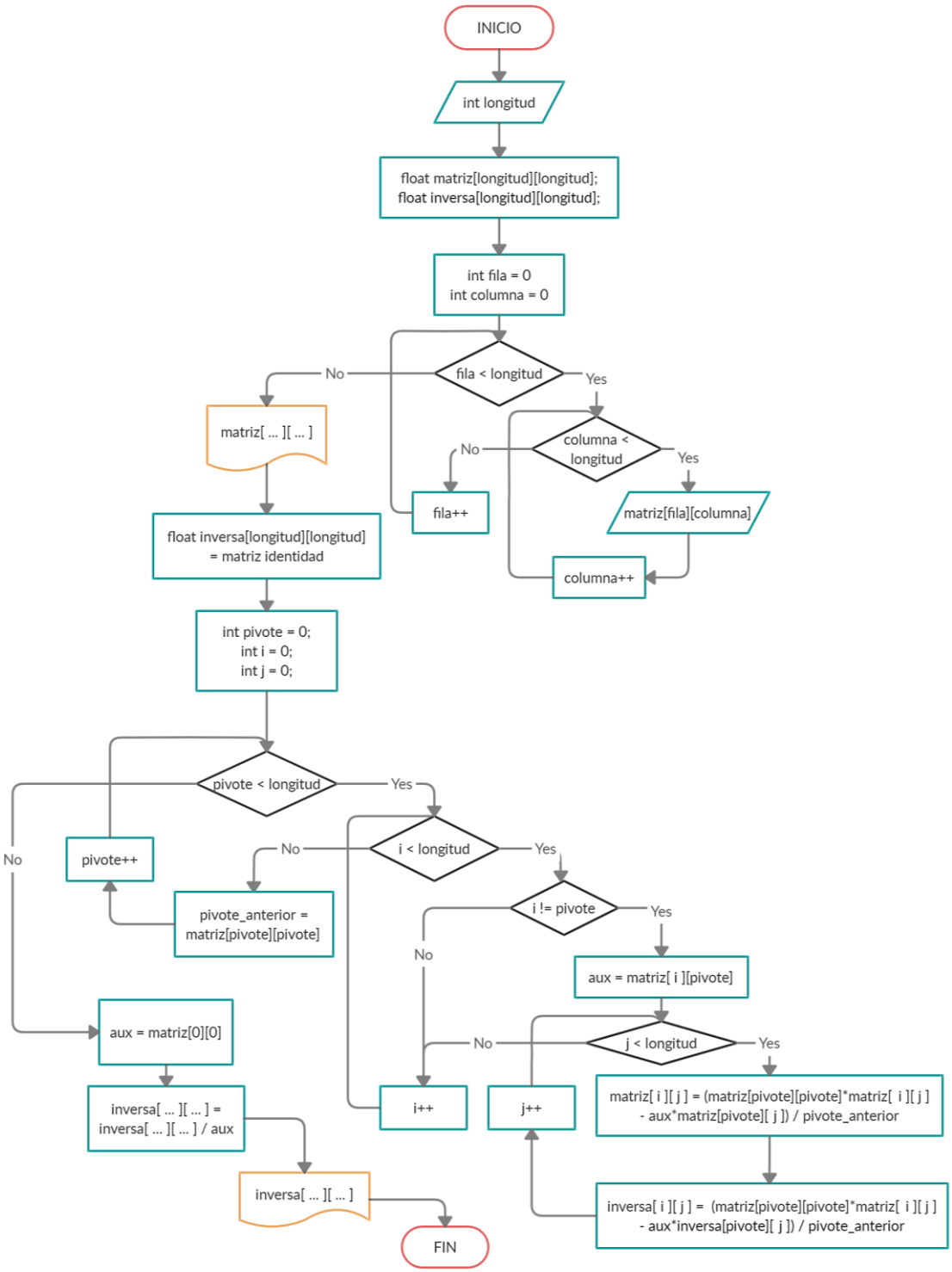
Requerimientos no Funcionales

El programa no puede calcular matrices inversas de matrices cuyos pivotes sean igual a cero.

El programa no puede calcular la matriz inversa de una matriz si su determinante es cero.

El programa no puede imprimir datos con más de dos decimales.

Diagrama de flujo



Código en C

```c
#include <stdlib.h>
#include <stdio.h>


int main(){
    int longitud = 0, indice = 0, salida = 0;
    float pivote_anterior = 1, aux = 0;


while(indice != 2){
    indice = 0;
    printf("\nElija una opcion:\n1) Crear Matriz inversa.\n2) Salir.\n");
    scanf("%d",&indice);
    fflush(stdin);

    if(indice == 1){
        longitud = 0;
        printf("Introduzca el numero de filas = columnas:\n");
        scanf("%d",&longitud);
        fflush(stdin);

        float matriz[longitud][longitud] = {};
        float inversa[longitud][longitud] = {};

        //INTRODUCIR DATOS DE MATRIZ
        printf("\nEscribe los datos de la matriz:\n");
        for(int fila=0; fila<longitud; fila++){
            for(int columna=0; columna<longitud; columna++){
                printf("Dato en [%d][%d]: ",fila,columna);
                scanf("%f",&matriz[fila][columna]);
            }
        }

        //IMPRESION DE LA MATRIZ
        for(int fila=0; fila<longitud; fila++){
            printf("\n");
            for(int columna=0; columna<longitud; columna++){
                printf("[ %.2f ]\t",matriz[fila][columna]);
            }
        }
```

```c
        //IMPRESION DE LA MATRIZ
        for(int fila=0; fila<longitud; fila++){
            printf("\n");
            for(int columna=0; columna<longitud; columna++){
                printf("[ %.2f ]\t",matriz[fila][columna]);
            }
        }

        //MATRIZ IDENTIDAD
            for(int fila=0; fila<longitud; fila++){
                for(int columna=0; columna<longitud; columna++){
                    inversa[fila][columna] = (fila == columna) ? 1:0;
                }
            }
```

```
    //MÉTODO DE MONTANTE
    for(int pivote=0; pivote<longitud; pivote++){
        if(matriz[pivote][pivote] == 0){
            salida = 1;
                    break;
        }

        for(int i=0; i<longitud; i++){   //matriz[J][i]
            if(i != pivote){
                aux = matriz[i][pivote];
                for(int j=0; j<longitud; j++){
                    matriz[i][j] = (matriz[pivote][pivote]*matriz[i][j] - aux*matriz[pivote][j])
                    /pivote_anterior;
                    inversa[i][j] = (matriz[pivote][pivote]*inversa[i][j] - aux*inversa[pivote][j])/pivote_anterior;
                }
            }
        }
        pivote_anterior = matriz[pivote][pivote];
        if(pivote_anterior == 0)
        break;
    }

 if(salida == 1)
 printf("EL PROGRAMA NO PUEDE CALCULAR LA INVERSA");

 else{
    aux = matriz[0][0];

    printf("\n\nMATRIZ INVERSA:\n");
    //IMPRESIÓN MATRIZ INVERSA
    for(int fila=0; fila<longitud; fila++){
        printf("\n");
        for(int columna=0; columna<longitud; columna++){
            printf("[ %.2f ]\t",inversa[fila][columna]/aux);
        }
    }
}
printf("\n\n");
}
```

Resultados (compilación)

```
C:\Users\kardo\Desktop\SEMESTRE 21-1\Programaci¾n Electiva\Practica 2\Practica2_4_Matriz inversa.exe
1
Introduzca el numero de filas = columnas:
3

Escribe los datos de la matriz:
Dato en [0][0]: 3
Dato en [0][1]: 4
Dato en [0][2]: 5
Dato en [1][0]: 9
Dato en [1][1]: 8
Dato en [1][2]: 0
Dato en [2][0]: 0
Dato en [2][1]: 1
Dato en [2][2]: 2

[ 3.00 ]        [ 4.00 ]        [ 5.00 ]
[ 9.00 ]        [ 8.00 ]        [ 0.00 ]
[ 0.00 ]        [ 1.00 ]        [ 2.00 ]

MATRIZ INVERSA:

[ 0.76 ]        [ -0.14 ]       [ -1.90 ]
[ -0.86 ]       [ 0.29 ]        [ 2.14 ]
[ 0.43 ]        [ -0.14 ]       [ -0.57 ]


Elija una opcion:
1) Crear Matriz inversa.
2) Salir.
```

PROGRAMA 5. OPERACIONES DE MATRICES

Descripción

Crear un programa que sume, reste y multiplique dos matrices. A cada resultado aplicar la transpuesta.

Análisis

10. ¿Cuáles son la entradas y salidas?

Entradas:

int tam_filaA, tam_columnaA; dimensiones de la matriz A.

int tam_filaB, tam_columnaB; dimensiones de la matriz B.

int matrizA[tam_filaA][tam_columnaA]; arreglo donde se guardan los valores enteros de la matriz A.

int matrizB[tam_filaB][tam_columnaB]; arreglo donde se guardan los valores enteros de la matriz B.

Salidas:

int resultado[][];

11. ¿Qué es lo que hará el programa?

Se solicitará al usuario que introduzca la cantidad de filas y columnas de cada matriz. En la suma y resta de matrices, la pareja de matrices que tengan las mismas dimensiones podrá realizar la operación. En la multiplicación, si el número de columnas en A coincide con el número de filas en B, la operación se puede realizar.

A cada operación, se obtiene su transpuesta.

12. ¿Qué espero de salida?

Imprimir -> resultado[][] y la transpuesta del resultado.

Requerimientos Funcionales

El programa sólo admite valores enteros tanto en las dimensiones como en los datos almacenados dentro de las matrices.

El programa realiza operaciones de suma, resta y multiplicación.

El programa muestra la matriz transpuesta de cada resultado de operación.

El programa se detiene hasta que el usuario de la opción de salir.

Requerimientos no Funcionales

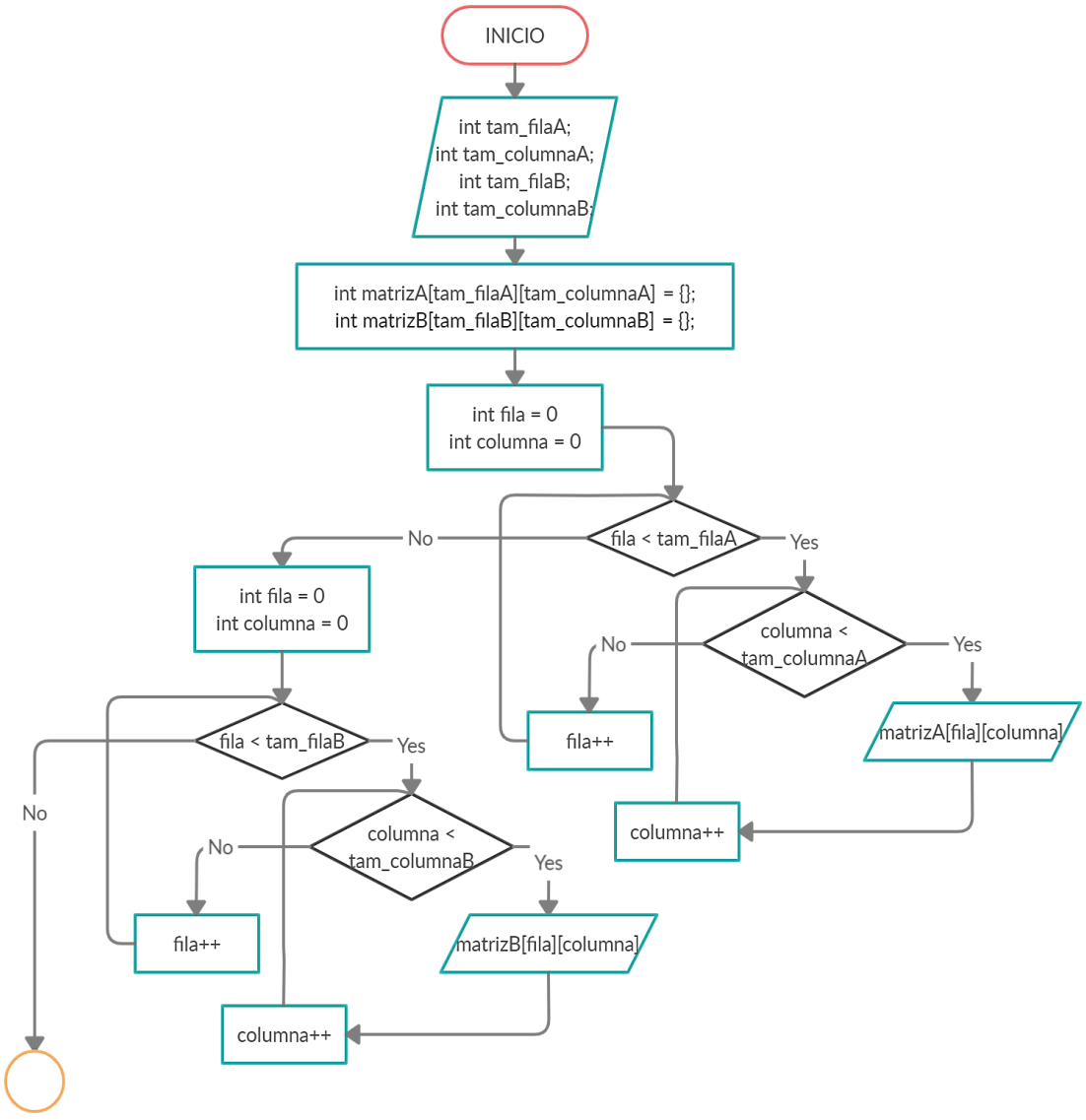
El programa no admite valores flotantes o decimales.

Si las matrices son de diferentes dimensiones, el programa no realiza suma ni resta.

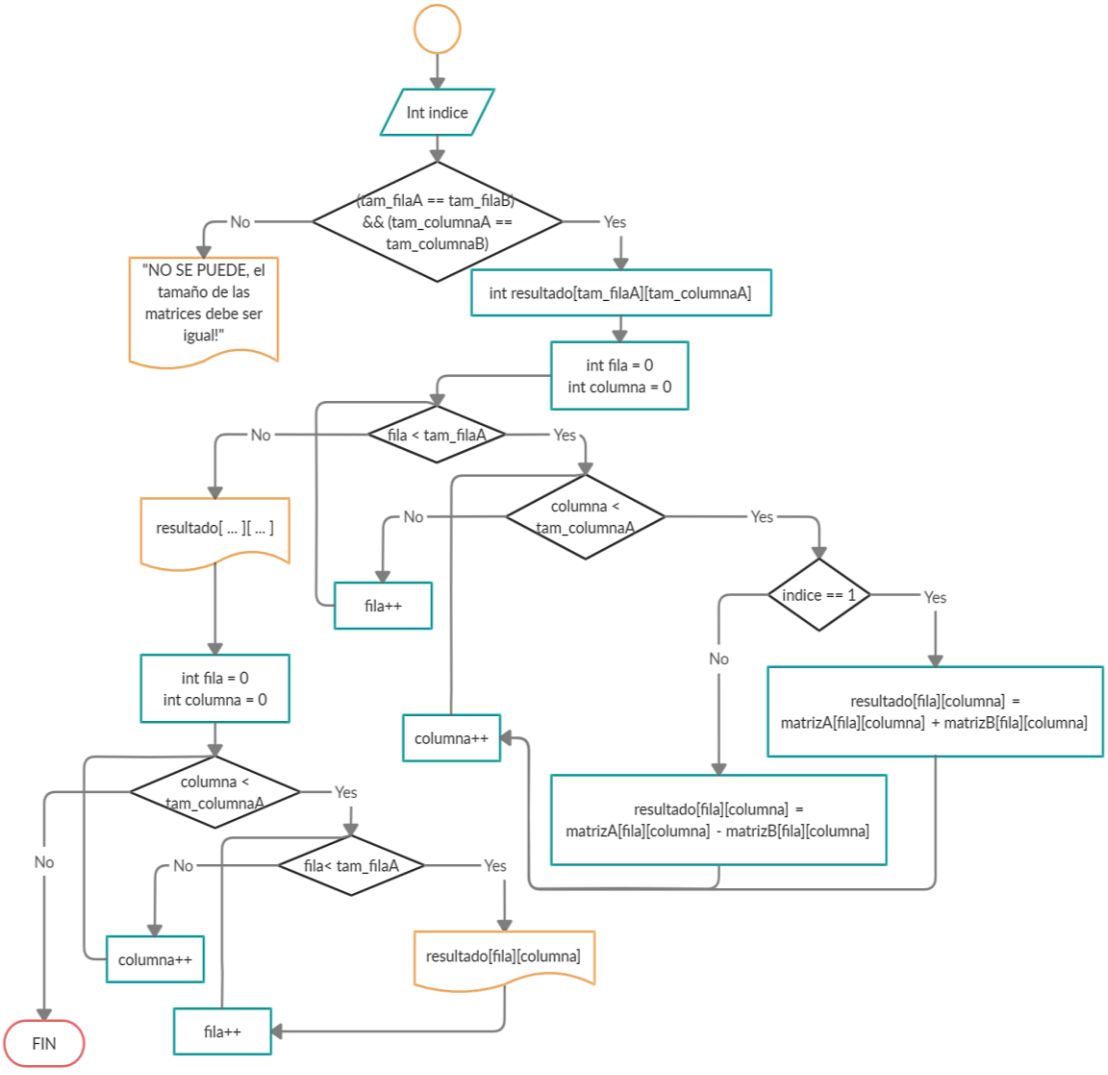
Si el número de columnas en A no coincide con el número de filas en B, el programa no realiza la operación.

Diagrama de flujo

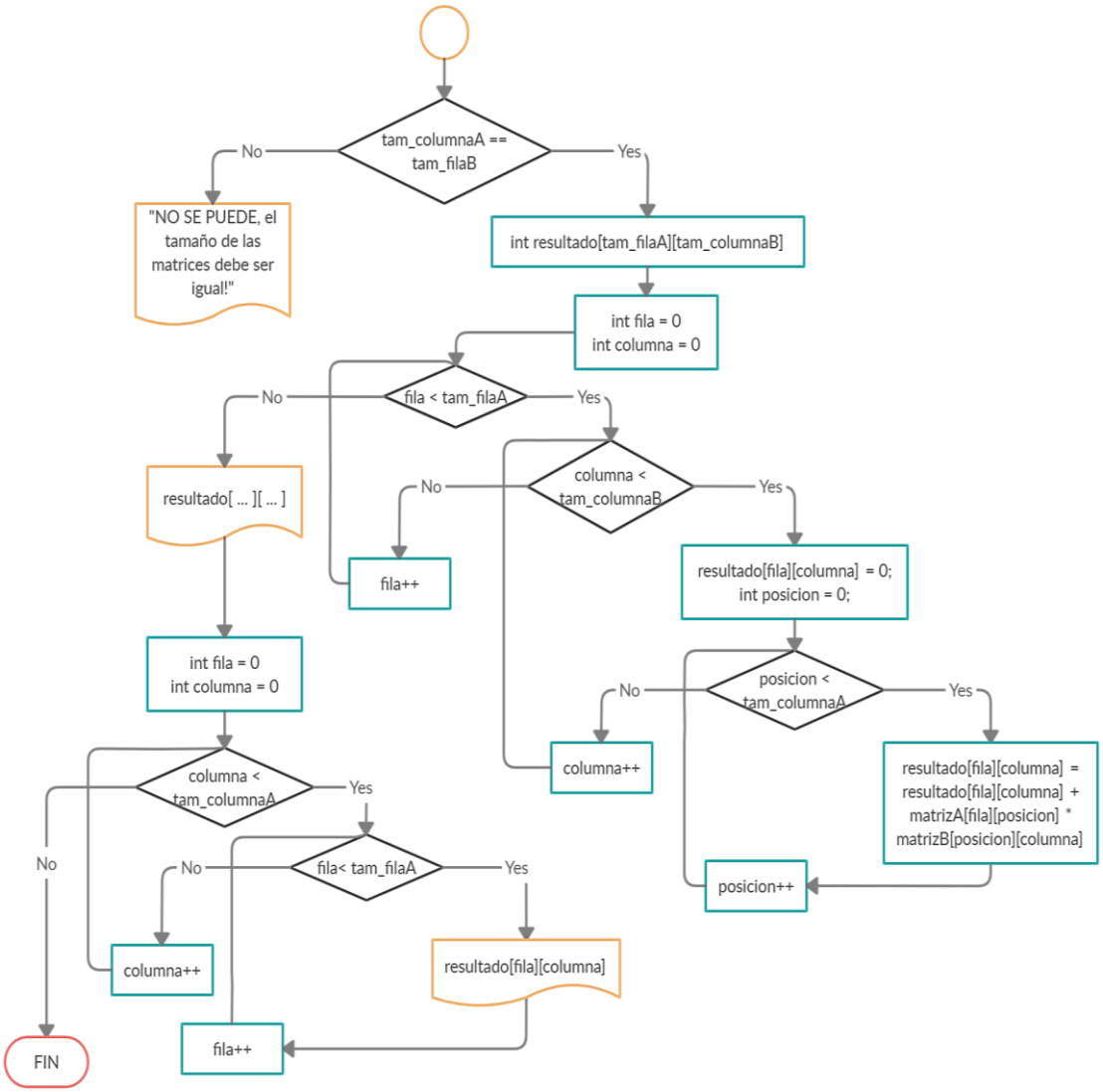
#Llenado de matrices.



#Suma, resta y la transpuesta



#Multiplicación y transpuesta

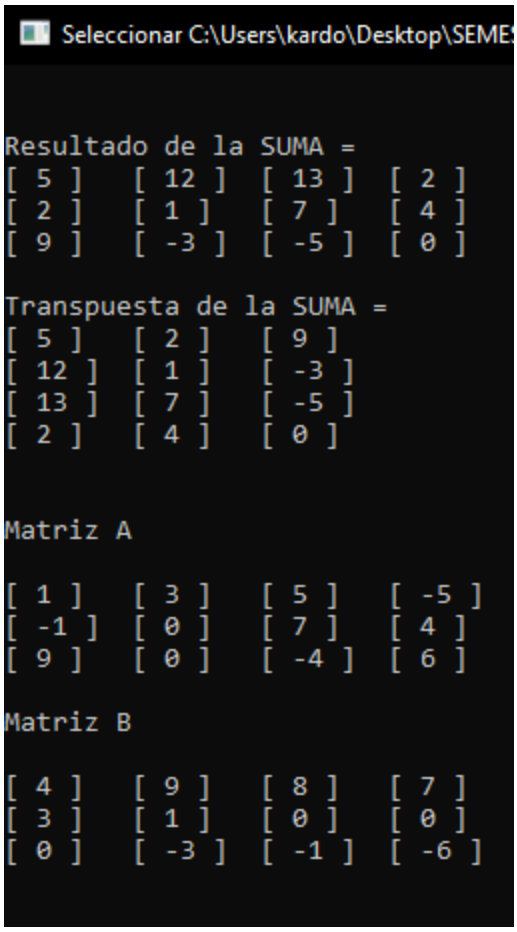


Código en C

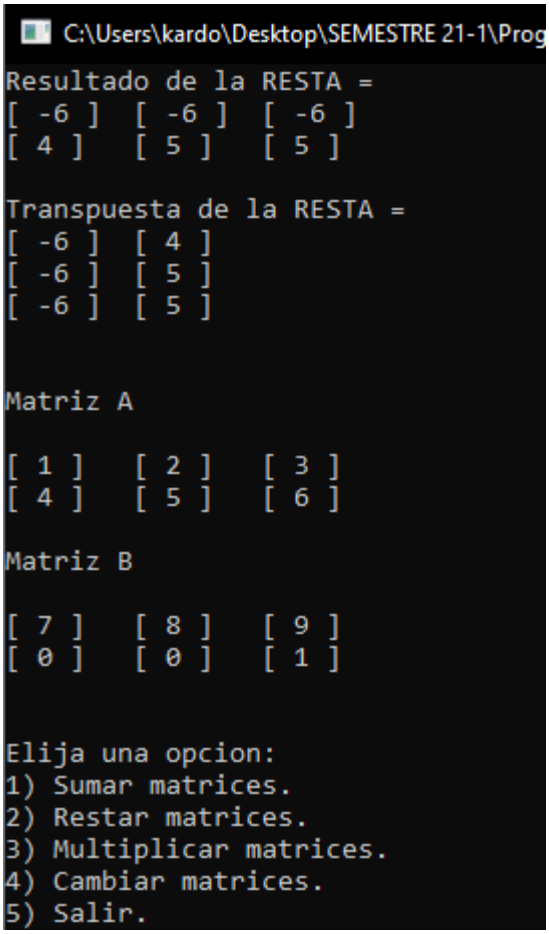
```c
#include <stdlib.h>
#include <stdio.h>

void SumaMatriz(int tam_fila,int tam_columna);

int main(){

    int indice = 0, menu = 0, longitud = 0;
    int tam_filaA,tam_columnaA, tam_filaB, tam_columnaB;

while(indice != 5){
    system("cls");
    indice = 0; tam_filaA = 0; tam_columnaA = 0; tam_filaB = 0; tam_columnaB = 0;

    printf("Introduzca cantidad de filas y columnas de matriz A: ");
    scanf("%d %d",&tam_filaA,&tam_columnaA);
    fflush(stdin);

    printf("Introduzca cantidad de filas y columnas de matriz B: ");
    scanf("%d %d",&tam_filaB,&tam_columnaB);
    fflush(stdin);

    int matrizA[tam_filaA][tam_columnaA] = {};
    int matrizB[tam_filaB][tam_columnaB] = {};

    //LLENADO DE LAS MATRICES
    printf("\nEscribe los datos de la matriz A:\n");
    for(int fila=0; fila<tam_filaA; fila++){
        for(int columna=0; columna<tam_columnaA; columna++){
            printf("Dato en [%d][%d]: ",fila,columna);
            scanf("%d",&matrizA[fila][columna]);
        }
    }
    fflush(stdin);

   printf("\nEscribe los datos de la matriz B:\n");
   for(int fila=0; fila<tam_filaB; fila++){
       for(int columna=0; columna<tam_columnaB; columna++){
           printf("Dato en [%d][%d]: ",fila,columna);
           scanf("%d",&matrizB[fila][columna]);
       }
   }
   fflush(stdin);

while(indice != 4 && indice != 5){ //MATRICES OPERANDO

       //IMPRESION DE LAS MATRICES
   printf("\nMatriz A\n");
   for(int fila=0; fila<tam_filaA; fila++){
       printf("\n");
       for(int columna=0; columna<tam_columnaA; columna++){
           printf("[ %d ]\t",matrizA[fila][columna]);
       }
   }

   printf("\n\nMatriz B\n");
   for(int fila=0; fila<tam_filaB; fila++){
       printf("\n");
       for(int columna=0; columna<tam_columnaB; columna++){
           printf("[ %d ]\t",matrizB[fila][columna]);
       }
   }

   printf("\n\n\nElija una opcion:\n1) Sumar matrices.\n2) Restar matrices.\n");
   printf("3) Multiplicar matrices.\n4) Cambiar matrices.\n5) Salir.");
   scanf("%d",&indice);
```

```
  if(indice == 1 || indice == 2){
      //SUMA Y RESTA
      if((tam_filaA == tam_filaB) && (tam_columnaA == tam_columnaB)){
              int resultado[tam_filaA][tam_columnaA];

              for(int fila=0; fila<tam_filaA; fila++){
              for(int columna=0; columna<tam_columnaA; columna++){
                  resultado[fila][columna] = (indice == 1) ? matrizA[fila][columna] + matrizB[fila][columna]:
                  matrizA[fila][columna] - matrizB[fila][columna];
              }
          }
          system("cls");
          (indice == 1) ? printf("\n\nResultado de la SUMA =") : printf("Resultado de la RESTA =");
          for(int fila=0; fila<tam_filaA; fila++){
              printf("\n");
              for(int columna=0; columna<tam_columnaA; columna++){
                  printf("[ %d ]\t",resultado[fila][columna]);
              }
          }
          //TRANSPUESTA
          (indice == 1) ? printf("\n\nTranspuesta de la SUMA =") : printf("\n\nTranspuesta de la RESTA =");
          for(int columna=0; columna<tam_columnaA; columna++){
              printf("\n");
              for(int fila=0; fila<tam_filaA; fila++){
                  printf("[ %d ]\t",resultado[fila][columna]);
              }
          }
          printf("\n\n");
      }
      else{
          system("cls");
          printf("\n\nNO SE PUEDE, el tamano de las matrices debe ser igual!\n\n");
      }
  }

    if(indice == 3){
    //MULTIPLICACION
        if(tam_columnaA == tam_filaB){
            int resultado[tam_filaA][tam_columnaB];

            for(int fila=0; fila<tam_filaA; fila++){
                for(int columna=0; columna<tam_columnaB; columna++){
                    resultado[fila][columna] = 0;
                    for(int posicion=0; posicion<tam_columnaA; posicion++){
                        resultado[fila][columna] = resultado[fila][columna] +
                        matrizA[fila][posicion] * matrizB[posicion][columna];
                    }
                }
            }
            system("cls");
            printf("\nResultado MULTIPLICACION =");
            for(int fila=0; fila<tam_filaA; fila++){
                printf("\n");
                for(int columna=0; columna<tam_columnaB; columna++){
                    printf("[ %d ]\t",resultado[fila][columna]);
                }
            }
            //TRANSPUESTA
            printf("\n\nTranspuesta de la MULTIPLICACION =");
            for(int columna=0; columna<tam_columnaB; columna++){
                printf("\n");
                for(int fila=0; fila<tam_filaA; fila++){
                    printf("[ %d ]\t",resultado[fila][columna]);
                }
            }
            printf("\n\n");
        }

        else{
            system("cls");
            printf("\n\nNO SE PUEDE, el numero de columnas de A debe ser igual al numero de filas de B!\n\n");
        }
    }
  }
}
    return 0;
}
```

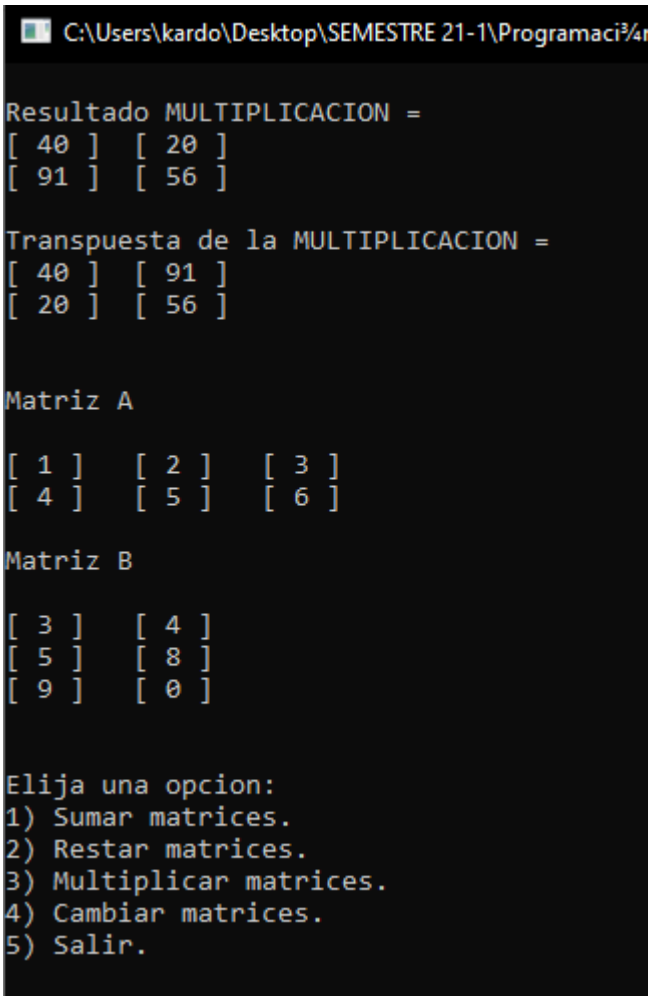
Resultados (compilación)



SUMA



RESTA



MULTIPLICACIÓN

PROGRAMA 6. DE MAYÚSCULAS A MINÚSCULAS

Descripción

Crear un programa que cambie un texto de mayúsculas a minúsculas.

Análisis

13. ¿Cuáles son la entradas y salidas?

Entradas:

char cadena[250]; arreglo donde se va a guardar la cadena de texto.

Salidas:

char cadena[250]; se devuelve la misma cadena pero con las mayúsculas convertidas en minúsculas.

14. ¿Qué es lo que hará el programa?

15.

El programa solicitará al usuario que introduzca una cadena de texto que él quiera. El programa verificará si existen mayúsculas y en caso de haber, las intercambiará por la misma letra, pero en minúscula.

16. ¿Qué espero de salida?

Imprimir -> cadena[250]

Requerimientos Funcionales

El programa sólo admite cadenas de texto que no superen la longitud de 250.

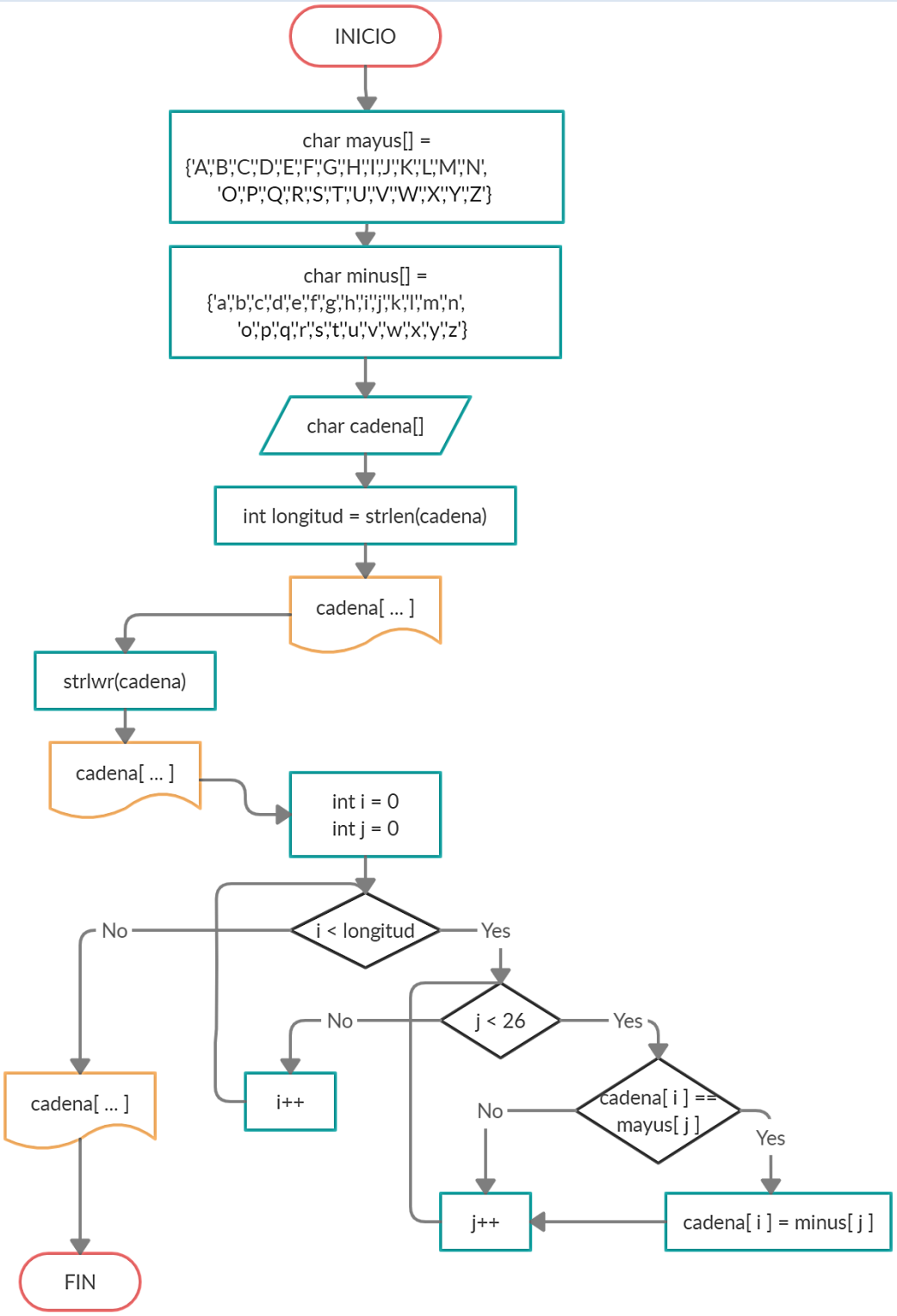
El programa convierte todas las letras mayúsculas del texto en minúsculas.

El programa se detiene cuando el usuario de la opción de salir.

Requerimientos no Funcionales

El programa no admite cadenas mayores de 250 elementos ni de longitud cero.

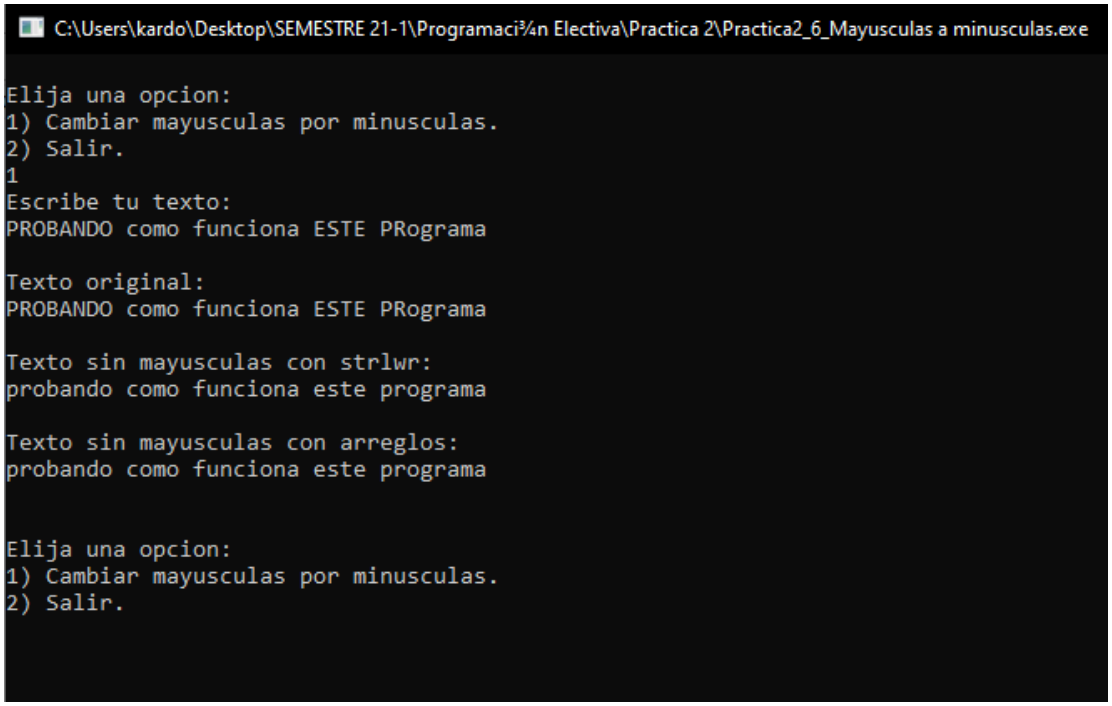
Diagrama de flujo



Código en C

```c
#include <stdlib.h>
#include <stdio.h>
#include <string.h>

int main(){

    char cadena[100];
    char mayus[] = {'A','B','C','D','E','F','G','H','I','J','K','L','M','N',
    'O','P','Q','R','S','T','U','V','W','X','Y','Z'};
    char minus[] = {'a','b','c','d','e','f','g','h','i','j','k','l','m','n',
    'o','p','q','r','s','t','u','v','w','x','y','z'};
    int indice = 0, longitud = 0;

while(indice != 2){
    printf("\nElija una opcion:\n1) Cambiar mayusculas por minusculas.\n2) Salir.\n");
    scanf("%d",&indice);
    fflush(stdin);

    if(indice == 1){
        printf("Escribe tu texto:\n");
        scanf("%[^\n]",&cadena);
        fflush(stdin);

        longitud = strlen(cadena);
        printf("\nTexto original:\n%s",cadena);

        //PRIMER CASO
        strlwr(cadena);
        printf("\n\nTexto sin mayusculas con strlwr:\n%s",cadena);

        //SEGUNDO CASO
        for(int i=0; i<longitud; i++){
            for(int j=0; j<26; j++){
                if(cadena[i] == mayus[j])
                    cadena[i] = minus[j];

            }
        }
        printf("\n\nTexto sin mayusculas con arreglos:\n%s\n\n",cadena);
    }
}

return 0;
}
```

Resultados (compilación)



cadena = “PROBANDO como funciona ESTE Programa”

Salida -> “probando como funciona este programa”

CONCLUSIONES

Las cadenas de texto en lenguaje C se almacenan en arreglos unidimensionales. Con los arreglos también se pueden almacenar y operar con matrices cuando éstos se utilizan bidimensionales.

Anidando ciclos for, se puede acceder a todas las posiciones de cualquier arreglo, ya sea para extraer el dato, cambiarlo o imprimirlo.

El manejo de arreglos es intuitivo, sin embargo, en mi caso, me tomó muchas líneas de código.