SEP
SECRETARÍA DE EDUCACIÓN PÚBLICA

Instituto Politécnico Nacional
"La Técnica al Servicio de la Patria"

upiita-ipn

UNIDAD 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
Recursividad y arreglos dinámicos

Ciudad de México; a 16 de enero de 2020.

Índice

INTRODUCCIÓN

Podemos definir a la recursividad como un método de definir un proceso a través del uso de premisas que no dan más información que el método en sí mismo o que utilizan los mismos términos que ya aparecen en su nombre, por ejemplo cuando se dice que la definición de algo es ese algo mismo.

La recursividad tiene como característica principal la sensación de infinito, de algo que es continuo y que por tanto no puede ser delimitado en el espacio o el tiempo porque se sigue replicando y multiplicando de manera lógica y matemática. Así, es común encontrar casos de recursividad por ejemplo en imágenes de espejos que hacen que la imagen sea replicada al infinito, una dentro de otra hasta que deja de verse pero no por eso deja de exisitr.

DESARROLLO

PROGRAMA CON FUNCIONES RECURSIVAS

Código del programa principal (función main y función para rellenar un arreglo de enteros).

```c
#include <stdlib.h>
#include <stdio.h>
#include <conio.h>
#include <string.h>
#include "recursividad.h"

void rellenar(int *arreglo,int longitud);

int main(){
    int *arreglo;
    int decimal, error;
    int longitud = 0,menu = 1;
    char *binario;

while(menu!=0){
    error = 0;
    system("cls");
    printf("MENU");
    printf("\n1) Ordenar de menor a mayor.");
    printf("\n2) De decimal a binario.");
    printf("\n3) De binario a decimal.");
    printf("\n4) Invertir arreglo de enteros.");
    printf("\n0) Salir.");
    scanf("%d",&menu);
    fflush(stdin);
```

```c
    switch(menu){
        case 1:
            do{
                printf("Introduce longitud del arreglo: ");
                error = scanf("%d",&longitud);
                fflush(stdin);
                if(error == 0)
                    printf("Dato incorrecto!\n");
            }while(error == 0);
```

```
        arreglo = (int*)malloc(sizeof(int)*longitud);
        rellenar(arreglo,longitud);
        printf("\nArreglo original:\n");
        leerArreglo(arreglo,(arreglo+longitud-1));
        menor_aMayor(arreglo,(arreglo+longitud));
        printf("\nArreglo ordenado:\n");
        leerArreglo(arreglo,(arreglo+longitud-1));
        getch();
    break;

    case 2:
        binario = (char*)malloc(sizeof(char)*100);
        do{
            printf("Introduce el numero entero decimal: ");
            error = scanf("%d",&decimal);
            fflush(stdin);
            if(error == 0)
                printf("Dato incorrecto!\n");
        }while(error == 0);

        binario = decimal_aBinario(decimal,binario,strlen(binario));
        printf("\nEl numero en binario es: %s\n",binario);
        getch();
    break;
```

```
    case 3:
        binario = (char*)malloc(sizeof(char)*100);
        printf("Introduce el numero entero binario: ");
        scanf("%s",binario);
        fflush(stdin);
        for(int i=0 ; i<strlen(binario) ; i++){
            if(*(binario+i) != '0' && *(binario+i) != '1'){
                error = 1;
            }
        }
        (error == 1) ? printf("El numero introducido no es valido!") :
        printf("\nEl numero en decimal es: %d\n",binario_aDecimal(binario));
        getch();
    break;
```

```
        case 4:
            do{
                printf("Introduce longitud del arreglo: ");
                error = scanf("%d",&longitud);
                fflush(stdin);
                if(error == 0)
                    printf("Dato incorrecto!\n");
            }while(error == 0);

            arreglo = (int*)malloc(sizeof(int)*longitud);
            rellenar(arreglo,longitud);
            printf("\nArreglo original:\n");
            leerArreglo(arreglo,(arreglo+longitud-1));
            invertirNum(arreglo,(arreglo+longitud-1));
            printf("\nArreglo invertido:\n");
            leerArreglo(arreglo,(arreglo+longitud-1));
            getch();
        break;
    }
}

    return 0;
}
```

```
void rellenar(int *arreglo,int longitud){
    int error = 0;
    for(int i=0; i<longitud; i++){
        do{
            printf("Arreglo[%d] = ",i);
            error = scanf("%d",(arreglo+i));
            fflush(stdin);
            if(error == 0)
                printf("Tipo de dato incorrecto, vuelva a escribirlo.\n");
        }while(error == 0);
    }
}
```

FUNCIÓN 1. ORDEN DE MAYOR A MENOR

Descripción

Escriba una función recursiva que ordene de menor a mayor un arreglo de enteros basándose en la siguiente idea: coloque el elemento más pequeño en la primera ubicación, y luego ordene el resto del arreglo con una llamada recursiva.

Análisis

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

Entradas:

int *arreglo; puntero en la posición inicial de un arreglo de enteros.

int *final; puntero en la posición final del mismo arreglo de enteros.

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

En el programa principal, se solicitará al usuario la longitud del arreglo y los datos dentro de este. Una vez que esté lleno correctamente, el programa llamará a la función de ordenamiento, donde mediante un for, se recorrerán las posiciones del arreglo comparándolas con el primer dato e intercambiando los valores cuando un número sea menor al comparado. Cuando todo el arreglo se haya recorrido, el programa manda a llamar nuevamente a la función con una posición del arreglo adelantada y repetir la acción hasta llegar al final.

3. **¿Qué espero de salida?**

La salida será el arreglo ya ordenado.

Requerimientos Funcionales

El programa acepta cualquier longitud de arreglo.

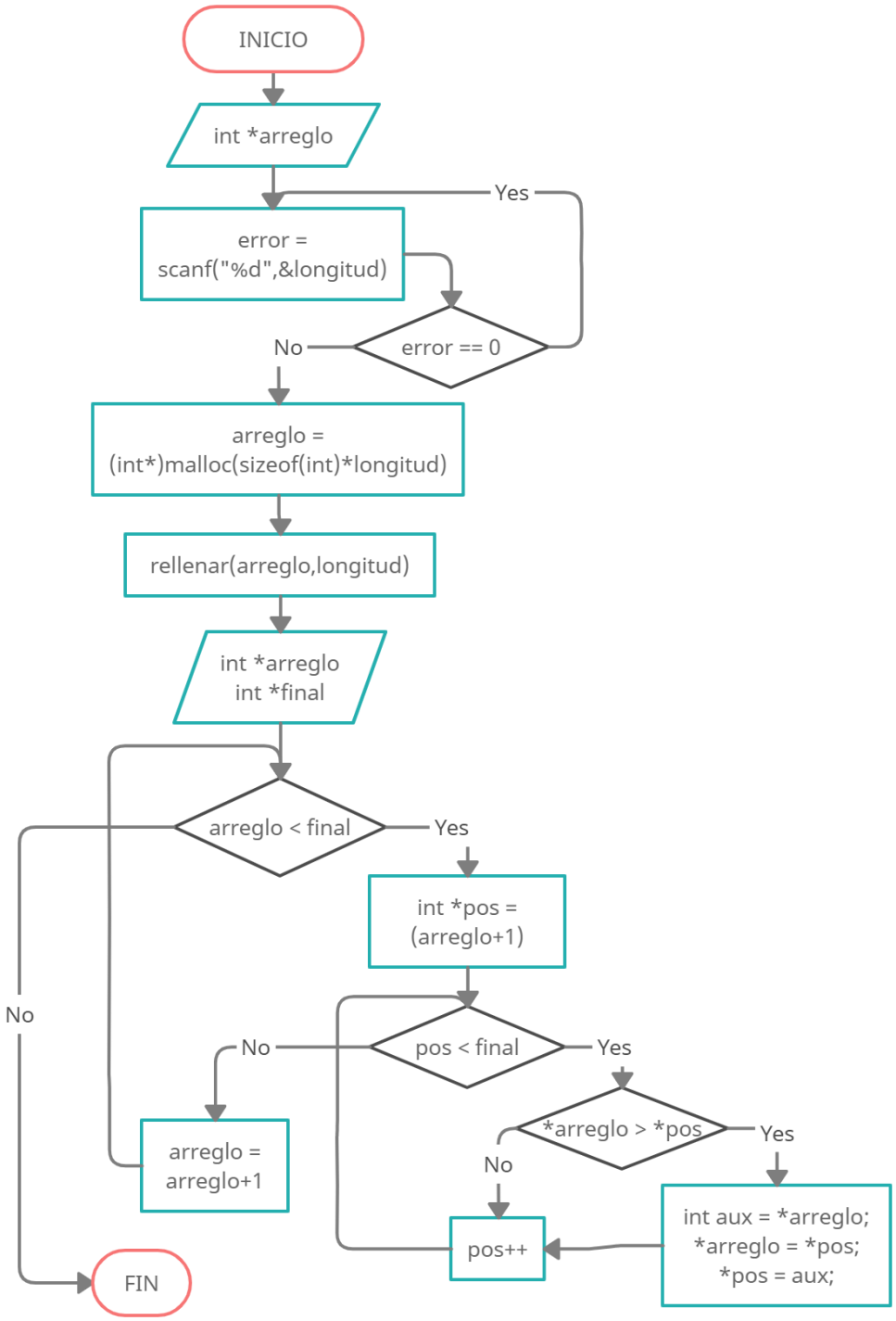
El programa únicamente acepta valores de tipo entero.

El programa avisa si hay un error con el tipo de dato y espera hasta que este sea introducido correctamente.

Requerimientos no Funcionales

El programa no acepta ningún otro valor que no sea entero y la función funciona adecuadamente sólo si se pasan correctamente los valores solicitados.

Diagrama de flujo



Código e C

Función en el archivo .h (recursividad.h)

```c
void menor_aMayor(int *arreglo,int *final){
    int aux;
    if(arreglo < final){
        for(int *pos = (arreglo+1); pos < final; pos++){
            if(*arreglo > *pos){
                aux = *arreglo;
                *arreglo = *pos;
                *pos = aux;
            }
        }
        return menor_aMayor((arreglo+1),final);
    }
    else
        return;
}
```

Resultados (compilación)

Compilación del programa principal, usando la función menor_aMayor del archivo .h

```
MENU
1) Ordenar de menor a mayor.
2) De decimal a binario.
3) De binario a decimal.
4) Invertir arreglo de enteros.
0) Salir.1
Introduce longitud del arreglo: 6
Arreglo[0] = 4
Arreglo[1] = 7
Arreglo[2] = 5
Arreglo[3] = 1
Arreglo[4] = 3
Arreglo[5] = 9

Arreglo original:
[4] [7] [5] [1] [3] [9]
Arreglo ordenado:
[1] [3] [4] [5] [7] [9]
```

FUNCIÓN 2A. DE DECIMAL A BINARIO

Descripción

Programe un método recursivo que transforme un número entero positivo a notación binaria.

Análisis

4. ¿Cuáles son la entradas y salidas?

Entradas:

int numero; número decimal a convertir en binario.

char *binario; arreglo donde se guardará el número en binario.

int longitud; longitud de elementos basura almacenados en el arreglo.

5. ¿Qué es lo que hará el programa?

En el programa principal, se solicitará al usuario introducir el número en base diez. El programa llamará a la función de conversión, si el número es mayor a uno, el número se dividirá entre dos y se almacenará este dato y el residuo. Dependiendo el valor del residuo, ese dato se guardará en la cadena. El número obtenido como cociente de la división se guarda y se pasa como nuevo valor a la siguiente llamada de la función. El proceso se repite hasta que el valor el número es indivisible entre dos, y ese valor se añade a la cadena con el resultado. Para terminar, se invierte la cadena para tener el número binario real.

6. ¿Qué espero de salida?

Se retorna la cadena con el número en notación binaria.

Requerimientos Funcionales

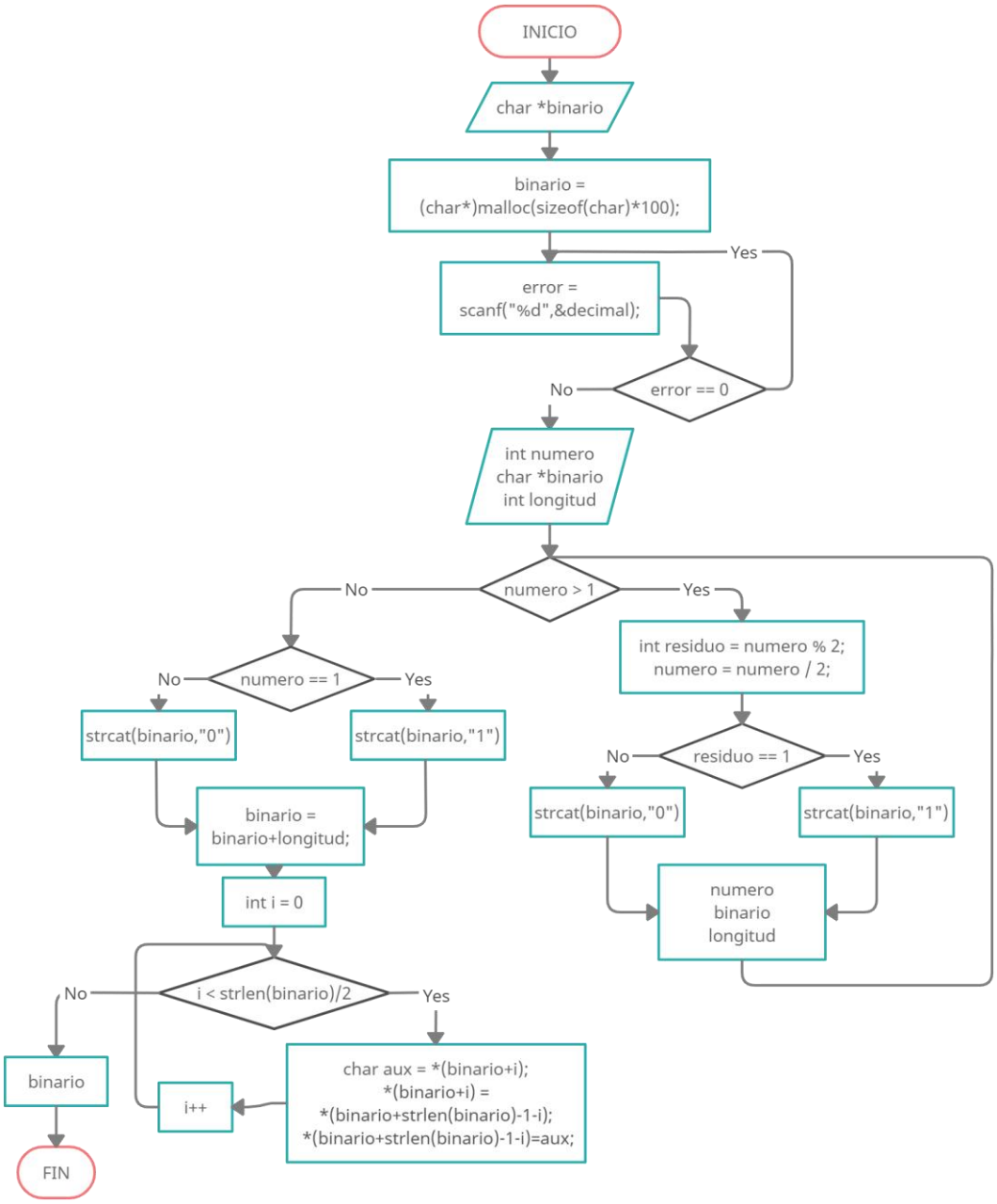
El programa acepta cualquier número entero positivo y negativo, que soporte el tipo de dato int.

El programa avisa si hay un error con el tipo de dato y espera hasta que este sea introducido correctamente.

Requerimientos no Funcionales

El programa no acepta ningún otro valor que no sea entero y la función funciona adecuadamente sólo si se pasan correctamente los valores solicitados.

Diagrama de flujo



Código e C

Función en el archivo .h (recursividad.h)

```
char *decimal_aBinario(int numero,char *binario,int longitud){
    if(numero > 1){
        int residuo = numero % 2;
        numero = numero / 2;
        (residuo == 1) ? strcat(binario,"1") : strcat(binario,"0");
        return decimal_aBinario(numero,binario,longitud);
    }
    else{
        (numero == 1) ? strcat(binario,"1") : strcat(binario,"0");
        binario = binario+longitud;
        for(int i = 0; i<strlen(binario)/2; i++){
            char aux = *(binario+i);
            *(binario+i) = *(binario+strlen(binario)-1-i);
            *(binario+strlen(binario)-1-i)=aux;
        }
        return binario;
    }
}
```

Resultados (compilación)

Compilación del programa principal, usando la función decimal_aBinario del archivo .h

```
MENU
1) Ordenar de menor a mayor.
2) De decimal a binario.
3) De binario a decimal.
4) Invertir arreglo de enteros.
0) Salir.2
Introduce el numero entero decimal: 785

El numero en binario es: 1100010001
```

FUNCIÓN 2B. DE BINARIO A DECIMAL

Descripción

Programe un método recursivo que transforme un número expresado en notación binaria a un número entero.

Análisis

7. ¿Cuáles son la entradas y salidas?

Entradas:

char *binario; arreglo donde va almacenado el número en binario.

8. ¿Qué es lo que hará el programa?

En el programa principal, se solicitará al usuario introducir el número en binario. El programa llamará a la función de conversión, y convertirá el primer dato señalado por el apuntador y lo convertirá en entero, el dato puede ser 0 o 1. Dicho valor se multiplicará por un dos elevado a su número de posición. La posición se irá determinando, contando la longitud de la cadena menos uno.

El proceso se va repitiendo con cada llamada de nuevo a la función, sumando una unidad a la cadena. Todos los resultados se van sumando y al final se retorna la sumatoria.

9. ¿Qué espero de salida?

Se retorna el valor en base diez en tipo entero.

Requerimientos Funcionales

El programa acepta cualquier número entero positivo y negativo, que soporte el tipo de dato int.

El programa avisa si hay un error con el tipo de dato y espera hasta que este sea introducido correctamente.

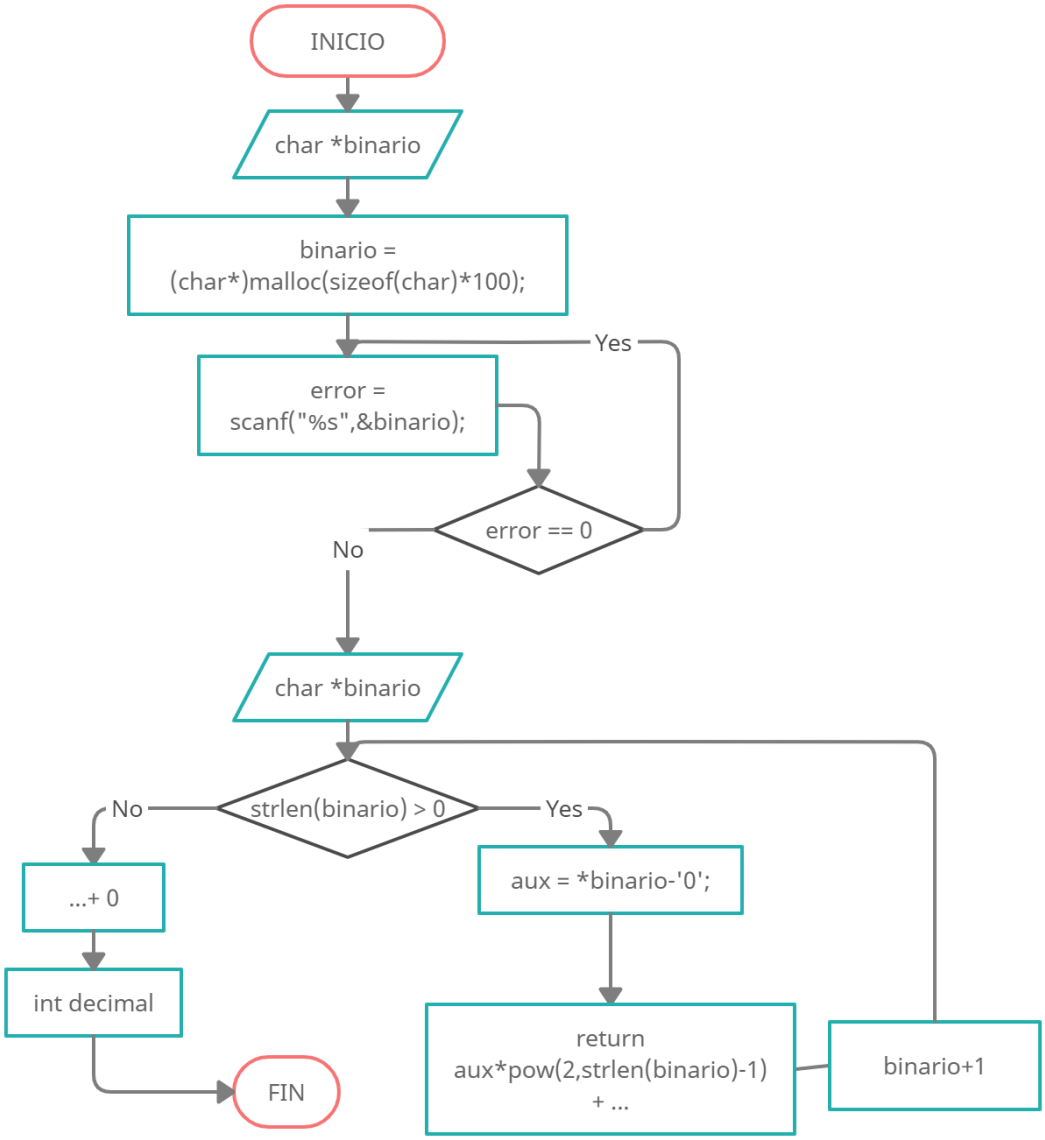
El programa muestra el resultado hasta el número máximo que el tipo de dato entero es capaz de almacenar.

Requerimientos no Funcionales

El programa no acepta ningún otro valor que no sea entero y la función funciona adecuadamente sólo si se pasan correctamente los valores solicitados.

El programa no da el resultado correcto con números que superen la capacidad del tipo de dato int.

Diagrama de flujo



Código e C

Función en el archivo .h (recursividad.h)

```c
int binario_aDecimal(char *binario){
    int aux;
    if(strlen(binario) > 0){
        aux = *binario-'0';
        return aux*pow(2,strlen(binario)-1) + binario_aDecimal(binario+1);
    }
    else return 0;
}
```

Resultados (compilación)

Compilación del programa principal, usando la función binario_aDecimal del archivo .h

```
MENU
1) Ordenar de menor a mayor.
2) De decimal a binario.
3) De binario a decimal.
4) Invertir arreglo de enteros.
0) Salir.3
Introduce el numero entero binario: 100010101011110101

El numero en decimal es: 142069
```

FUNCIÓN 3. INVERTIR ARREGLO

Descripción

Programe un método recursivo que invierta los números de un arreglo de enteros.

Análisis

10. ¿Cuáles son la entradas y salidas?

Entradas:

int *inicio; puntero en la posición inicial del arreglo de enteros.

int *final; puntero en la posición final del arreglo de enteros.

11. ¿Qué es lo que hará el programa?

En el programa principal, se solicitará al usuario introducir la longitud y los enteros guardados dentro del arreglo. El programa llamará a la función de inversión, e invertirá las posiciones de los datos guardados.

Siempre que la posición de inicio del arreglo sea menor a la final, con ayuda de una variable auxiliar se intercambiará el dato del inicio y el final. Se vuelve a llamar a la función añadiendo una unidad al puntero del inicio y restando uno a la del final. Se repite el proceso hasta que la condición ya no se cumpla y termina la recursividad.

12. ¿Qué espero de salida?

Se retorna el arreglo con el nuevo orden.

Requerimientos Funcionales

El programa acepta cualquier longitud del arreglo.

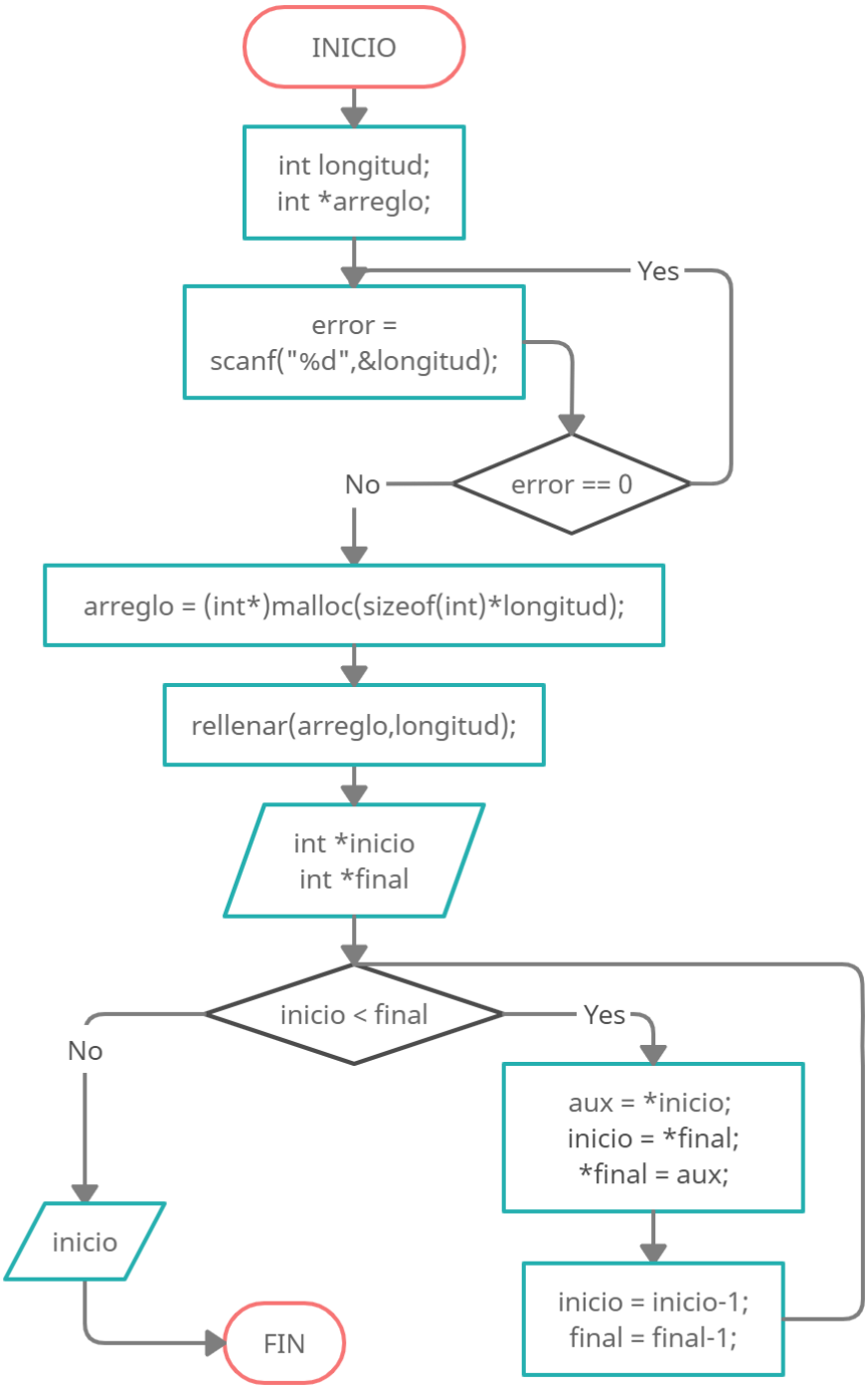
El programa solo acepta valores de tipo int para el arreglo.

El programa avisa si hay un error con el tipo de dato y espera hasta que este sea introducido correctamente.

Requerimientos no Funcionales

El programa no acepta ningún otro valor que no sea entero.

Diagrama de flujo



Código e C

Función en el archivo .h (recursividad.h)

```c
int *invertirNum(int *inicio,int *final){
    int aux;
    if(inicio < final){
        aux = *inicio;
        *inicio = *final;
        *final = aux;
        return invertirNum(inicio+1,final-1);
    }
    else
        return inicio;
}
```

Resultados (compilación)

Compilación del programa principal, usando la función invertirNum del archivo .h

```
MENU
1) Ordenar de menor a mayor.
2) De decimal a binario.
3) De binario a decimal.
4) Invertir arreglo de enteros.
0) Salir.4
Introduce longitud del arreglo: 8
Arreglo[0] = 5
Arreglo[1] = 6
Arreglo[2] = 7
Arreglo[3] = 8
Arreglo[4] = 9
Arreglo[5] = 7
Arreglo[6] = 5
Arreglo[7] = 4

Arreglo original:
[5] [6] [7] [8] [9] [7] [5] [4]
Arreglo invertido:
[4] [5] [7] [9] [8] [7] [6] [5]
```

FUNCIÓN 4. LEER ARREGLO

Descripción

Crear un arreglo dinámico de enteros de tamaño -n- y crear una función que lea el contenido el mismo.

Análisis

13. ¿Cuáles son la entradas y salidas?

Entradas:

int *inicio; puntero en la posición inicial del arreglo de enteros.

int *final; puntero en la posición final del arreglo de enteros.

14. ¿Qué es lo que hará el programa?

En el programa principal, se solicitará al usuario introducir la longitud y los enteros guardados dentro del arreglo. El programa llamará a la función de inversión, e invertirá las posiciones de los datos guardados.

Siempre que la posición de inicio del arreglo sea menor a la final, con ayuda de una variable auxiliar se intercambiará el dato del inicio y el final. Se vuelve a llamar a la función añadiendo una unidad al puntero del inicio y restando uno a la del final. Se repite el proceso hasta que la condición ya no se cumpla y termina la recursividad.

15. ¿Qué espero de salida?

Se retorna el arreglo con el nuevo orden.

Requerimientos Funcionales

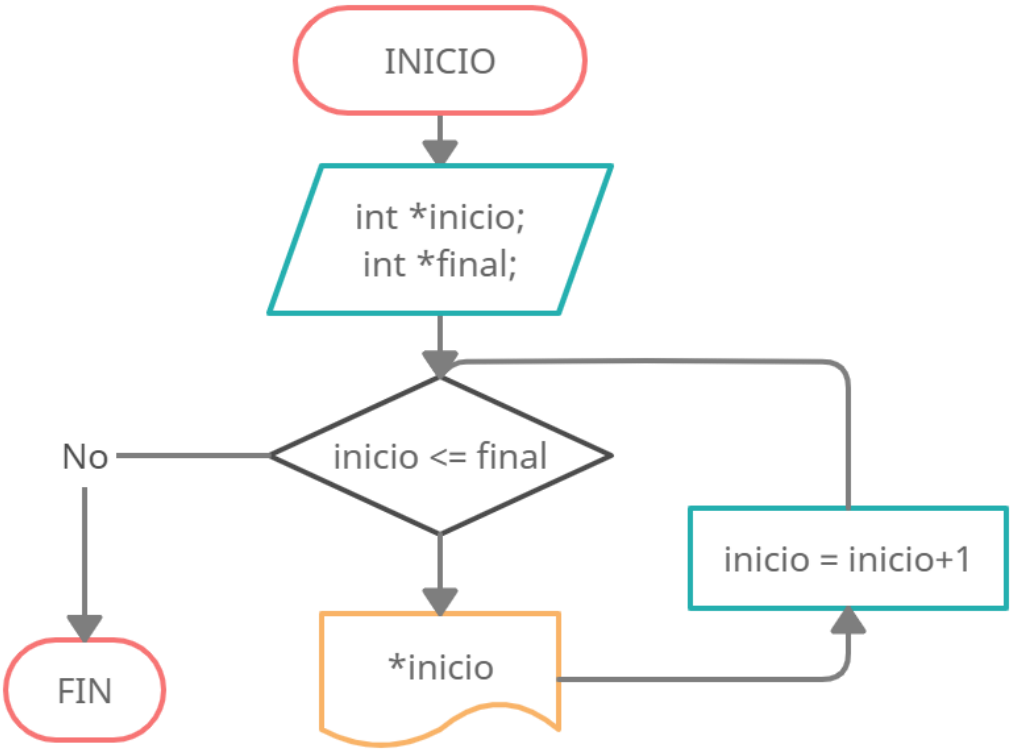
El programa sólo lee arreglos que sean de tipo entero.

El programa acepta cualquier longitud del arreglo.

Requerimientos no Funcionales

El programa no acepta ningún otro valor que no sea entero.

Diagrama de flujo



Código e C

Función en el archivo .h (recursividad.h)

```
void leerArreglo(int *inicio,int *final){
    if(inicio <= final){
        printf("[%d] ",*inicio);
        return leerArreglo((inicio+1),final);
    }
    else
        return;
}
```

CONCLUSIONES

La recursividad sirve para simplificar códigos que requerirían ciclos repetitivos con una mínima variante. Con recursividad la función se llama a sí misma hasta cumplir una condición para finalizar.

Usar archivos .h es como crear una propia biblioteca, ya que en este archivo se pueden crear las funciones que se consideren de uso general y con una simple línea de código #include “archivo.h” se puede utilizar todo lo escrito dentro de este archivo en cualquier otro ejecutable, sin necesidad de tener todo el código en un solo documento.