LENGUAJE C

PROF. YULITH VANESSA ALTAMIRANO FLORES

REPORTE DE PRACTICA #5
CLASES DE ALMACENAMIENTO

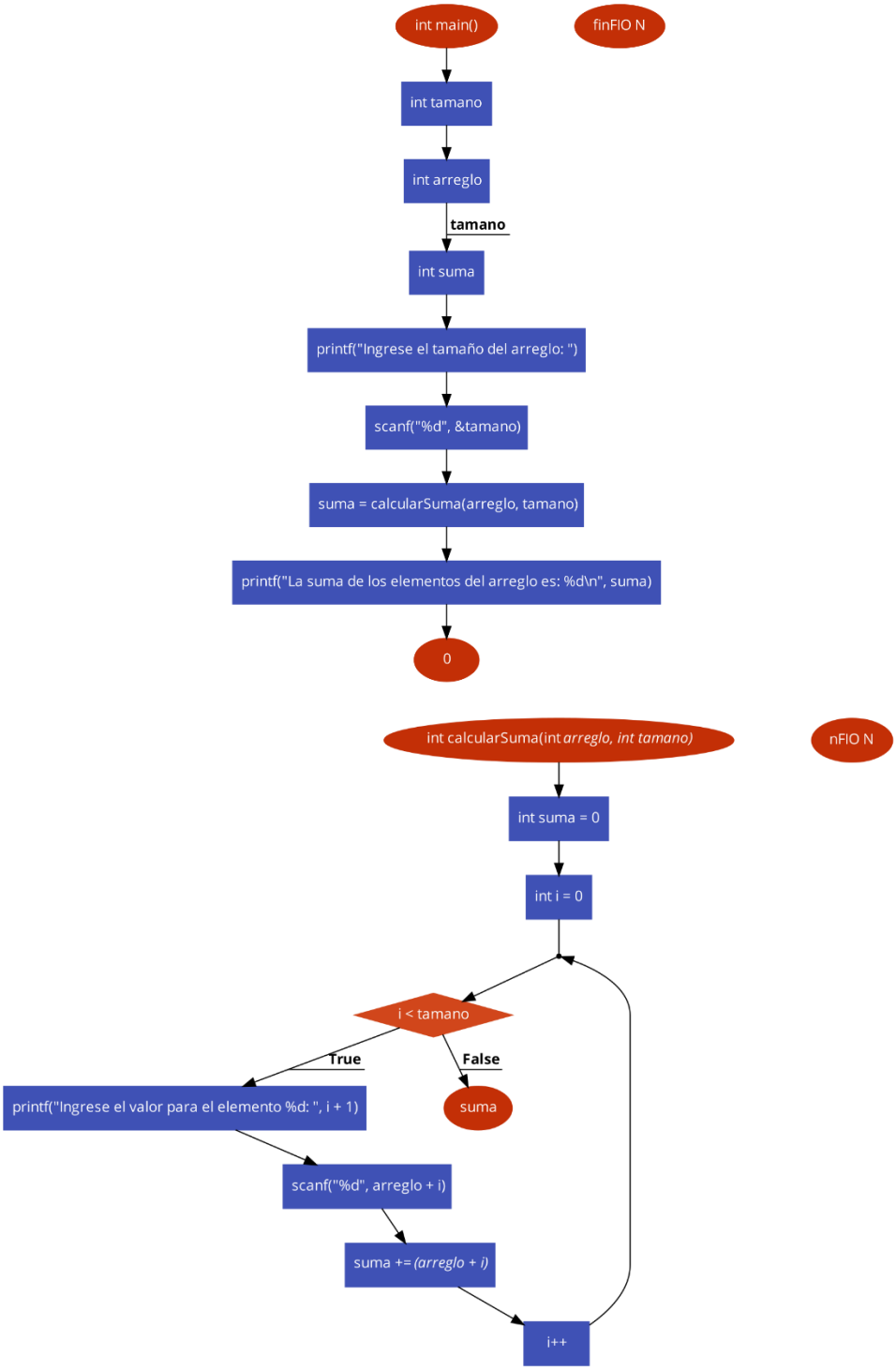
KEVIN ALEJANDRO GONZALEZ TORRES
GRUPO 932

REPOSITORIO

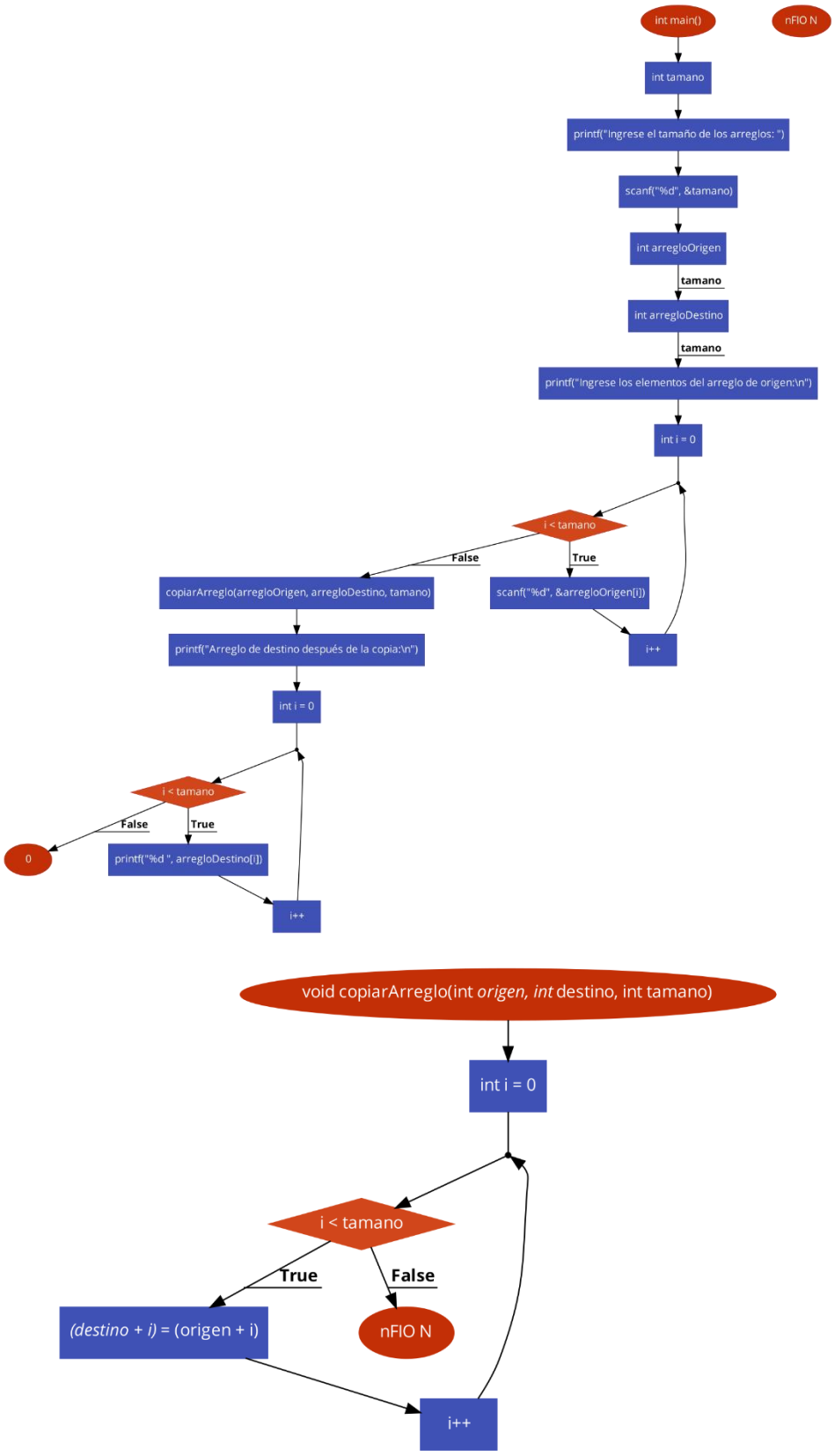
https://github.com/keevin-21/KAGT_Lenguaje_C_932

DIAGRAMA DE FLUJO

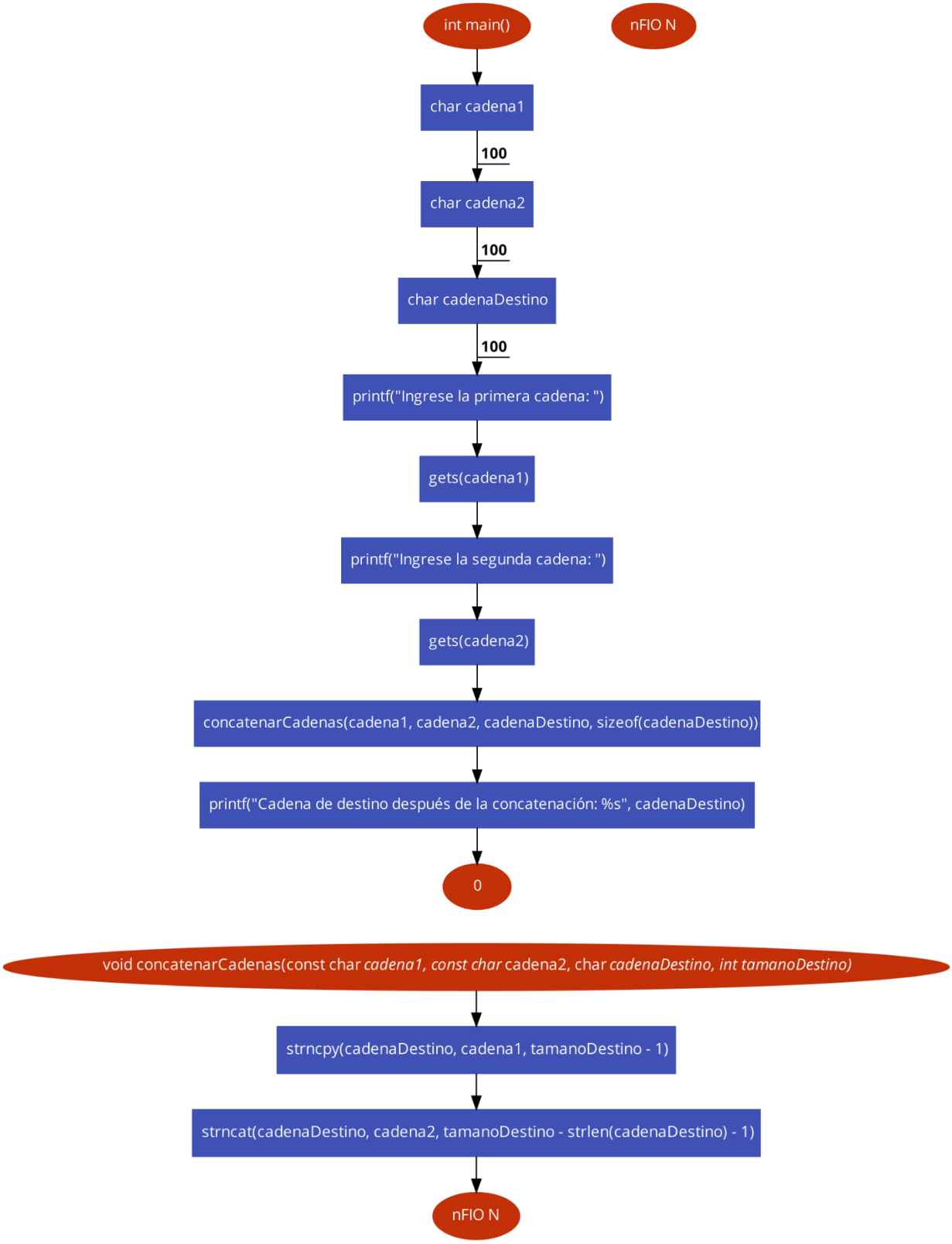
Actividad 1:



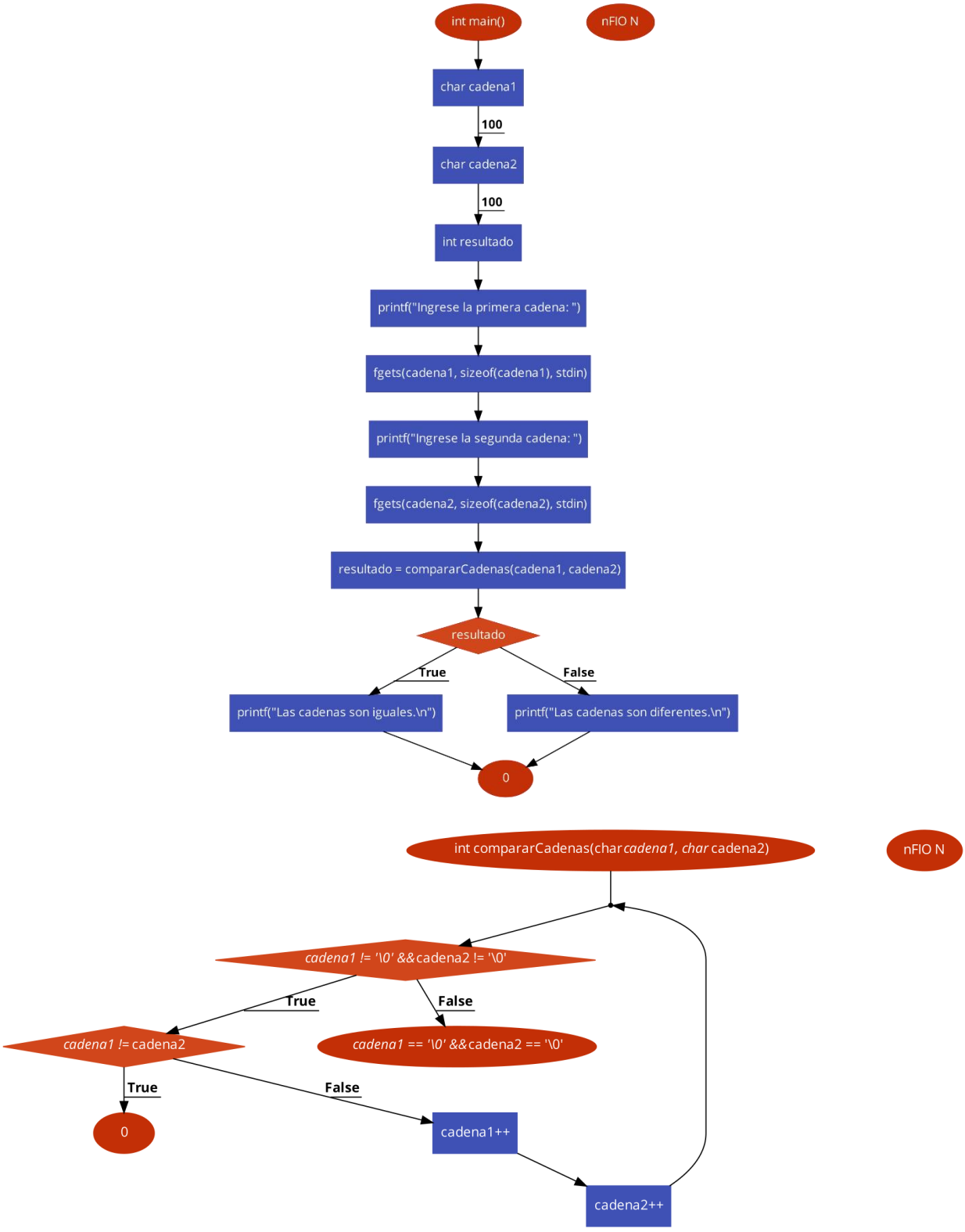
Actividad 2:


int main()
nFIO N
int tamano
printf("Ingrese el tamaño de los arreglos: ")
scanf("%d", &tamano)
int arregloOrigen
tamano
int arregloDestino
tamano
printf("Ingrese los elementos del arreglo de origen:\n")
int i = 0
i < tamano
False
True
copiarArreglo(arregloOrigen, arregloDestino, tamano)
scanf("%d", &arregloOrigen[i])
printf("Arreglo de destino después de la copia:\n")
i++
int i = 0
i < tamano
False
True
0
printf("%d ", arregloDestino[i])
i++
void copiarArreglo(int origen, int destino, int tamano)
int i = 0
i < tamano
True
False
(destino + i) = (origen + i)
nFIO N
i++

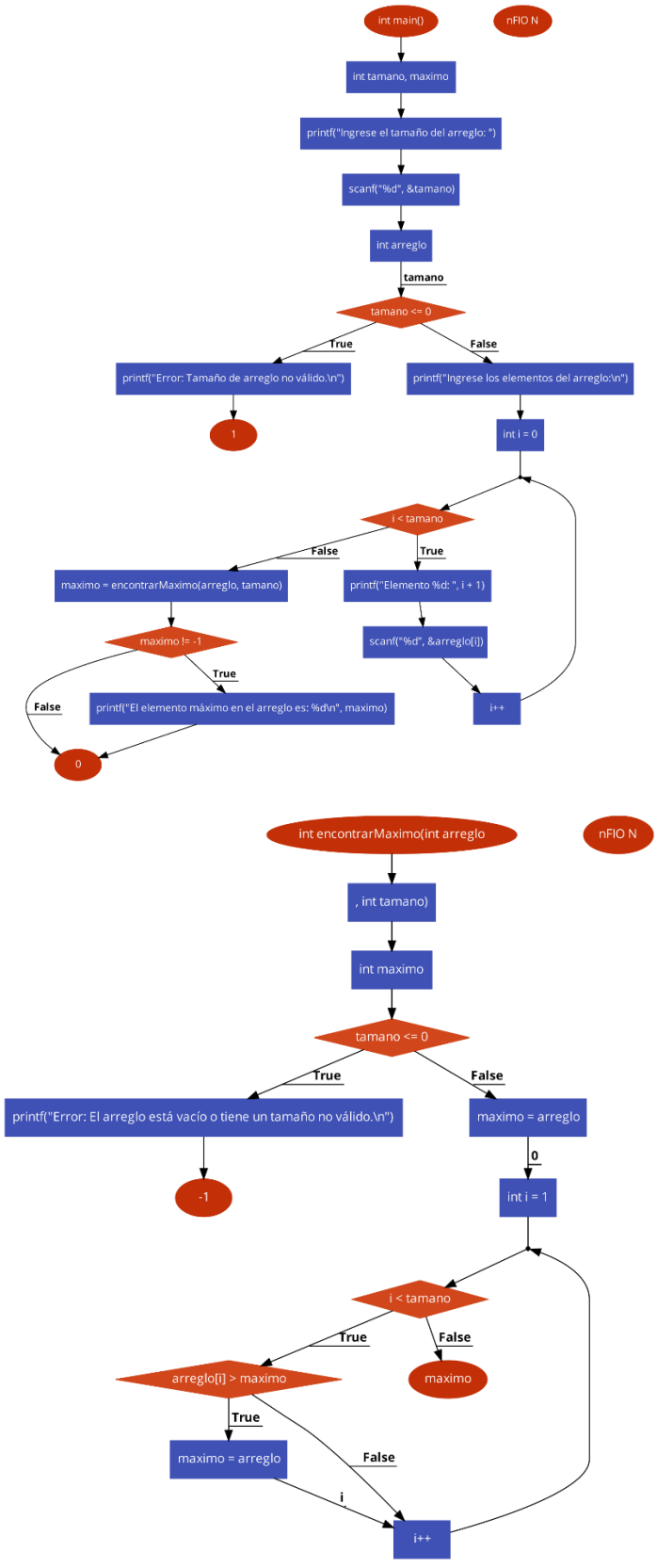
Actividad 3:


int main()
nFIO N
char cadena1
100
char cadena2
100
char cadenaDestino
100
printf("Ingrese la primera cadena: ")
gets(cadena1)
printf("Ingrese la segunda cadena: ")
gets(cadena2)
concatenarCadenas(cadena1, cadena2, cadenaDestino, sizeof(cadenaDestino))
printf("Cadena de destino después de la concatenación: %s", cadenaDestino)
0
void concatenarCadenas(const char cadena1, const char cadena2, char cadenaDestino, int tamanoDestino)
strncpy(cadenaDestino, cadena1, tamanoDestino - 1)
strncat(cadenaDestino, cadena2, tamanoDestino - strlen(cadenaDestino) - 1)
nFIO N

Actividad 4:


int main()
nFIO N
char cadena1
100
char cadena2
100
int resultado
printf("Ingrese la primera cadena: ")
fgets(cadena1, sizeof(cadena1), stdin)
printf("Ingrese la segunda cadena: ")
fgets(cadena2, sizeof(cadena2), stdin)
resultado = compararCadenas(cadena1, cadena2)
resultado
True
False
printf("Las cadenas son iguales.\n")
printf("Las cadenas son diferentes.\n")
0
int compararCadenas(char cadena1, char cadena2)
nFIO N
cadena1 != '\0' && cadena2 != '\0'
True
False
cadena1 != cadena2
cadena1 == '\0' && cadena2 == '\0'
True
False
0
cadena1++
cadena2++

Actividad 5:


int main()
nFIO N
int tamano, maximo
printf("Ingrese el tamaño del arreglo: ")
scanf("%d", &tamano)
int arreglo
tamano
tamano <= 0
True
False
printf("Error: Tamaño de arreglo no válido.\n")
printf("Ingrese los elementos del arreglo:\n")
1
int i = 0
i < tamano
False
True
maximo = encontrarMaximo(arreglo, tamano)
printf("Elemento %d: ", i + 1)
maximo != -1
scanf("%d", &arreglo[i])
True
False
printf("El elemento máximo en el arreglo es: %d\n", maximo)
i++
0
int encontrarMaximo(int arreglo
nFIO N
, int tamano)
int maximo
tamano <= 0
True
False
printf("Error: El arreglo está vacío o tiene un tamaño no válido.\n")
maximo = arreglo
0
-1
int i = 1
i < tamano
True
False
arreglo[i] > maximo
maximo
True
maximo = arreglo
False
i
i++

EJERCICIO_1()

{

Variables:

```
5   int main()
6   {
7       int tamano;
8       int arreglo[tamano];
9       int suma;
```

```
21  int calcularSuma(int *arreglo, int tamano)
22  {
23      int suma = 0;
```

Proceso principal:

```
11      printf("Ingrese el tamaño del arreglo: ");
12      scanf("%d", &tamano);
13
14      suma = calcularSuma(arreglo, tamano);
15
16      printf("La suma de los elementos del arreglo es: %d\n", suma);
```

```
23      int suma = 0;
24
25      for (int i = 0; i < tamano; i++) {
26          printf("Ingrese el valor para el elemento %d: ", i + 1);
27          scanf("%d", arreglo + i);
28          suma += *(arreglo + i);
29      }
```

Variables:

- tamano: Almacena el tamaño del arreglo que el usuario ingresará.
- arreglo: Declara un arreglo de enteros llamado arreglo, pero en este punto, no se le ha asignado un tamaño válido.
- suma: Se utilizará para almacenar la suma de los elementos del arreglo.

Proceso Principal:

El programa en C comienza por incluir la biblioteca estándar de entrada/salida y declara una función llamada calcularSuma. Luego, en la función main, el usuario ingresa el tamaño de un arreglo, el cual se almacena en la variable tamano. A continuación, se llama a la función calcularSuma, pasando un arreglo vacío arreglo y el tamaño ingresado por el usuario, para ingresar los valores de los elementos del arreglo, calcular su suma y almacenarla en la variable suma. Finalmente, se muestra la suma de los elementos del arreglo en la pantalla y el programa termina con un valor de retorno de 0. Sin embargo, cabe destacar que el código tiene un problema en la declaración del arreglo antes de conocer su tamaño, lo que podría provocar errores en tiempo de ejecución, ya que el arreglo debería declararse después de obtener el tamaño del usuario.

}

EJERCICIO_2()

{

Variables:

```
5   int main()
6   {
7       int tamano;
8       int arregloOrigen[tamano];
9       int arregloDestino[tamano];
```

Proceso principal:

```
5   int main()
6   {
7       int tamano;
8       int arregloOrigen[tamano];
9       int arregloDestino[tamano];
10
11      printf("Ingrese el tamaño de los arreglos: ");
12      scanf("%d", &tamano);
13
14      printf("Ingrese los elementos del arreglo de origen:\n");
15      for (int i = 0; i < tamano; i++)
16      {
17          scanf("%d", &arregloOrigen[i]);
18      }
19
20      copiarArreglo(arregloOrigen, arregloDestino, tamano);
21
22      printf("Arreglo de destino después de la copia:\n");
23      for (int i = 0; i < tamano; i++)
24      {
25          printf("%d ", arregloDestino[i]);
26      }
27
28      return 0;
29  }
```

```
31  void copiarArreglo(int *origen, int *destino, int tamano)
32  {
33      for (int i = 0; i < tamano; i++)
34      {
35          *(destino + i) = *(origen + i);
36      }
37  }
38
```

Variables:

- tamano; Esta variable almacena el tamaño de los arreglos que el usuario ingresará. Es un entero que representa la cantidad de elementos que contendrán los arreglos.
- arregloOrigen[tamano] y arregloDestino[tamano]: Estas son declaraciones de arreglos de enteros. arregloOrigen y arregloDestino son arreglos de tamaño tamano, que se define después de que el usuario ingrese el valor.
- i: Esta variable se usa como contador en los bucles for para iterar a través de los elementos de los arreglos. Se utiliza para recorrer los elementos en los bucles que leen y copian valores entre los arreglos.

Proceso Principal:

El código en C comienza por incluir la biblioteca estándar de entrada/salida (stdio.h). Luego, en la función main, el usuario ingresa el tamaño de dos arreglos, que se almacenan en la variable tamano. Se declaran dos arreglos, arregloOrigen y arregloDestino, de acuerdo con el tamaño ingresado. Luego, se solicita al usuario ingresar elementos para el arreglo de origen, que se almacenan en arregloOrigen. A continuación, se llama a la función copiarArreglo, pasando arregloOrigen y arregloDestino junto con el tamaño, para copiar los elementos del arreglo de origen en el arreglo de destino. Finalmente, se muestra el contenido del arreglo de destino en la pantalla. La función copiarArreglo realiza la copia de elementos entre los dos arreglos mediante el uso de punteros.

}

EJERCICIO_3()

{

Variables:

```
8       char cadena1[100];
9       char cadena2[100];
10      char cadenaDestino[100];
```

Proceso principal:

```
6   int main()
7   {
8       char cadena1[100];
9       char cadena2[100];
10      char cadenaDestino[100];
11
12      printf("Ingrese la primera cadena: ");
13      gets(cadena1);
14      printf("Ingrese la segunda cadena: ");
15      gets(cadena2);
16
17      concatenarCadenas(cadena1, cadena2, cadenaDestino, sizeof(cadenaDestino));
18
19      printf("Cadena de destino después de la concatenación: %s", cadenaDestino);
20
21      return 0;
22  }
23
24  void concatenarCadenas(const char *cadena1, const char *cadena2, char *cadenaDestino, int tamanoDestino)
25  {
26      strncpy(cadenaDestino, cadena1, tamanoDestino - 1);
27      strncat(cadenaDestino, cadena2, tamanoDestino - strlen(cadenaDestino) - 1);
28  }
29
```

Variables:

- char cadena1[100], cadena2[100], cadenaDestino[100]: Estas son declaraciones de arreglos de caracteres que almacenan las cadenas de texto. Cada uno de estos arreglos tiene un tamaño de 100 caracteres, lo que permite almacenar cadenas de hasta 99 caracteres de longitud, ya que el último carácter generalmente se usa para el carácter nulo de terminación ('\0') que indica el final de la cadena.
- char *cadena1 y char *cadena2: Estos son punteros a cadenas de caracteres. Los argumentos cadena1 y cadena2 de la función son punteros a cadenas de

caracteres constantes. Esto significa que no se pueden modificar las cadenas de caracteres a las que apuntan estos punteros.

- *cadenaDestino: Este es un puntero a una cadena de caracteres en la que se almacenará el resultado de la concatenación de cadena1 y cadena2. El puntero es no constante, lo que significa que se permite modificar la cadena de destino.
- tamanoDestino: Este argumento especifica el tamaño del arreglo al que apunta cadenaDestino. Se usa para asegurarse de que no se exceda el tamaño del arreglo al realizar la concatenación.

Proceso Principal:

El proceso principal implica que el usuario ingrese dos cadenas, que se almacenan en cadena1 y cadena2. Luego, se llama a la función concatenarCadenas con estas cadenas y el tamaño de cadenaDestino. La función utiliza las funciones strncpy y strncat para concatenar las cadenas de manera segura, asegurándose de que no se exceda el tamaño del arreglo de destino. Finalmente, se muestra el resultado de la concatenación en la pantalla. Sin embargo, cabe señalar que la función gets utilizada para la entrada de cadenas es propensa a desbordamientos de búfer y no se recomienda su uso en aplicaciones de producción debido a problemas de seguridad.

}

EJERCICIO_4()

{

Variables:

```
7       char cadena1[100];
8       char cadena2[100];
9       int resultado;
```

Proceso principal:

```
5   int main()
6   {
7       char cadena1[100];
8       char cadena2[100];
9       int resultado;
10
11      printf("Ingrese la primera cadena: ");
12      fgets(cadena1, sizeof(cadena1), stdin);
13
14      printf("Ingrese la segunda cadena: ");
15      fgets(cadena2, sizeof(cadena2), stdin);
16
17      resultado = compararCadenas(cadena1, cadena2);
18
19      if (resultado) {
20          printf("Las cadenas son iguales.\n");
21      } else {
22          printf("Las cadenas son diferentes.\n");
23      }
24
25      return 0;
26  }
27
28  int compararCadenas(char *cadena1, char *cadena2)
29  {
30      while (*cadena1 != '\0' && *cadena2 != '\0') {
31          if (*cadena1 != *cadena2) {
32              return 0;
33          }
34          cadena1++;
35          cadena2++;
36      }
37
38      return (*cadena1 == '\0' && *cadena2 == '\0');
39  }
```

Variables:

- cadena1[100] y cadena2[100]: Estos son arreglos de caracteres que almacenan las cadenas de texto ingresadas por el usuario. Tienen una capacidad para almacenar cadenas de hasta 99 caracteres.
- resultado: Esta variable almacena el resultado de la comparación de las cadenas, con un valor de 1 si son iguales y 0 si son diferentes.

Proceso Principal:

En el proceso principal, el usuario ingresa dos cadenas de texto utilizando fgets para asegurar una entrada segura. Luego, se llama a la función compararCadenas con las dos cadenas como argumentos. La función compararCadenas compara las cadenas carácter por carácter y devuelve 1 si son idénticas y 0 si son diferentes. Según el resultado, se muestra un mensaje indicando si las cadenas son iguales o diferentes. El código utiliza un bucle para recorrer las cadenas y comparar los caracteres, deteniéndose si encuentra un carácter diferente o si llega al final de ambas cadenas.

}

EJERCICIO_5()

{

Variables:

```
7       int tamano, maximo;
8       int arreglo[tamano];
```

```
35      int maximo;
```

Proceso principal:

```c
int main()
{
    int tamano, maximo;
    int arreglo[tamano];

    printf("Ingrese el tamaño del arreglo: ");
    scanf("%d", &tamano);

    if (tamano <= 0) {
        printf("Error: Tamaño de arreglo no válido.\n");
        return 1;
    }

    printf("Ingrese los elementos del arreglo:\n");
    for (int i = 0; i < tamano; i++) {
        printf("Elemento %d: ", i + 1);
        scanf("%d", &arreglo[i]);
    }

    maximo = encontrarMaximo(arreglo, tamano);

    if (maximo != -1) {
        printf("El elemento máximo en el arreglo es: %d\n", maximo);
    }

    return 0;
}

int encontrarMaximo(int arreglo[], int tamano)
{
    int maximo;

    if (tamano <= 0) {
        printf("Error: El arreglo está vacío o tiene un tamaño no válido.\n");
        return -1;
    }

    maximo = arreglo[0];

    for (int i = 1; i < tamano; i++) {
        if (arreglo[i] > maximo) {
            maximo = arreglo[i];
        }
    }

    return maximo;
}
```

Variables:

- tamano y maximo: Estas variables almacenan el tamaño del arreglo ingresado por el usuario y el valor máximo encontrado en el arreglo, respectivamente.
- arreglo[tamano]: Este es un arreglo de enteros, pero hay un problema aquí. En C, no puedes declarar un arreglo con un tamaño variable de esta manera. Deberías declarar el arreglo después de conocer el tamaño ingresado por el usuario. Esta declaración no es válida en C y puede causar comportamientos inesperados.

Proceso Principal:

En el proceso principal, el usuario ingresa el tamaño del arreglo, y si el tamaño es menor o igual a cero, se muestra un mensaje de error y el programa termina. Luego, se le pide al usuario que ingrese los elementos del arreglo. Luego, se llama a la función encontrarMaximo con el arreglo y su tamaño como argumentos para encontrar el elemento máximo en el arreglo. Si el resultado de la función no es -1 (lo que indica un error o un arreglo vacío), se muestra el valor máximo. Sin embargo, como mencioné, el código tiene un problema en la declaración del arreglo antes de conocer su tamaño, lo que puede provocar errores en tiempo de ejecución. El arreglo debería declararse después de obtener el tamaño del usuario para que el código sea válido.

}