UNIVERSIDAD AUTONOMA DE BAJA CALIFORNIA
FACULTAD DE INGENIERIA, ARQUITECTURA Y DISEÑO

INGENIERIA EN SOFTWARE Y TECNOLOGIAS EMERGENTES

GRUPO: 932

MATERIA: Lenguaje C

MAESTRA: Yulith Vanessa Altamirano Flores

TITULO:

Práctica6: Apuntadores y Argumentos

ALUMNA: *Teresa Rivas Gómez*

MATRICULA: *372565*

CODIGO:

INFORMACIÓN

```
/*
NOMBRE DEL ARCHIVO: Practica6_ApuntadoresYArgumentos_RivasGomezTeresa
AUTOR: Teresa Rivas Gomez
FECHA DE CREACION: Oct - 19 - 2023
DESCRIPCION: Crear un programa en C que realice operaciones básicas en arreglos utilizando
apuntadores y funciones. Deberás crear un menú interactivo que permita al usuario elegir
entre varias operaciones en un arreglo. Las operaciones incluirán la suma de elementos de un
arreglo, la copia de arreglos, la concatenación de arreglos y la comparación de arreglos.
*/

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

FUNCIONES UTILIZADAS

```
int msges();
void menu();
int suma(int *arreglo, int longitud);
void copia(int *a_origen, int *a_destino, int longitud);
void concatenacion(int *A_Entrada1, int *A_Entrada2, int *resultado, int longitud1, int
longitud2);
int comparacion(int *arreglo1, int *arreglo2, int longitud);
int encontrar(int *arreglo, int longitud);
int ValidarCadena(char mensj[], int ri, int rf);
```

MENU Y MENSAJES PARA EL USUARIO

```
int main()
{
    menu();
    return 0;
}
int msges()
{
    int op;
    printf("\n M E N U : Practica 6 \n");
    printf("1.- Suma de elementos en un arreglo.\n");
    printf("2.- Copia de arreglos.\n");
    printf("3.- Concatenacion de arreglos.\n");
    printf("4.- Comparacion de arreglos.\n");
    printf("5.- Encontrar el elemento maximo.\n");
    printf("0.- Salir del menu.\n");
    op = ValidarCadena("Seleccione una opcion del 0 al 5", 0, 5);
    return op;
}
```

VARIABLES QUE SE UTILIZARON PARA LA ELABORACION DE CADA CASO

```
void menu()
{
    int op;
    int arreglo1[] = {1, 2, 3, 4, 5};
    int arreglo2[] = {6, 7, 8, 9, 0};
    int longitud1 = sizeof(arreglo1) / sizeof(int);
    int longitud2 = sizeof(arreglo2) / sizeof(int);
    int longitud;
    int resultado;
    int resultado1[20];
    int maximo;
    do{
        system("CLS");
        op = msges();
```

```
        switch(op)
        {
```

SE DEFINE QUE EL RESULTADO ES LA FUNCION DE SUMA

```
            case 1:
                printf("1.- Suma de elementos en un arreglo.\n");
                printf("Arreglo 1: 1, 2, 3, 4, 5.\n");
                resultado = suma(arreglo1, longitud1);
                printf("La suma de los elemntos en el arreglo es: %d\n", resultado);
                system("PAUSE");
                break;
```

SE LLAMA A LA FUNCION DE "COPIA" Y SE USAN CICLOS PARA CADA ARREGLO, ESTO PARA IMPRIMIRLOS COMPLETOS Y POSTERIORMENTE COPIAR EL DE ORIGEN AL DE DESTINO

```
            case 2:
                printf("2.- Copia de arreglos.\n");
                printf("Arreglo 1: 1, 2, 3, 4, 5.\n");
                printf("Arreglo 2: 6, 7, 8, 9, 0.\n");
                copia(arreglo1, arreglo2, longitud1);
                printf("Arreglo de origen: \n");
                for (int i = 0; i < longitud1; i++)
                {
                    printf("%d, ", arreglo1[i]);
                }
                printf("\n");
                printf("Arreglo de destino (copia): \n");
                for (int i = 0; i < longitud1; i++)
                {
                    printf("%d, ", arreglo2[i]);
                }
                system("PAUSE");
                break;
```

SE DEFINE QUE LA LONGITUD TOTAL ES LA SUMA DE AMBAS LONGITUDES DE ARREGLOS, SE USAN DE NUEVO DOS CICLOS PARA IMPRIMIR LOS ARREGLOS Q Y 2 Y UNO TERCERO PARA IMPRIMIR EL RESULTADO QUE SE ALMACENA EN EL ARREGLO DESTINO

```
            case 3:
                printf("3.- Concatenacion de arreglos.\n");
                printf("Arreglo 1: 1, 2, 3, 4, 5.\n");
                printf("Arreglo 2: 6, 7, 8, 9, 0.\n");
                longitud = longitud1 + longitud2;
                concatenacion(arreglo1, arreglo2, resultado1, longitud1, longitud2);
                printf("Arreglo 1: ");
                for (int i = 0; i < longitud1; i++)
                {
                    printf("%d ", arreglo1[i]);
                }
                printf("\n");
                printf("\nArreglo 2: ");
                for (int i = 0; i < longitud2; i++)
                {
                    printf("%d ", arreglo2[i]);
                }
                printf("\n");
                printf("\nArreglo de destino: ");
                for (int i = 0; i < longitud; i++)
                {
```

```
                printf("%d ", resultado1[i]);
            }
            system("PAUSE");
            break;
```

SE HACE UNA CONDICION DONDE SI LA LONGITUS DEL ARREGLO 1 ES DIFERENTE DE LA DEL ARREGLO 2, SE DICE QUE NO SON IGUALES Y POR LO CONTRARIO SI SON IGUALES SE HACE UNA COMPARACION CON LA FUNCION POSTERIORMENTE MOSTRADA

```
        case 4:
            printf("4.- Comparacion de arreglos.\n");
            printf("Arreglo 1: 1, 2, 3, 4, 5.\n");
            printf("Arreglo 2: 6, 7, 8, 9, 0.\n");
            if (longitud1 != longitud2)
            {
                printf("Los arreglos no tienen la misma longitud, entonces no son
iguales.\n");
            }
            else
            {
                int resultado = comparacion(arreglo1, arreglo2, longitud1);
                if (resultado)
                {
                    printf("Los arreglos son iguales.\n");
                }
                else
                {
                    printf("Los arreglos no son iguales.\n");
                }
            }
            system("PAUSE");
            break;
```

SE MANDA A LLAMAR A LA FUNCION QUE BUSCA EL NUMERO MAXIMO ENCONTRADO

```
        case 5:
            printf("5.- Encontrar el elemento maximo.\n");
            printf("Arreglo 1: 1, 2, 3, 4, 5.\n");
            maximo = encontrar(arreglo1, longitud1);
            printf("El elemento maximo en el arreglo es: %d\n", maximo);
            system("PAUSE");
            break;
        case 0:
            printf("Saliste del programa.\n");
            break;
    }
} while(op != 0);
}
```

VALIDACION DE CADENA EN EL CASO DEL MENU

```
int ValidarCadena(char mensj[], int ri, int rf)
{
    int num;
    char cadena[200];
    do
    {
        printf("%s", mensj);
        fflush(stdin);
        gets(cadena);
        num = atoi(cadena);
```

```
    } while (num < ri || num > rf);
    return num;
}
```

SUMA, FUNCION RETORNABLE

```
/* Suma de elementos en un arreglo: Crea una función que calcule la suma de los
elementos en un arreglo utilizando apuntadores y aritmética de direcciones. La función
debe tomar un apuntador al arreglo y devolver la suma. */
int suma(int *arreglo, int longitud)
{
    int suma = 0;
    int i;
    for(i = 0; i < longitud; i++)
    {
        suma += *arreglo;
        arreglo++;
    }
    return suma;
}
```

COPIA, FUNCION NO RETORNABLE

```
/* Copia de arreglos: Crea una función que copie un arreglo de origen en un arreglo de
destino. Utiliza apuntadores para realizar esta operación. La función debe tomar dos
apuntadores como argumentos, uno para el arreglo de origen y otro para el arreglo de
destino. */
void copia(int *a_origen, int *a_destino, int longitud)
{
    for (int i = 0; i < longitud; i++)
    {
        *a_destino = *a_origen;
        a_origen++;
        a_destino++;
    }
}
```

CONCATENACION, FUNCION NO RETORNABLE

```
/* Concatenación de arreglos: Crea una función que tome dos arreglos de entrada y
los concatene en un tercer arreglo. Utiliza apuntadores y aritmética de direcciones para
realizar la concatenación. La función debe tomar tres apuntadores como argumentos:
dos para los arreglos de entrada y uno para el arreglo de destino. */
void concatenacion(int *A_Entrada1, int *A_Entrada2, int *resultado, int longitud1, int
longitud2)
{
    for (int i = 0; i < longitud1; i++)
    {
        *resultado = *A_Entrada1;
        A_Entrada1++;
        resultado++;
    }

    for (int i = 0; i < longitud2; i++)
    {
        *resultado = *A_Entrada2;
        A_Entrada2++;
        resultado++;
    }
}
```

COMPARACION, FUNCION RETORNABLE

```
/* Comparación de arreglos: Crea una función que compare dos arreglos y determine
si son iguales. Utiliza apuntadores y aritmética de direcciones para realizar la
comparación. La función debe tomar dos apuntadores como argumentos y devolver
un valor que indique si los arreglos son iguales. */
int comparacion(int *arreglo1, int *arreglo2, int longitud)
```

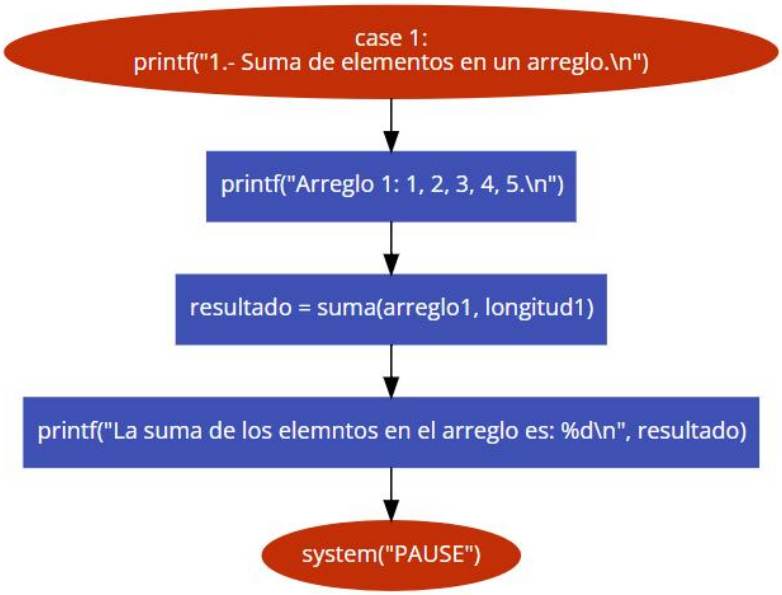
```
{
    for (int i = 0; i < longitud; i++)
    {
        if (*arreglo1 != *arreglo2)
        {
            return 0;
        }
        arreglo1++;
        arreglo2++;
    }
    return 1;
}
```

ENCONTRAR, FUNCION RETORNABLE

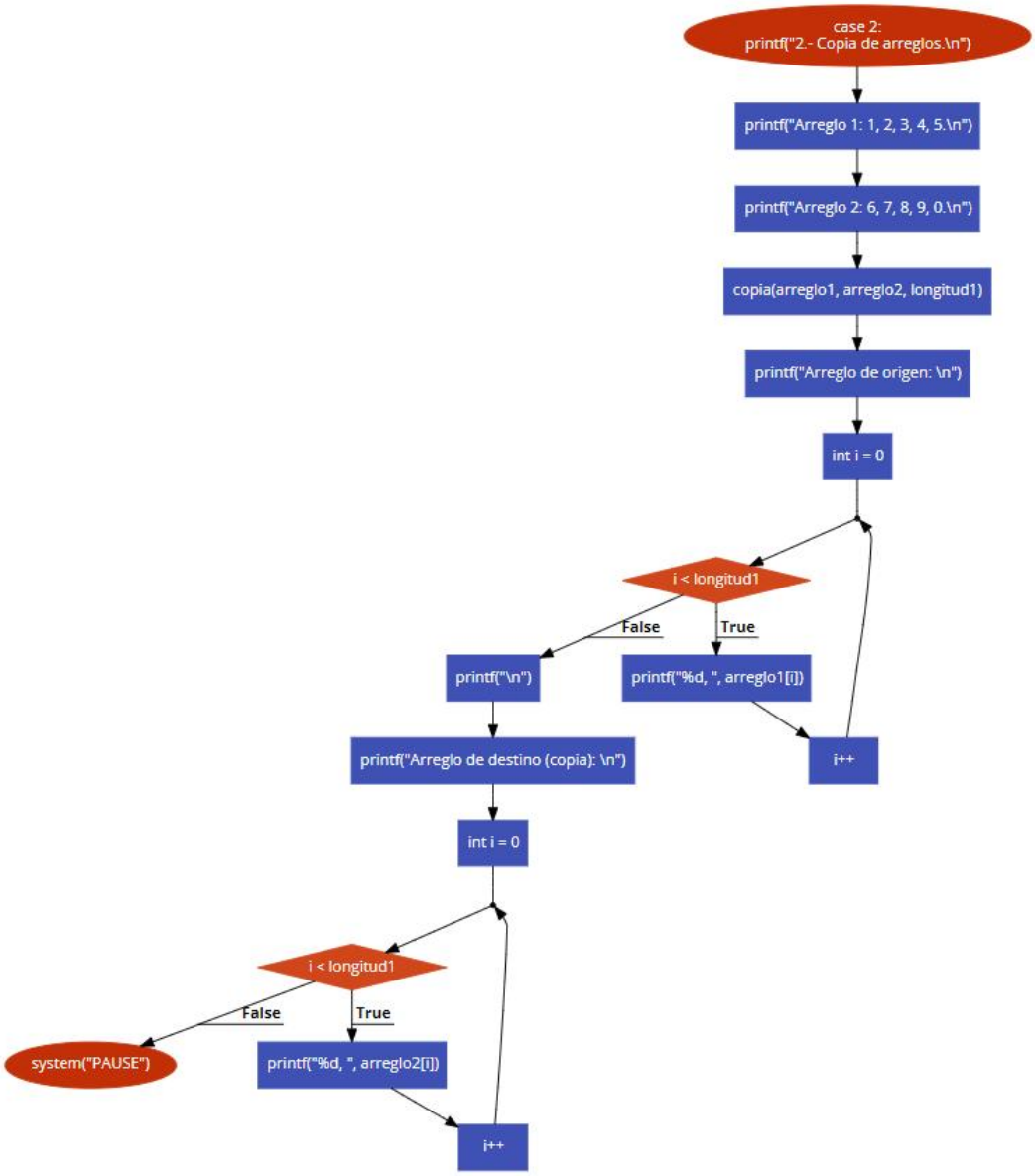
```
/* Encontrar el elemento máximo: Crea una función que encuentre y devuelva el
elemento máximo en el arreglo. */
int encontrar(int *arreglo, int longitud)
{
    int Elemento_Maximo = arreglo[0];
    for (int i = 1; i < longitud; i++)
    {
        if (arreglo[i] > Elemento_Maximo)
        {
            Elemento_Maximo = arreglo[i];
        }
    }
    return Elemento_Maximo;
}
```

DIAGRAMA DE FLUJO:

CASO 1:

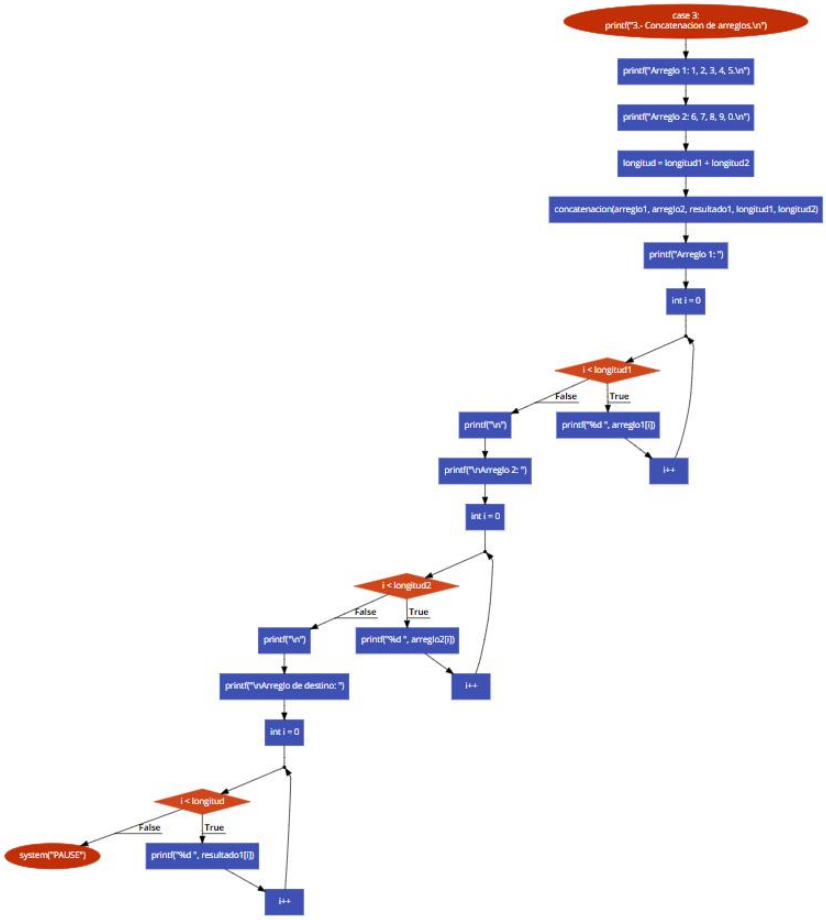


CASO 2:

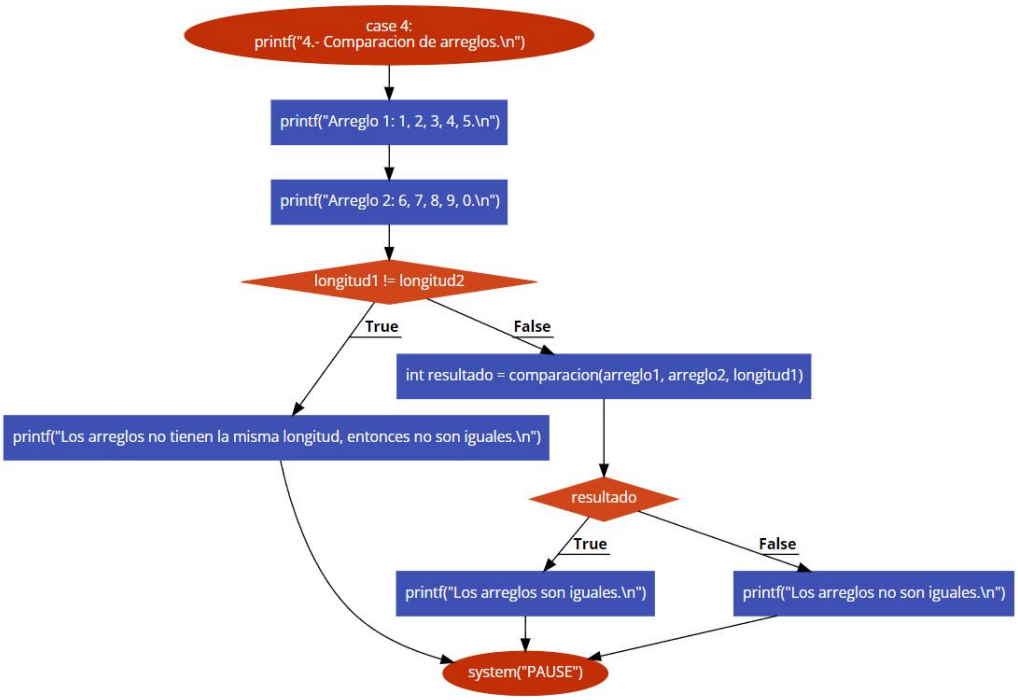


Teresa Rivas Gómez

CASO 3:

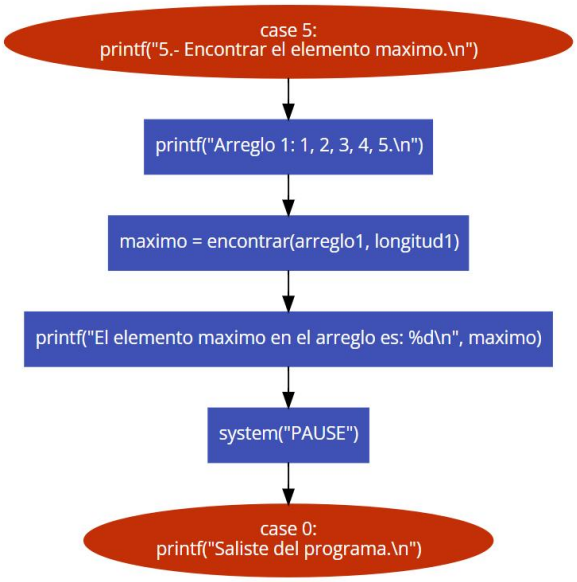


CASO 4:

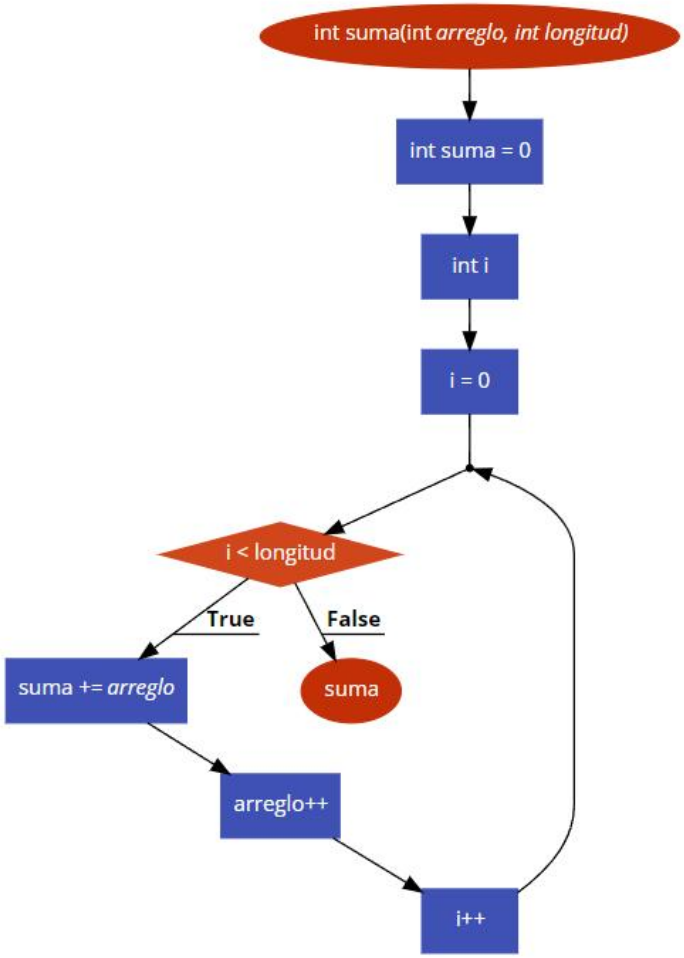


Teresa Rivas Gómez

CASO 5:

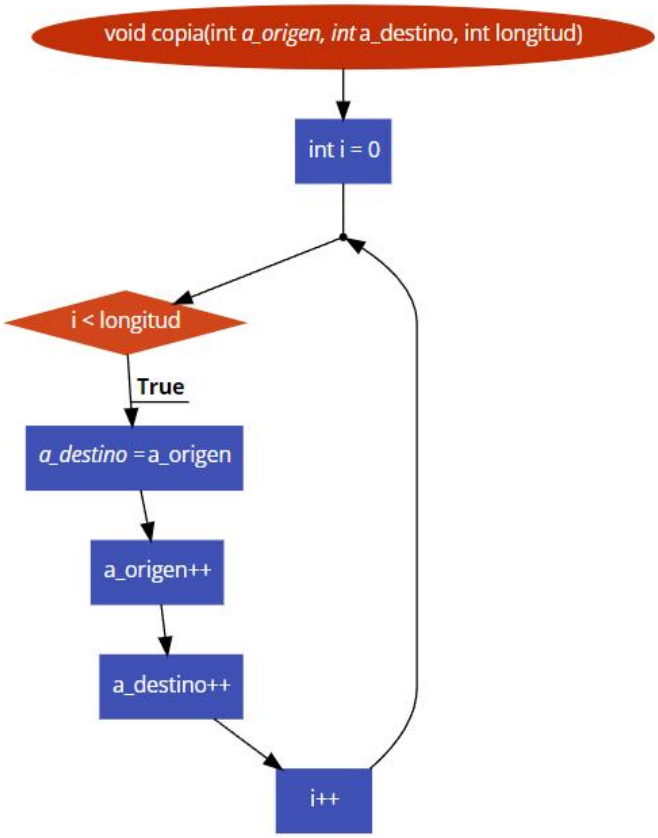


FUNCION SUMA:

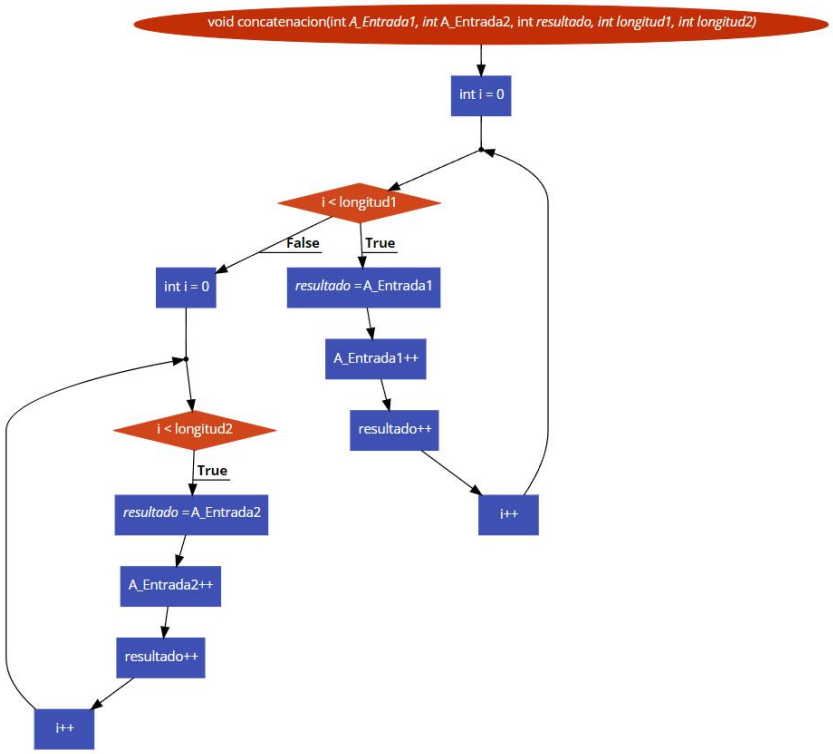


Teresa Rivas Gómez

FUNCION COPIA:

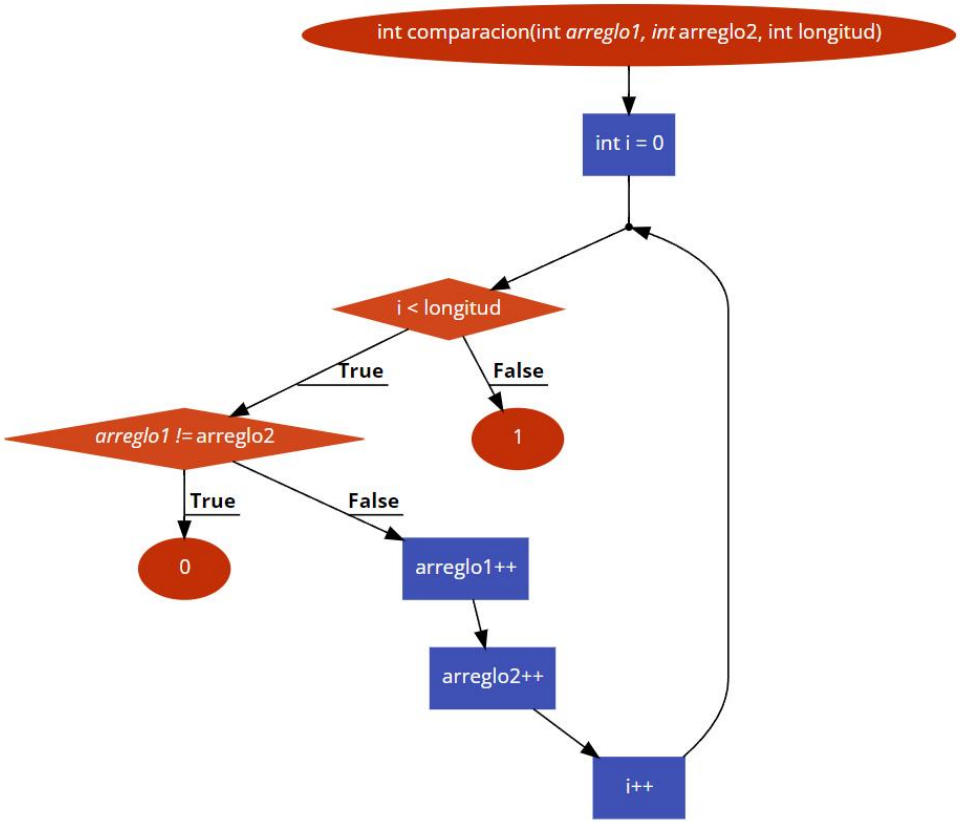


FUNCION CONCATENACION:

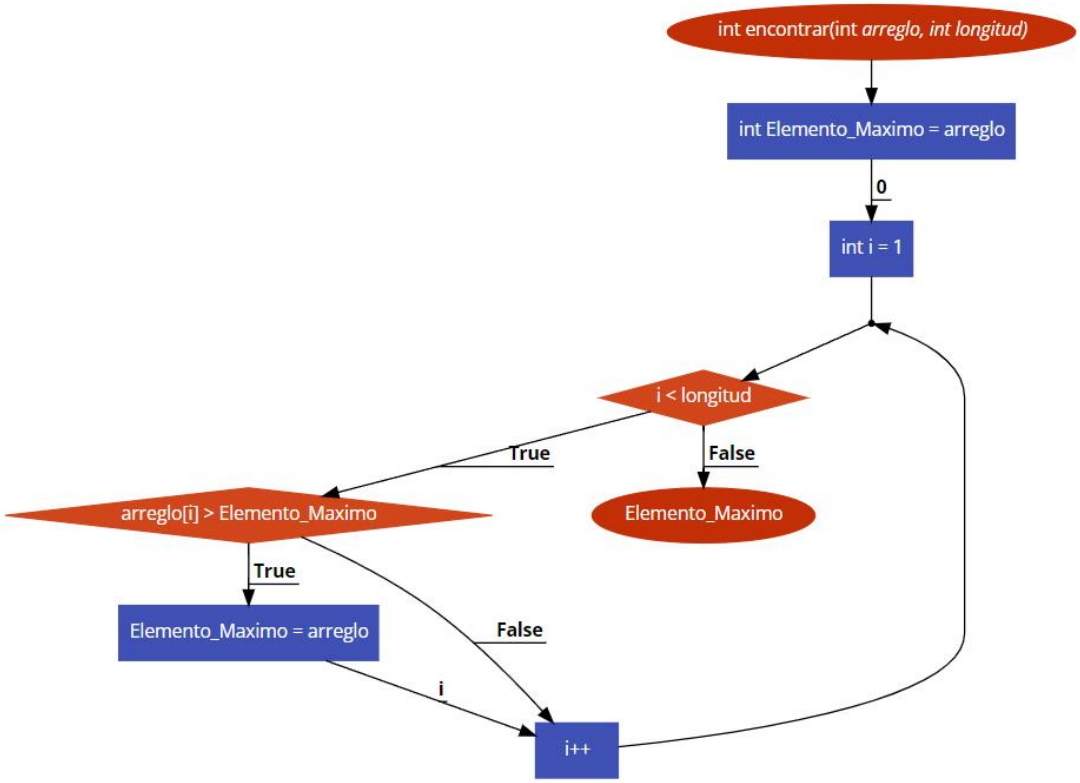


Teresa Rivas Gómez

FUNCION COMPARAR:

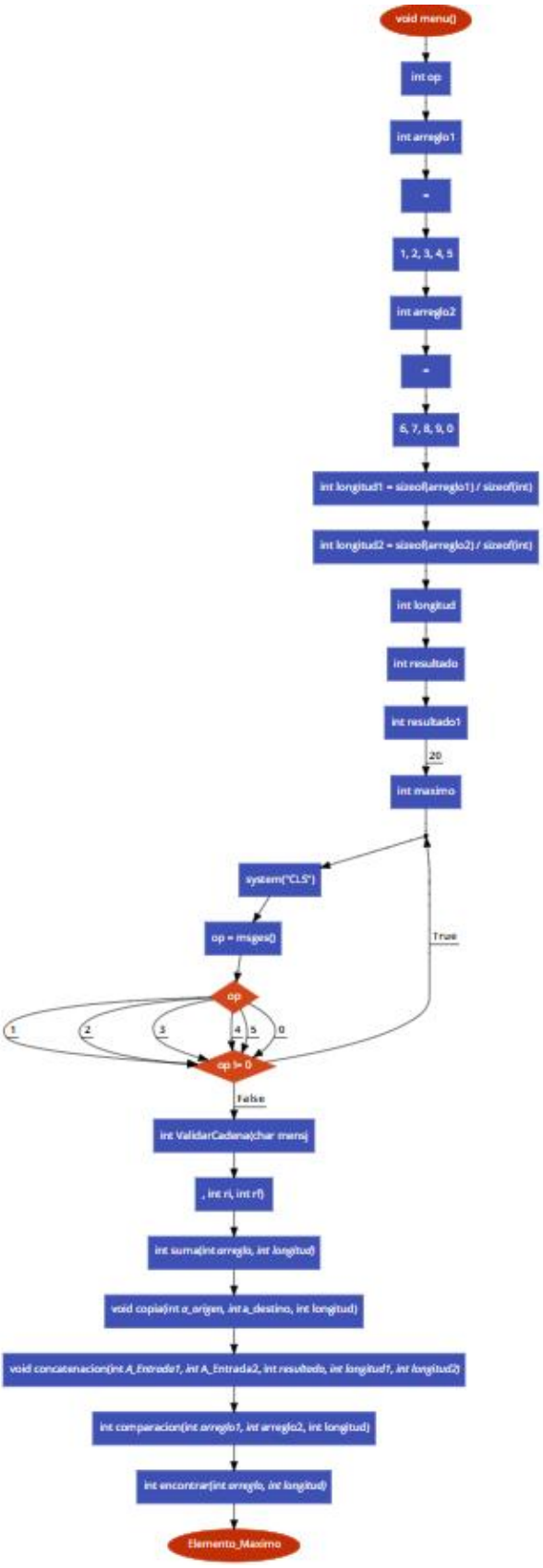


FUNCION ENCONTRAR MAXIMO:



Teresa Rivas Gómez

TODA LA ESTRUCTURA:



Teresa Rivas Gómez