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 3
Recursividad

Ciudad de México; a 01 de noviembre de 2020.

Índice

INTRODUCCIÓN

La recursividad es un tipo de modelo cíclico que consiste en una función que se llama a sí misma hasta que cumple una condición y rompe el ciclo.

En esta práctica se utilizará para realizar programas que requieren de una misma actividad repetida (cálculo de factorial, suma de una secuencia sucesiva de números, un número elevado a una potencia) hasta que cumple una determinada condición.

DESARROLLO

PROGRAMA 1. FACTORIAL

Descripción

Obtener el factorial de cualquier número con recursividad.

Análisis

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

Entradas:

int numero; valor entero donde se almacena el dato del que se quiere obtener el factorial.

Salidas:

int resultado;

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

Se solicitará al usuario que introduzca el número. Utilizando una función fuera del main (para la recursividad), la función se llama a sí misma restando un uno al número (hasta llegar a uno) en cada llamado y lo multiplica por el dato anterior. El resultado es el producto de todos los valores dados por la función.

3. **¿Qué espero de salida?**

Imprimir -> resultado o “No se puede obtener el factorial del número”.

Requerimientos Funcionales

El programa calcula el factorial de números positivos desde cero hasta doce.

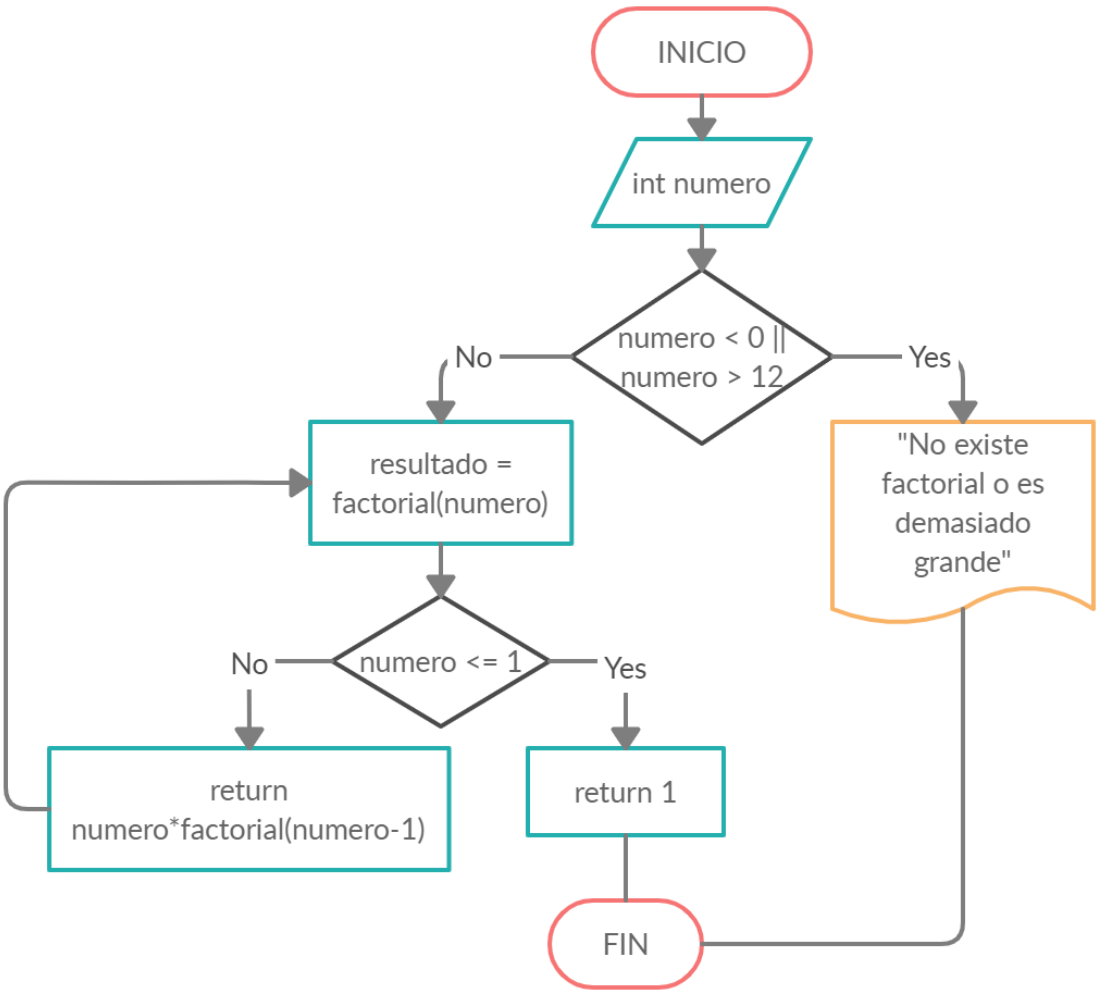
El programa muestra en pantalla el proceso del factorial imprimiendo el producto de todos los números.

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

Requerimientos no Funcionales

El programa no calcula factorial de números negativos y decimales (no existen) ni superiores a 12 (la capacidad del tipo de dato no soporta más).

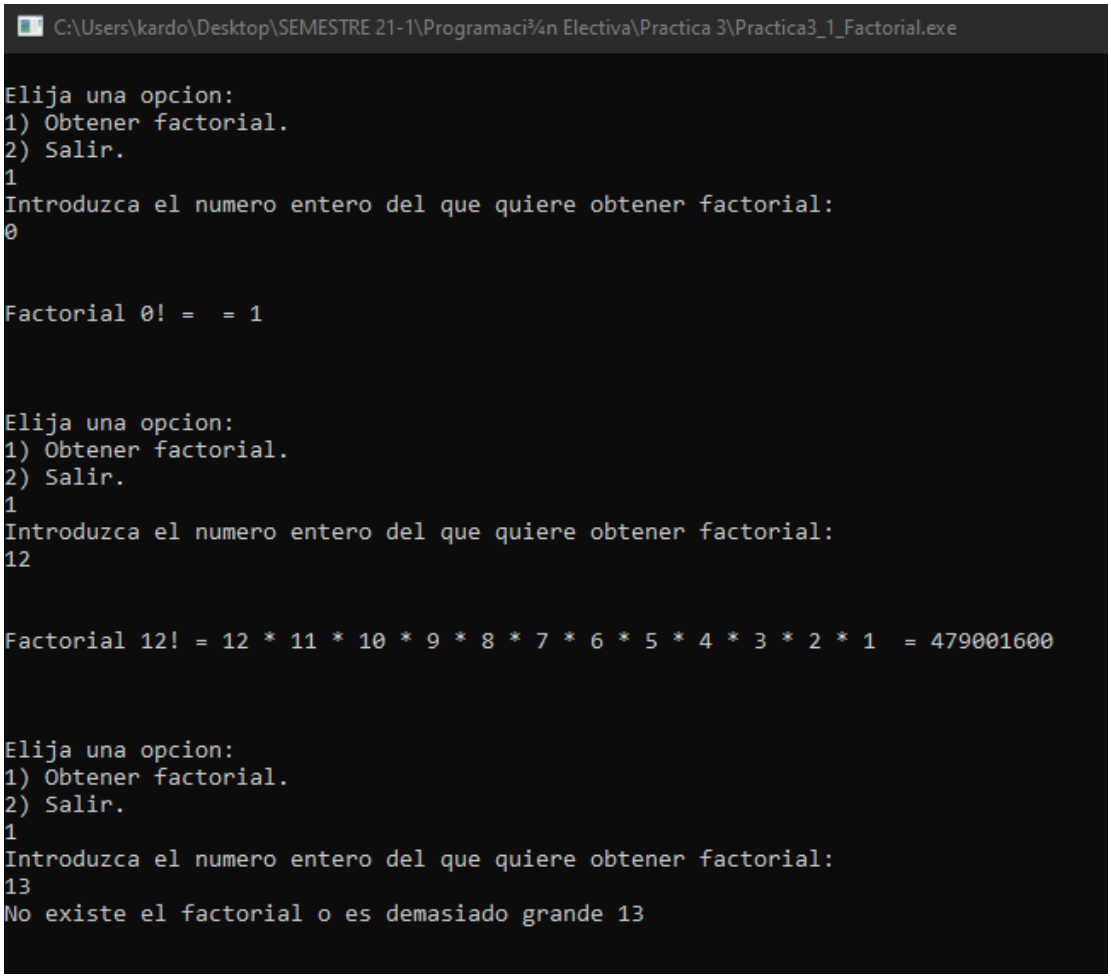
Diagrama de flujo



Código en C

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

int factorial(int numero);

int main(){
    int indice = 0, numero = 0, resultado = 0;

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

    if(indice == 1){
        printf("Introduzca el numero entero del que quiere obtener factorial:\n");
        scanf("%d",&numero);
        fflush(stdin);
        if(numero < 0 || numero > 12)
            printf("No existe el factorial o es demasiado grande %d\n\n\n",numero);
        else{
            printf("\n\nFactorial %d! = ",numero);
            resultado = factorial(numero);
            printf(" = %d\n\n\n",resultado);
        }
    }
}
    return 0;
}

int factorial(int numero){
    if(numero != 0)
    printf("%d ",numero);
    if(numero<=1)
        return 1;
    else{
        printf(" * ");
        return numero*factorial(numero-1);
    }
}
```

Resultados (compilación)



Caso 1:	Caso 2:	Caso 1:
numero = 0	numero = 12	numero = 13
factorial = 1	factorial = 479001600	“Demasiado grande”

PROGRAMA 2. SUMATORIA

Descripción

Obtener la suma de cualquier número, es decir, si pongo el valor 3. Se espera lo siguiente 6 (3+2+1).

Análisis

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

Entradas:

int numero; valor entero donde se almacena el dato del que se quiere obtener la suma.

Salidas:

int resultado;

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

Se solicitará al usuario que introduzca el número que será el mayor de la sumatoria consecutiva. Utilizando una función fuera del main (para la recursividad), la función se llama a sí misma restando un uno al número (hasta llegar a cero) en cada llamado y lo suma al dato anterior. El resultado es la suma de todos los números dados por la función.

3. **¿Qué espero de salida?**

Imprimir -> resultado o “No se puede obtener el factorial del número”.

Requerimientos Funcionales

El programa calcula la suma de números enteros positivos y consecutivos, siendo el número mayor introducido por el usuario y el menor cero.

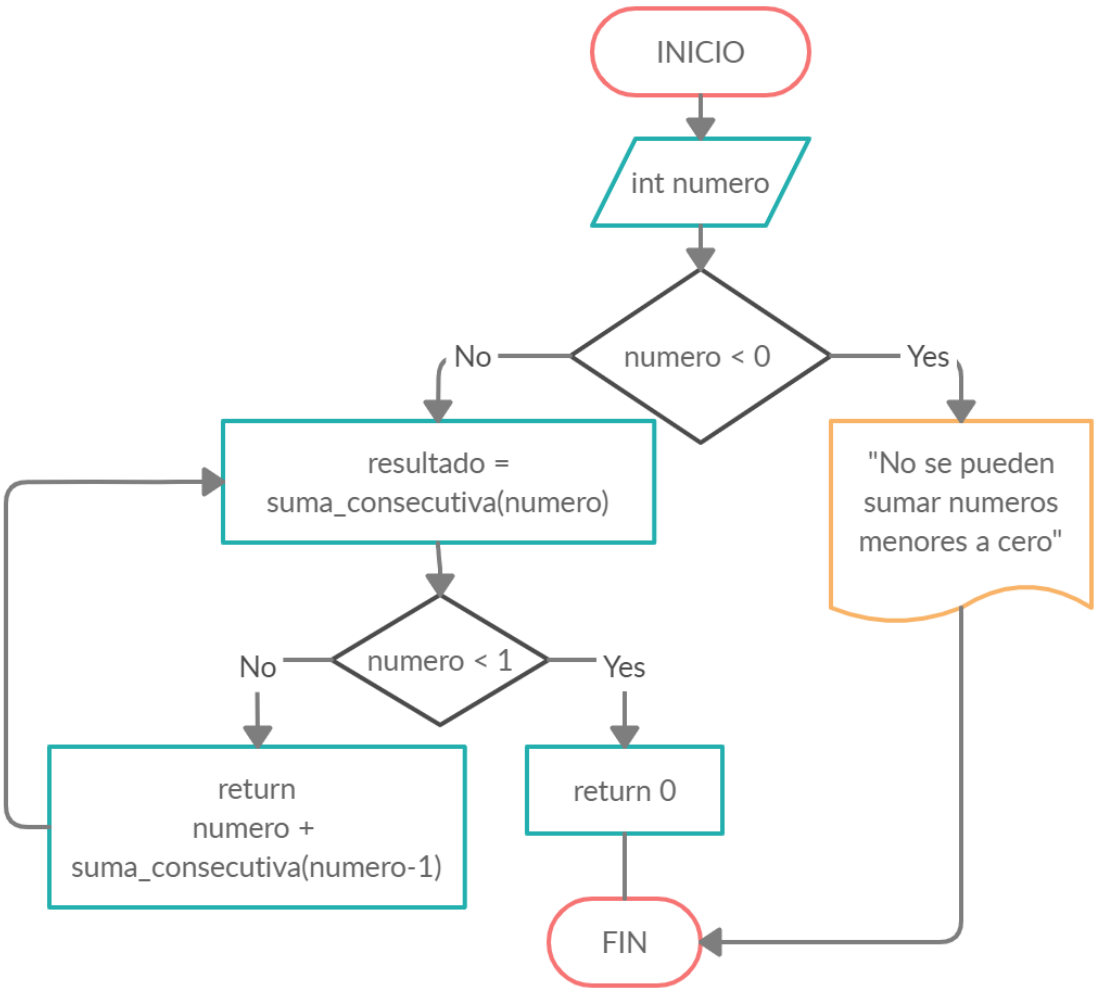
El programa muestra en pantalla la suma completa con todos los factores incluidos.

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

Requerimientos no Funcionales

El programa no admite un número negativo, decimal o flotante ya que no cumplen con los requerimientos.

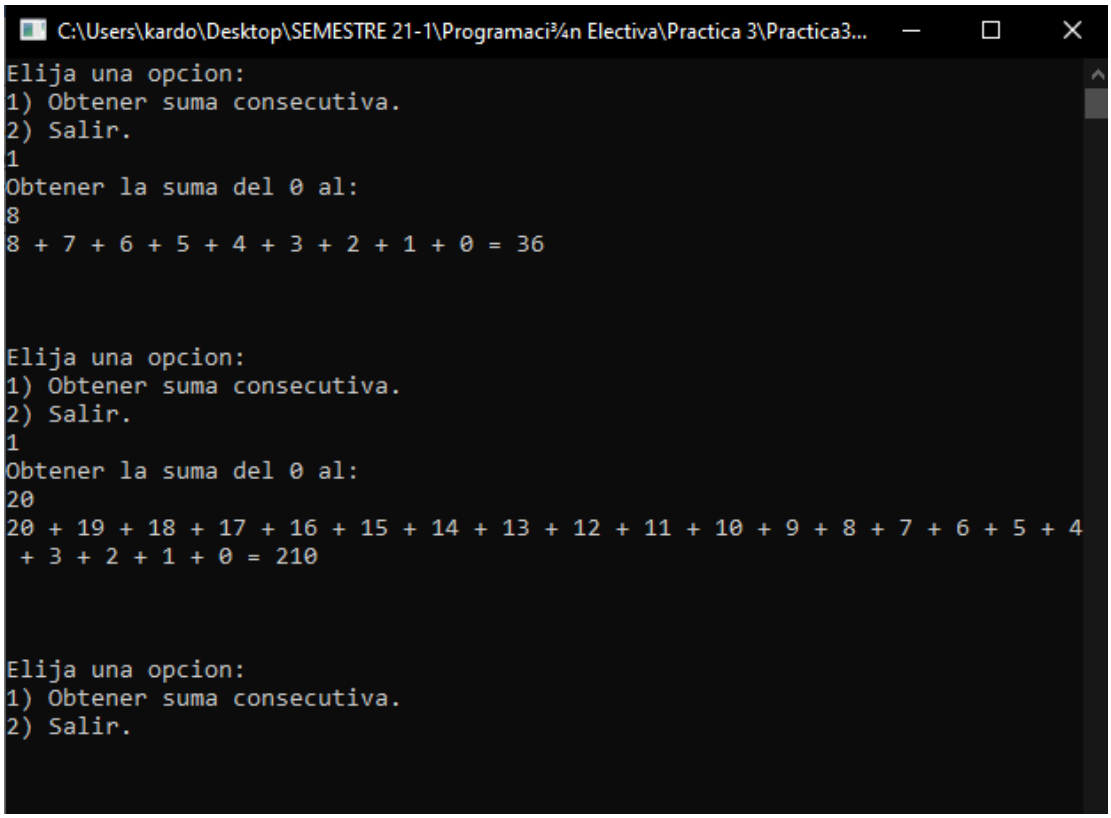
Diagrama de flujo



Código en C

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

int suma_consecutiva(int numero);

int main(){

int indice = 0, numero = 0, resultado;

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

    if(indice == 1){
        printf("Obtener la suma del 0 al:\n");
        scanf("%d",&numero);
        fflush(stdin);

        if(numero < 0)
            printf("No se pueden sumar numeros menores a cero.\n\n\n",numero);
        else{
            resultado = suma_consecutiva(numero);
            printf("= %d\n\n\n",resultado);
        }


    }
}
    return 0;
}

int suma_consecutiva(int numero){
    printf("%d ",numero);
    if(numero<1)
        return 0;
    else{
        printf("+ ");
        return numero+suma_consecutiva(numero-1);
    }
}
```

Resultados (compilación)



Caso 1:

numero = 8

resultado = 36

Caso 2:

Numero = 20

Resultado = 210

PROGRAMA 3. POTENCIA

Descripción

Obtener la potencia de cualquier número.

Análisis

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

Entradas:

float numero; valor flotante donde se almacena el número que se quiere elevar.

Int potencia; valor entero donde se almacena la potencia.

Salidas:

int resultado;

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

Se solicitará al usuario que introduzca el número que desea elevar y la potencia. Utilizando una función fuera del main (para la recursividad), la función se llama a sí misma multiplicando al número la cantidad de veces que indique la potencia, la potencia se va restando uno en cada llamado hasta que llega a uno, de ser cero, se devuelve un uno. El resultado el número multiplicado potencia-veces.

6. **¿Qué espero de salida?**

Imprimir -> resultado o “No se puede calcular potencias menores a cero”.

Requerimientos Funcionales

El programa sólo admite potencias positivas enteras (mayores o iguales a cero).

El programa permite calcular la potencia de un número entero o flotante positivo o negativo.

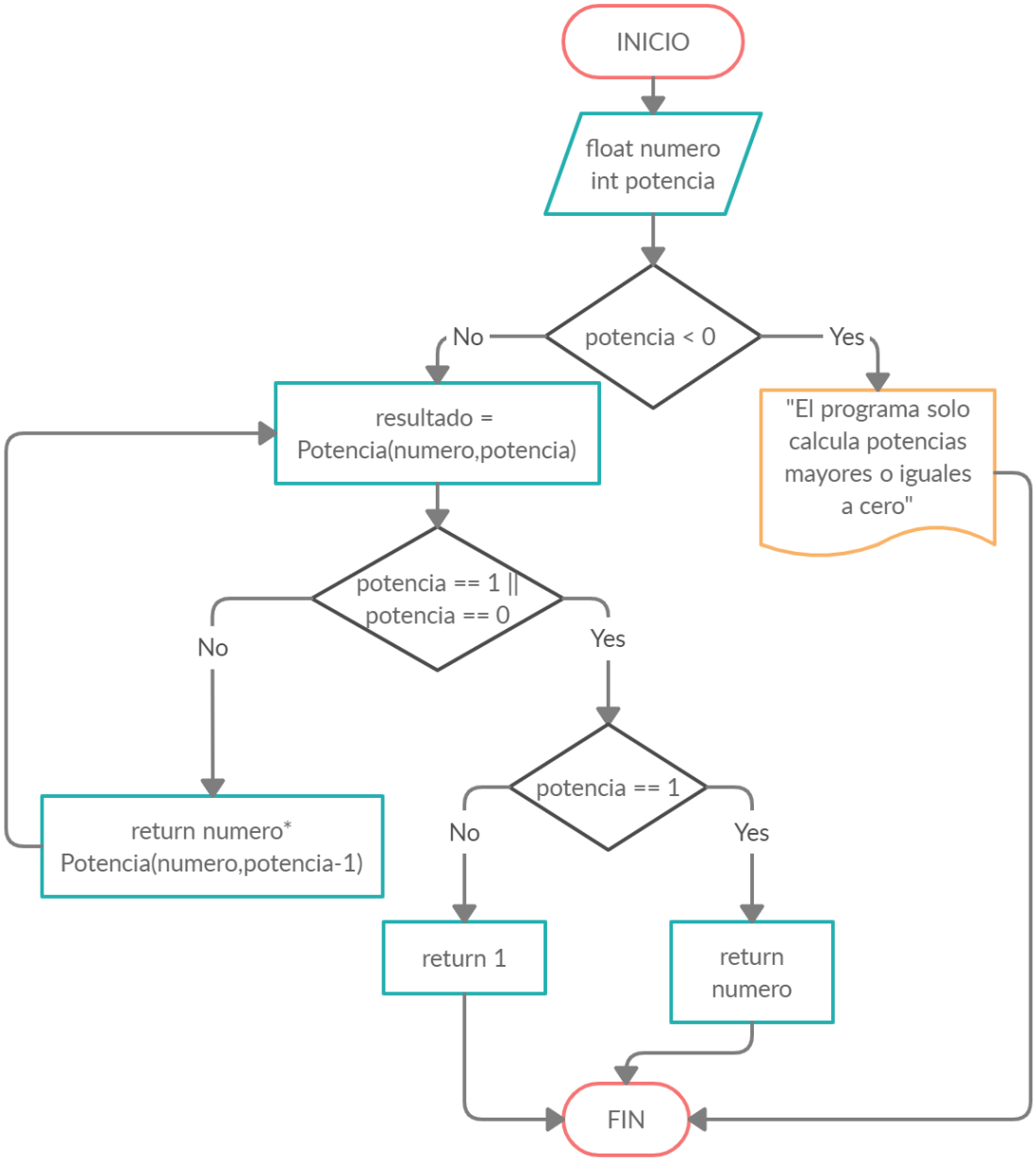
El programa termina hasta que el usuario de la opción de salir.

Requerimientos no Funcionales

El programa no admite potencias decimales ni negativas.

El programa muestra en pantalla únicamente resultados hasta con dos decimales.

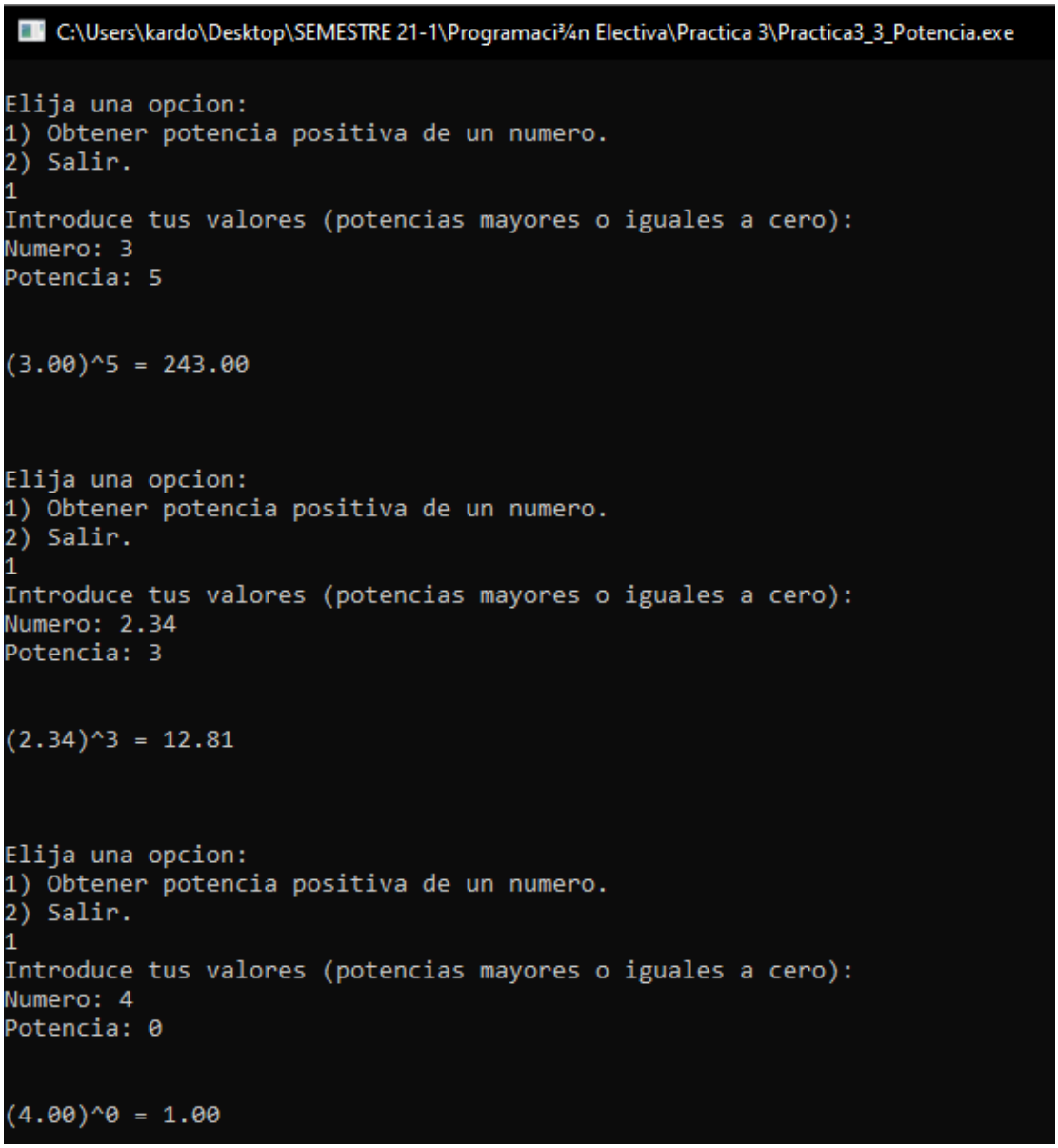
Diagrama de flujo



Código en C

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

float Potencia(float numero, int potencia);

int main(){
    float numero = 0;
    int potencia = 0, indice = 0;

    while(indice != 2){
    numero = 0;
    potencia = 0;
    printf("\nElija una opcion:\n1) Obtener potencia positiva de un numero.\n2) Salir.\n");
    scanf("%d",&indice);
    fflush(stdin);

    if(indice == 1){
        printf("Introduce tus valores (potencias mayores o iguales a cero):\nNumero: ");
        scanf("%f",&numero);
        fflush(stdin);
        printf("Potencia: ");
        scanf("%d",&potencia);
        fflush(stdin);
        if(potencia<0)
            printf("\n\nEl programa solo calcula potencias mayores o iguales a cero.\n\n");

        else{
            float resultado = Potencia(numero,potencia);
            printf("\n\n(%.2f)^%d = %.2f\n\n\n", numero,potencia,resultado);
        }
    }

}
    return 0;
}

float Potencia(float numero, int potencia){
    if(potencia == 1 || potencia == 0){
        return (potencia == 1) ? numero:1;
    }
    else
        return numero*Potencia(numero, potencia-1);
}
```

Resultados (compilación)



Caso 1	Caso 2	Caso 3
numero = 3	numero = 2.34	numero = 4
potencia = 5	potencia = 3	potencia = 0
resultado = 243	resultado = 12.81	resultado = 1

CONCLUSIONES

Utilizar recursividad optimiza el código ya que la cantidad de sentencias se reduce, además facilita el uso de ciclos aunque inicialmente puede resultar confuso al leer el código por primera vez.