ESTUDIOS
GENERALES
CIENCIAS

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

FUNDAMENTOS DE PROGRAMACIÓN
LABORATORIO 5
PROPUESTAS DE SOLUCIÓN
SEMESTRE ACADÉMICO 2022-2

Horarios: Todos los horarios

Elaborado por Mag. Silvia Vargas

INDICACIONES:

- Debe utilizar variables descriptivas, comentarios y mensajes descriptivos.
- El orden y la eficiencia de su implementación serán considerados en la calificación.

RESULTADOS ESPERADOS:

- Al finalizar la sesión, el alumno construirá programas usando programación modular usando paso de parámetro por valor y simulando el paso de parámetros por referencia

CONSIDERACIONES:

- La solución presentada para cada problema corresponde a una propuesta de solución por parte del autor.
- En programación pueden existir muchas soluciones para un mismo problema pero debe cumplir con todo lo solicitado, incluyendo las restricciones brindadas.

Desarrolle los siguientes problemas en lenguaje C:

1. Números trimórficos y números insólitos - Horarios: 0390

Existen diversos tipos de números, los números perfectos, amigos, abundantes, primos, etc.

Un número es trimórfico si al elevarlo al cubo, el resultado tiene los últimos dígitos iguales al número. Por ejemplo: 25 es un número trimórfico porque 25^3=15625 y los últimos 2 dígitos de este número son 25. Los primeros números trimórficos son: 1, 4, 5, 6, 9, 24, 25, 49, 51, 75, 76, 99, 125, 249, 251, 375, 376, 499, etc.

Un número es insólito si es divisible entre la suma del cuadrado de sus dígitos y entre la multiplicación del cuadrado de sus dígitos. Por ejemplo: 11112 es un número insólito porque es divisible entre 8=1^2+1^2+1^2+1^2+2^2 y entre 4= 1^2*1^2*1^2*1^2*2^2 . Los primeros números insólitos son: 1, 111, 11112, 1122112, etc.

Se le pide implementar un programa en lenguaje C que muestre un menú de opciones al usuario para que pueda elegir si va mostrar una lista de números trimórficos (T) o de números insólitos (I), para ello debe solicitar un valor que representa la cantidad de dígitos de los números que analizará. También debe mostrar la cantidad de números que se encontraron dentro del rango.

En el caso que se solicite la lista de números trimórficos (T) la cantidad de dígitos debe ser positiva y menor a 4, si se solicita la lista de números insólitos la cantidad de dígitos debe ser positiva y menor a 8.

Para desarrollar el problema debe utilizar el paradigma de programación modular, implementando como mínimo 5 módulos incluído el principal. De los módulos que implemente por lo menos uno debe simular el uso parámetros por referencia modificando más de un parámetro (este módulo no se puede utilizar para leer los datos) y por lo

menos tres módulos deben devolver un valor (sin modificar parámetros). Para calcular las operaciones con los dígitos para el número insólito debe usar una sola iterativa.

Debe usar los mensajes que se muestran en los casos de prueba para el desarrollo del programa.

Casos de prueba

```
Ingrese la opción a evaluar
T: números trimórficos - I: números insólitos
F
Opción incorrecta
```

```
Ingrese la opción a evaluar
T: números trimórficos - I: números insólitos
t
Opción incorrecta
```

```
Ingrese la opción a evaluar
T: números trimórficos - I: números insólitos
T
Ingrese la cantidad de dígitos: 6
La cantidad de dígitos no corresponde a la opción
```

```
Ingrese la opción a evaluar
T: números trimórficos - I: números insólitos
T
Ingrese la cantidad de dígitos: 2
Lista de números trimórficos
24
25
49
51
75
76
99
En el rango [10,99] existen 7 números trimórficos
```

```
Ingrese la opción a evaluar
T: números trimórficos - I: números insólitos
I
Ingrese la cantidad de dígitos: 12
La cantidad de dígitos no corresponde a la opción
```

```
Ingrese la opción a evaluar
T: números trimórficos - I: números insólitos
I
Ingrese la cantidad de dígitos: 6
En el rango [100000,999999] no existen números insólitos
```

```
Ingrese la opción a evaluar
T: números trimórficos - I: números insólitos
I
Ingrese la cantidad de dígitos: 7
Lista de números insólitos
1122112
En el rango [1000000,9999999] existe 1 número insólito
```

Programa 1: Propuesta de solución - Sumatorias

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

int validar(int, char );
int contarNumTrimorficos(int ,int, int );
int contarNumInsolitos(int ,int , int);
void calcularOpeDigitos(int ,int, int *,int *);

int main(){
        int cantDigitos,cantNum,inicio,fin;
        char opcion;
        printf("Ingrese la opción a evaluar\n");
        printf("T: números trimórficos – I: números insólitos\n");
        scanf(" %c",&opcion);
        //evaluación de opciones
        if (opcion=='T'|| opcion=='I'){
                printf("Ingrese la cantidad de dígitos: ");
                scanf(" %d",&cantDigitos);
                //validar para opción la cantidad de dígitos
                if (validar(cantDigitos,opcion)){
                        inicio=(int)pow(10,cantDigitos-1);
                        fin=(int)pow(10,cantDigitos)-1;
                        if (opcion=='T'){
                                cantNum=contarNumTrimorficos(inicio,fin,cantDigitos);
                                if (cantNum>1)
                                        printf("En el rango [ %d, %d] existen  %d números trimórficos\n",inicio,fin,
                                                cantNum);
                                else
                                        if (cantNum==1)
                                                printf("En el rango [ %d, %d] existe  %d número trimórfico\n",inicio,fin,cantNum);
                                        else
                                                printf("En el rango [ %d, %d] no existen números trimórficos\n",inicio,fin);
                        }
                        else{
                                cantNum=contarNumInsolitos(inicio,fin,cantDigitos);
                                if (cantNum>1)
                                        printf("En el rango [ %d, %d] existen  %d números insólitos\n",inicio,fin,cantNum);
                                else
                                        if (cantNum==1)
                                                printf("En el rango [ %d, %d] existe  %d número insólito\n",inicio,fin,cantNum);
                                        else
                                                printf("En el rango [ %d, %d] no existen números insólitos\n",inicio,fin);
                        }
                }
                else
                        printf("La cantidad de dígitos no corresponde a la opción\n");
        }
        else
                printf("Opción incorrecta\n");
        return 0;
}

int validar(int cantDigitos,char opcion){
        if (opcion=='T')
                return cantDigitos>0 && cantDigitos<4;
```

```
55          else
56                  return cantDigitos>0 && cantDigitos<8;
57  }
58
59  int contarNumTrimorficos(int inicio,int fin,int cantDig){
60          int num,digitos,cont=0;
61          while(inicio<=fin){
62                  num=(int)pow(inicio,3);
63                  //extracción de los últimos dígitos del número
64                  digitos=num %(int)pow(10,cantDig);
65                  if (inicio==digitos){
66                          if (cont==0)
67                                  //muestra solo una vez el título de la lista y solo si por lo menos
68                                  //hay un número
69                                  printf("Lista de números trimórficos\n");
70                          printf(" %d\n",inicio);
71                          cont++;
72                  }
73                  inicio++;
74          }
75          return cont;
76  }
77
78  int contarNumInsolitos(int inicio,int fin,int cantDig){
79          int num,digitos,cont=0;
80          int sumaCuad,multCuad;
81          while(inicio<=fin){
82                  calcularOpeDigitos(inicio,cantDig,&sumaCuad,&multCuad);
83                  if (multCuad>0)
84                          if (inicio %sumaCuad==0 && inicio %multCuad==0){
85                                  if (cont==0)
86                                          printf("Lista de números insólitos\n");
87                          printf(" %d\n",inicio);
88                          cont++;
89                  }
90                  inicio++;
91          }
92          return cont;
93  }
94
95  void calcularOpeDigitos(int num,int cantDig,int *sumaCuad,int *multCuad){
96          int i=1,digito;
97          *sumaCuad=0;
98          *multCuad=1;
99          /*ya que no es posible realizar iterativas anidadas, se usar selectivas
100         anidadas, debido a ello si se permite repetir instrucciones*/
101         digito=num %10;
102         num/=10;
103         *sumaCuad+=(int)pow(digito,2);
104         *multCuad*=(int)pow(digito,2);
105         if (num>0){
106                 digito=num %10;
107                 num/=10;
108                 *sumaCuad+=(int)pow(digito,2);
109                 *multCuad*=(int)pow(digito,2);
110                 if (num>0){
111                         digito=num %10;
112                         num/=10;
113                         *sumaCuad+=(int)pow(digito,2);
114                         *multCuad*=(int)pow(digito,2);
115                         if (num>0){
116                                 digito=num %10;
117                                 num/=10;
118                                 *sumaCuad+=(int)pow(digito,2);
119                                 *multCuad*=(int)pow(digito,2);
120                                 if (num>0){
121                                         digito=num %10;
```

```
                                        num/=10;
                                        *sumaCuad+=(int)pow(digito,2);
                                        *multCuad*=(int)pow(digito,2);
                                        if (num>0){
                                                digito=num %10;
                                                num/=10;
                                                *sumaCuad+=(int)pow(digito,2);
                                                *multCuad*=(int)pow(digito,2);
                                                if (num>0){
                                                        digito=num %10;
                                                        num/=10;
                                                        *sumaCuad+=(int)pow(digito,2);
                                                        *multCuad*=(int)pow(digito,2);
                                                }
                                        }
                                }
                        }
                }
        }
        /*en caso se permita utilizar iterativas anidadas*/
        /*while(i<=cantDig){
                digito=num %10;
                num/=10;
                *sumaCuad+=(int)pow(digito,2);
                *multCuad*=(int)pow(digito,2);
                i++;
        }*/
}
```

2. Números de Zuckermann y números divisibles entre operaciones con divisores - Horarios: B301, 0386, 0387, 0391

Existen diversos tipos de números, los números perfectos, amigos, abundantes, primos, etc.

Un número de Zuckermann es aquel que es divisible por el producto de sus dígitos. Por ejemplo: 212 es un número de Zuckermann porque 212 es divisible entre 4=2*1*2. Los primeros números de Zuckermann son: 1, 2, 3, 4, 5, 6, 7, 8, 9, 11, 12, 15, 24, 36, 111, 112, 115, 128, 132, 135, etc.

También existen números que son divisibles entre la suma de sus divisores, la cantidad de sus divisores y la multiplicación de sus divisores, sin considerar al mismo número como divisor. Los primeros números de 2 dígitos que son divisibles entre la suma de sus divisores son: 11, 13, 17, 19, 23, 28, etc. Los primeros números de 2 dígitos que son divisibles entre la cantidad de sus divisores son: 11, 13, 15, 16, 17, 19, 20, etc. Los primeros números de 2 dígitos que son divisibles entre la multiplicación de sus divisores son: 10, 14, 15, 17, 19, 21, etc. Los primeros números de 2 dígitos que son divisibles entre la suma, cantidad y multiplicación de sus divisores son: 11, 13, 17, 19, 23, 29, etc.

Se le pide implementar un programa en lenguaje C que muestre un menú de opciones al usuario para que pueda elegir si va evaluar un número de Zuckermann (Z) o si un número es divisible entre las operaciones con sus divisores (D), para ello debe solicitar un número positivo, el cual tener como máximo 6 dígitos.Al evaluar el número de Zuckermann debe mostrar la multiplicación de los dígitos del número y si el número es o no divisible entre ese valor. Al evaluar el número con las operaciones de sus divisores, debe mostrar el resultado de las operaciones y por cuales de ellos es divisible el número. Además debe controlar que al realizar la multiplicación de los divisores no se exceda del límite de los enteros que almacena el lenguaje C, para lo cual debe usar la constante INT_MAX definida en la biblioteca de funciones limits.h .

Para desarrollar el problema debe utilizar el paradigma de programación modular, implementando como mínimo 5 módulos incluído el principal. De los módulos que implemente por lo menos uno debe simular el uso parámetros por referencia modificando más de un parámetro (este módulo no se puede utilizar para leer los datos) y por lo menos dos módulos deben devolver un valor (sin modificar parámetros).

Para calcular la cantidad de dígitos de un número no debe utilizar una iteración, use la parte entera del logaritmo en base 10 del número y adicione uno. Para multiplicar los dígitos del número debe utilizar una iteración. Para calcular las operaciones con los dígitos debe usar una sola iteración.

Debe usar los mensajes que se muestran en los casos de prueba para el desarrollo del programa.

Casos de prueba

```
Ingrese la opción a ejecutar: Z: número Zuckermann - D: Operaciones con dígitos
d
Opción incorrecta
```

```
Ingrese la opción a ejecutar: Z: número Zuckermann - D: Operaciones con dígitos
Z
Ingresar el número a evaluar: 12500000
Número incorrecto
```

```
Ingrese la opción a ejecutar: Z: número Zuckermann - D: Operaciones con dígitos
Z
Ingresar el número a evaluar: 123
La multiplicación de los dígitos del número es 6
El número 123 NO es divisible entre 6, por lo cual NO es un número Zuckermann
```

```
Ingrese la opción a ejecutar: Z: número Zuckermann - D: Operaciones con dígitos
Z
Ingresar el número a evaluar: 112116
La multiplicación de los dígitos del número es 12
El número 112116 es divisible entre 12, por lo cual es un número Zuckermann
```

```
Ingrese la opción a ejecutar: Z: número Zuckermann - D: Operaciones con dígitos
D
Ingresar el número a evaluar: 789899999
Número incorrecto
```

```
Ingrese la opción a ejecutar: Z: número Zuckermann - D: Operaciones con dígitos
D
Ingresar el número a evaluar: 12345
La suma de los divisores del número es 7431
La cantidad de los divisores del número es 7
La multiplicación de los divisores del número esta fuera del límite de los enteros
El número 12345 NO es divisible entre la suma, ni cantidad ni multiplicación de sus
divisores
```

```
Ingrese la opción a ejecutar: Z: número Zuckermann - D: Operaciones con dígitos
D
Ingresar el número a evaluar: 123
La suma de los divisores del número es 45
La cantidad de los divisores del número es 3
La multiplicación de los divisores del número es 123
El número 123 es divisible entre la cantidad y multiplicación de sus divisores
```

```
Ingrese la opción a ejecutar: Z: número Zuckermann - D: Operaciones con dígitos
D
Ingresar el número a evaluar: 73
La suma de los divisores del número es 1
La cantidad de los divisores del número es 1
La multiplicación de los divisores del número es 1
El número 73 es divisible entre la suma, cantidad y multiplicación de sus divisores
```

```
Ingrese la opción a ejecutar: Z: número Zuckermann - D: Operaciones con dígitos
D
Ingresar el número a evaluar: 75
La suma de los divisores del número es 49
La cantidad de los divisores del número es 5
La multiplicación de los divisores del número es 5625
El número 75 es divisible solo entre la cantidad de sus divisores
```

```
Ingrese la opción a ejecutar: Z: número Zuckermann - D: Operaciones con dígitoss
D
Ingresar el número a evaluar: 1234
La suma de los divisores del número es 620
La cantidad de los divisores del número es 3
La multiplicación de los divisores del número es 1234
El número 1234 es divisible solo entre la multiplicación de sus divisores
```

Programa 2: Propuesta de solución - Sumatorias

```c
#include <stdio.h>
#include <math.h>
#include <limits.h>

void evaluarZuckermann(int );
int calcularMultDig(int );
int calcularCantDigitos(int );
void evaluarDivisores(int );
void calcularOpeDivisores(int ,int *,int *,int *);

int main(){
        int numero;
        char opcion;
        printf("Ingrese la opción a ejecutar: ");
        printf("Z: número Zuckermann – D: Operaciones con dígitos\n");
        scanf(" %c",&opcion);
        //evaluación de opciones
        if (opcion=='Z' || opcion=='D'){
                printf("Ingresar el número a evaluar: ");
                scanf(" %d",&numero);
                if (numero>0 && numero<=(int)pow(10,7)-1)
                        if (opcion=='Z')
                                evaluarZuckermann(numero);
                        else
                                evaluarDivisores(numero);
                else
                        printf("Número incorrecto\n");
        }
        else
                printf("Opción incorrecta\n");
        return 0;
}

/*números de Zuckermann*/
void evaluarZuckermann(int num){
        int multDig;
        multDig=calcularMultDig(num);
        printf("La multiplicación de los dígitos del número es  %d\n",multDig);
        if (num %multDig==0)
                printf("El número  %d es divisible entre  %d, por lo cual es un número Zuckermann\n",num,multDig);
        else
                printf("El número  %d NO es divisible entre  %d, por lo cual NO es un número Zuckermann\n",num,multDig);
}

int calcularMultDig(int num){
        int cantDig,i=1,digito,multDig=1;
```

```
47          cantDig=calcularCantDigitos(num);
48          while(i<=cantDig){
49                  digito=num %10;
50                  num/=10;
51                  multDig*=digito;
52                  i++;
53          }
54          return multDig;
55  }
56
57
58  int calcularCantDigitos(int num){
59          return (int)log10(num)+1;
60  }
61
62  void evaluarDivisores(int num){
63          int sumaDiv,contDiv,multDiv,propSuma,propCont,propMult;
64          calcularOpeDivisores(num,&sumaDiv,&contDiv,&multDiv);
65          printf("La suma de los divisores del número es  %d\n",sumaDiv);
66          printf("La cantidad de los divisores del número es  %d\n",contDiv);
67          if (multDiv==-1){
68                  printf("La multiplicación de los divisores del número esta fuera del límite de los enteros\n");
69                  propMult=0;
70          }
71          else{
72                  printf("La multiplicación de los divisores del número es  %d\n",multDiv);
73                  propMult=num %multDiv==0;
74          }
75          propSuma=num %sumaDiv==0;
76          propCont=num %contDiv==0;
77          if (propSuma && propCont && propMult)
78                  printf("El número  %d es divisible entre la suma, cantidad y multiplicación de sus divisores\n",num);
79          else
80                  if (propSuma && propCont)
81                          printf("El número  %d es divisible entre la suma y cantidad de sus divisores\n",num);
82                  else
83                          if (propCont && propMult)
84                                  printf("El número  %d es divisible entre la cantidad y multiplicación de sus divisores\n",num);
85                          else
86                                  if (propSuma && propMult)
87                                          printf("El número  %d es divisible entre la suma y multiplicación de sus divisores\n",num);
88                                  else
89                                          if (propSuma)
90                                                  printf("El número  %d es divisible solo entre la suma de sus divisores\n",num);
91                                          else
92                                                  if (propMult)
93                                                          printf("El número  %d es divisible solo entre la multiplicación de sus
                                                                divisores\n",num);
94                                                  else
95                                                          if (propCont)
96                                                                  printf("El número  %d es divisible solo entre la cantidad de sus
                                                                        divisores\n",num);
97                                                          else
98                                                                  printf("El número  %d NO es divisible entre la suma, ni cantidad
                                                                        ni multiplicación de sus divisores\n",num);
99  }
100
101 void calcularOpeDivisores(int num,int *sumaDiv,int *contDiv,int *multDiv){
102         int i=1;
103         *sumaDiv=0;
104         *contDiv=0;
105         *multDiv=1;
106         while(i<num){
107                 if (num %i==0){
108                         (*sumaDiv)+=i;
109                         //control de que la multiplicación no sobrepase el límite de los enteros
110                         if (*multDiv!=-1 && INT_MAX/i>=(*multDiv))
```

```
111                                 (*multDiv)*=i;
112                         else
113                                 (*multDiv)=-1;
114                         (*contDiv)++;
115                 }
116                 i++;
117         }
118 }
```

3. Números de Pastel y operaciones con los dígitos de números - Horarios: 0380, 0381, 0384, 0385

Existen diversos tipos de números, los números perfectos, amigos, abundantes, primos, etc.

[1] Un número pastel C_n es el número máximo de regiones en las que se puede dividir un cubo tridimensional por exactamente n planos. El número de pastel se llama así porque se puede imaginar cada partición del cubo generada por un plano, como una rebanada hecha por un cuchillo a través de un pastel en forma de cubo. Para encontrar un número pastel C_n, se utiliza la siguiente fórmula:

$$C_n = \binom{n+1}{3} + n + 1$$

donde:

$$\binom{n}{k} = \frac{n!}{k!(n-k)!}$$

Los primeros números pastel son: 1, 2, 4, 8, 15, 26, 42, 64, 93, 130, 176, 232, 299, 378, 470, etc. Tenga presente que el número pastel 1 corresponde al dividir un cubo tridimensional por exactamente 0 planos.

Otra forma de encontrar los números pastel C_n, es utilizando la siguiente fórmula más sencilla:

$$C_n = \frac{n^3 + 5n + 6}{6}$$

Con los digitos de un número es posible realizar diversas operaciones matemáticas. Una forma es intercalar las operaciones básicas suma, resta, multiplicación y división usando los dígitos del número (empezando desde el dígito de la izquierda o el más significativo) e incluso modificar el signo de cada dígito también en forma intercalada (empezando con el signo positivo y luego se cambia al signo negativo). Por ejemplo: para el número 1234567, la operación a realizar sería ((((((1 + -2)) - 3) * -4) / 5) + -6) - 7 que es igual a -9.8

Se le pide implementar un programa en lenguaje C que muestre un menú de opciones al usuario para que pueda elegir si va calcular un número pastel (1) o si realizará operaciones con los dígitos (2). En caso se elija la opción 1, debe solicitar el número para el cual desea calcular el número pastel Cn y la fórmula a utilizar, usando la fórmula más sencilla (F) o el combinatorio (C). Si selecciona usar el combinatorio, el número debe ser mayor o igual a 0 y como máximo puede ser 10; si seleccionar usar la fórmula más sencilla, el número debe ser mayor o igual a 0 y como máximo puede ser 1000.

En caso se elija la opción 2, debe solicitar el número para el cual desea operar sus dígitos, este número debe tener como mínimo 2 dígitos y como máximo 9 dígitos. Si uno de los dígitos es 0 y será el divisor de una división, no se ejecuta esta operación y se pasa a la siguiente.

Para desarrollar el problema debe utilizar el paradigma de programación modular, implementando como mínimo 5 módulos incluído el principal. De los módulos que implemente por lo menos uno debe simular el uso parámetros por referencia modificando más de un parámetro (este módulo no se puede utilizar para leer los datos) y por lo menos dos módulos deben devolver un valor (sin modificar parámetros).

[1] Tomado de `https://es.wikipedia.org/wiki/Nmero_del_pastel`

Para calcular la cantidad de dígitos de un número no debe utilizar una iteración, use la parte entera del logaritmo en base 10 del número y adicione uno. Para operar los dígitos del número debe usar solo una iteración y selectivas anidadas. Si en el combinatorio debe calcular el factorial de un número negativo, asuma que es 1, igual que para el factorial de 0.

Debe usar los mensajes que se muestran en los casos de prueba para el desarrollo del programa.

Casos de prueba

```
Ingrese la opcion que desea ejecutar (1: número pastel, 2: operaciones con dígitos): 5
Opción incorrecta
```

```
Ingrese la opcion que desea ejecutar (1: número pastel, 2: operaciones con dígitos): 1
Ingrese el número  para calcular el número pastel Cn: -5
Ingrese la forma  para calcular el número pastel Cn (F:fórmula,C:combinatorio): F
Error en los datos ingresados para calcular el número pastel
```

```
Ingrese la opcion que desea ejecutar (1: número pastel, 2: operaciones con dígitos): 1
Ingrese el número para calcular el número pastel Cn: 123
Ingrese la forma  para calcular el número pastel Cn (F:fórmula,C:combinatorio): C
Error en los datos ingresados para calcular el número pastel
```

```
Ingrese la opcion que desea ejecutar (1: número pastel, 2: operaciones con dígitos): 1
Ingrese el número para calcular el número pastel Cn: 40
Ingrese la forma  para calcular el número pastel Cn (F:fórmula,C:combinatorio): F
Cn calculado con la fórmula para el n·mero 40 es 10701
```

```
Ingrese la opcion que desea ejecutar (1: número pastel, 2: operaciones con dígitos): 1
Ingrese el número para calcular el número pastel Cn: 6
Ingrese la forma  para calcular el número pastel Cn (F:fórmula,C:combinatorio): C
Cn calculado con el combinatorio para el n·mero 6 es 42
```

```
Ingrese la opcion que desea ejecutar (1: número pastel, 2: operaciones con dígitos): 2
Ingrese el número para realizar las operaciones: -6
Número incorrecto
```

```
Ingrese la opcion que desea ejecutar (1: número pastel, 2: operaciones con dígitos): 2
Ingrese el número para realizar las operaciones: 567
El resultado es -8.000000
```

```
Ingrese la opcion que desea ejecutar (1: número pastel, 2: operaciones con dígitos): 2
Ingrese el número para realizar las operaciones: 98563472
El resultado es 6.000000
```

Programa 3: Propuesta de solución - Límites de velocidad

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

int validar(int,char);
int calcularNumeroPastel(int posicion,char forma);
int calcularNumeroPastelComb(int num);
int calcularCantDigitos(int num);
double operarNumero(int ,int );

int main(){
```

```
11          int numPastel,opcion,cantDig,num;
12          char forma;
13          double resultado;
14          printf("Ingrese la opción que desea ejecutar (1: número pastel, 2: operaciones con dígitos): ");
15          scanf(" %d",&opcion);
16          //evaluación de opciones
17          if (opcion==1 || opcion==2){
18                  if (opcion==1){
19                          printf("Ingrese el número para calcular el número pastel Cn: ");
20                          scanf(" %d",&num);
21                          printf("Ingrese la forma para calcular el número pastel Cn (F:fórmula,C:combinatorio): ");
22                          scanf("\n %c",&forma);
23                          //validación de datos para el número pastel
24                          if (validar(num,forma)){
25                                  numPastel=calcularNumeroPastel(num,forma);
26                                  if (forma=='F')
27                                          printf("Cn calculado con la fórmula para el número  %d es  %d\n",num,numPastel);
28                                  else
29                                          printf("Cn calculado con el combinatorio para el número  %d es  %d\n",num,numPastel);
30                          }
31                          else
32                                  printf("Error en los datos ingresados para calcular el número pastel\n");
33                  }
34                  else{
35                          printf("Ingrese el número para realizar las operaciones: ");
36                          scanf(" %d",&num);
37                          //validación del número para realizar las operaciones
38                          if (num>10 && num<=(int)pow(10,9)-1){
39                                  cantDig=calcularCantDigitos(num);
40                                  resultado=operarNumero(num,cantDig);
41                                  printf("El resultado es  %lf\n",resultado);
42                          }
43                          else
44                                  printf("Número incorrecto\n");
45                  }
46          }
47          else
48                  printf("Opción incorrecta\n");
49          return 0;
50  }
51
52  int validar(int posicion,char forma){
53          int valComb,valFor;
54          valComb=forma=='C' && posicion>=0 && posicion<=10;
55          valFor=forma=='F' && posicion>=0 && posicion<=1000;
56          return valComb || valFor;
57  }
58
59  int calcularNumeroPastel(int num,char forma){
60          int numPastel;
61          if (forma=='C')
62                  numPastel=calcularNumeroPastelComb(num);
63          else
64                  numPastel=((int)pow(num,3)+5*num+6)/6;
65          return numPastel;
66  }
67
68  int calcularNumeroPastelComb(int num){
69          int i=1,comb,factNum=1,factDenIzq=1;
70          // para el combinatorio solo se usa una iterativa
71          while (i<=num+1){
72                  factNum*=i;
73                  if (i<=num-2)
74                          factDenIzq*=i;
75                  i++;
76          }
77          return (factNum/(factDenIzq*6))+num+1;
```

```
}

int calcularCantDigitos(int num){
        return (int)log10(num)+1;
}

double operarNumero(int num,int cantDigitos){
        int i=cantDigitos,simb=1,signo=1,digito;
        double res=0;
        while(i>=1){
                //extracción de los dígitos desde la izquierda
                digito=num/(int)pow(10,i-1);
                num=num %(int)pow(10,i-1);
                //cambiar el signo del dígito
                digito*=signo;
                signo*=-1;
                //operaciones con los dígitos
                if (simb==1)
                        res=digito;
                else
                        if (simb==2)
                                res+=digito;
                        else
                                if (simb==3)
                                        res-=digito;
                                else
                                        if (simb==4)
                                                res*=digito;
                                        else{
                                                if (digito!=0)
                                                        res/=(double)digito;
                                                simb=1;
                                        }
                simb++;
                i--;
        }
        return res;
}
```

4. Operaciones con los dígitos de números - Horarios: 0382, 0383, 0388, 0389

Con los digitos de un número es posible realizar diversas operaciones matemáticas. Los dígitos se pueden extraer del número desde la parte derecha o desde la izquierda. Por ejemplo, para el número 9342158 (gráficamente se observa en la figura 4):

Número	**9**	**3**	**4**	**2**	**1**	**5**	**8**
Posición de los dígitos iniciando desde la derecha	7	6	5	4	3	2	1
Posición de los dígitos iniciando desde la izquierda	1	2	3	4	5	6	7

- si extraemos los dígitos de uno en uno desde la derecha, el primero será el 8, el segundo el 5, el tercero el 1 y así hasta llegar al último el dígito 9 en la posición 7.
- si extraemos los dígitos de uno en uno desde la izquierda, el primero será el 9, el segundo el 3, el tercero el 4 y así hasta llegar al último el dígito 8 en la posición 7.

También podemos extraer los dígitos cuyas posiciones son múltiplos de un número específico, por ejemplo, para el número 9342158:

- si extraemos los dígitos en posiciones múltiplos de 3 desde la derecha, el primero será el 1 y el segundo el 3.

- si extraemos los dígitos en posiciones múltiplos de 3 desde la izquierda, el primero será el 4 y el segundo el 5.

Con lo descrito anteriormente es posible realizar operaciones con los dígitos en posiciones múltiplos de un número, con la parte izquierda y derecha que quedan del número al extraer el dígito específico.

Para el número 9342158, extrayendo los dígitos en las posiciones múltiplos de 3 desde la derecha, las operaciones a realizar son las siguientes:

Primera Operación:

- Dígito en la primera posición múltiplo de 3 (posición 3) es el 1, lo queda del número al extraer el 1, a la izquierda 9342 y a la derecha 58. Operación (58 + 9342) / 1 = 9400
- Dígito en la segunda posición múltiplo de 3 (posición 6) es el 3, lo queda del número al extraer el 3, a la izquierda 9 y a la derecha 42158. Operación (42158 + 9) / 3 = 14055.666667. Sumamos este resultado al anterior 9400 + 14055.666667 = 23455.666667

Segunda Operación:

- Dígito en la primera posición múltiplo de 3 (posición 3) es el 1, lo queda del número al extraer el 1, a la izquierda 9342 y a la derecha 158. Operación (58 - 9342) * 1 = -9284
- Dígito en la segunda posición múltiplo de 3 (posición 6) es el 3, lo queda del número al extraer el 3, a la izquierda 9 y a la derecha 42158. Operación (42158 - 9) * 3 = 126447. Restamos este resultado al anterior -9284 - 126447 = -135731

Para el número 9342158, extrayendo los dígitos en las posiciones múltiplos de 3 desde la izquierda, las operaciones a realizar son las siguientes:

Primera Operación:

- Dígito en la primera posición múltiplo de 3 (posición 3) es el 4, lo queda del número al extraer el 4, a la izquierda 93 y a la derecha 2158. Operación 2158 * 4 = 8632
- Dígito en la segunda posición múltiplo de 3 (posición 6) es el 5, lo queda del número al extraer el 3, a la izquierda 93421 y a la derecha 8. Operación 8 * 5 = 40. Sumamos este resultado al anterior 8632 + 40 = 8672

Segunda Operación:

- Dígito en la primera posición múltiplo de 3 (posición 3) es el 4, lo queda del número al extraer el 4, a la izquierda 93 y a la derecha 2158. Operación (93 - 2158) / 4 = -516.25
- Dígito en la segunda posicion múltiplo de 3 es el 5, lo queda del número al extraer el 5, a la izquierda 93421 y a la derecha 8. Operación (93421 - 8) / 5 = 18682.6. Restamos este resultado al anterior -516.25 - 18646.6 = -19198.85

Se le pide implementar un programa en lenguaje C que solicite un número, la forma de extraer los dígitos (desde la izquierda: I o i, desde la derecha: D o d) y el múltiplo de dígitos para procesar el número; con lo cual debe calcular y mostrar las operaciones solicitadas de acuerdo a lo explicado anteriormente.

Debe validar que el número a ingresar debe ser positivo, como máximo puede tener 9 dígitos y que ninguno de sus dígitos sea 0. También debe validar que la forma de extracción de dígitos sean solo los caractes D, d, I o i, finalmente debe validar que la cantidad de dígitos del número sea menor o igual al múltiplo de dígitos ingresado.

Para desarrollar el problema debe utilizar el paradigma de programación modular, implementando como mínimo 5 módulos incluído el principal. De los módulos que implemente por lo menos uno debe simular el uso parámetros

por referencia modificando más de un parámetro (este módulo no se puede utilizar para leer los datos) y por lo menos dos módulos deben devolver un valor (sin modificar parámetros).

Para calcular la cantidad de dígitos de un número no debe utilizar una iteración, use la parte entera del logaritmo en base 10 del número y adicione uno. Para extraer y operar los dígitos del número y las partes derecha e izquierda del número, debe usar solo una iteración y selectivas anidadas.

Debe usar los mensajes que se muestran en los casos de prueba para el desarrollo del programa.

Casos de prueba

```
Ingrese el número: -12
Número incorrecto
```

```
Ingrese el número: 1234567881
Número incorrecto
```

```
Ingrese el número: 1323045
Alguno de los dígitos del número es 0, no se pueden realizar las operaciones solicitadas
```

```
Ingrese el número: 6756734
Ingrese la forma para extraer los dígitos del número: s
Error al ingresar la forma de extraer los dígitos del número
```

```
Ingrese el número: 34581234
Ingrese la forma para extraer los dígitos del número: D
Ingrese el múltiplo de dígitos para extraer del número: 12
Error al ingresar el múltiplo de dígitos
```

```
Ingrese el número: 9342158
Ingrese la forma para extraer los dígitos del número: D
Ingrese el múltiplo de dígitos para extraer del número: 3
Las operaciones realizadas con el número 9342158, extrayendo 3 dígitos desde la derecha
son:
Primer resultado = 23455.666667
Segundo resultado = -135731.000000
```

```
Ingrese el número: 9342158
Ingrese la forma para extraer los dígitos del número: I
Ingrese el múltiplo de dígitos para extraer del número: 3
Las operaciones realizadas con el número 9342158, extrayendo 3 dígitos desde la izquierda
son:
Primer resultado = 8672.000000
Segundo resultado = -19198.850000
```

```
Ingrese el número: 674326
Ingrese la forma para extraer los dígitos del número: D
Ingrese el múltiplo de dígitos para extraer del número: 2
Las operaciones realizadas con el número 674326, extrayendo 2 dígitos desde la derecha
son:
Primer resultado = 15860.416667
Segundo resultado = -460466.000000
```

```
Ingrese el número: 545
Ingrese la forma para extraer los dígitos del número: d
Ingrese el múltiplo de dígitos para extraer del número: 2
Las operaciones realizadas con el número 545, extrayendo 2 dígitos desde la derecha son:
Primer resultado = 2.500000
Segundo resultado = 0.000000
```

```
Ingrese el número: 87375
Ingrese la forma para extraer los dígitos del número: I
Ingrese el múltiplo de dígitos para extraer del número: 1
Las operaciones realizadas con el número 87375, extrayendo 1 dígitos desde la izquierda
son:
Primer resultado = 61885.000000
Segundo resultado = -2744.846429
```

Programa 4: Propuesta de solución - Encuestas

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

int calcularCantDigitos(int );
int validarNumero(int,int);
void operarDigitos(int ,char ,int ,int ,double *,double *);
void escribirResultados(int ,char ,int ,double ,double );

int main(){
	int num,cantDigProc,cantDigNum;
	char forma;
	double primResultado,segResultado;
	printf("Ingrese el número: ");
	scanf(" %d",&num);
	if (num>0 && num>0 && num<=(int)pow(10,9)-1){
		cantDigNum=calcularCantDigitos(num);
		//validación que no existan dígitos 0
		if (validarNumero(num,cantDigNum)){
			printf("Ingrese la forma para extraer los dígitos del número: ");
			scanf("\n%c",&forma);
			if (forma=='D'|| forma=='d' || forma=='I' || forma=='i'){
				printf("Ingrese el múltiplo de dígitos para extraer del número: ");
				scanf(" %d",&cantDigProc);
				if (cantDigNum>=cantDigProc){
					operarDigitos(num,forma,cantDigNum,cantDigProc,&primResultado,&segResultado);
					escribirResultados(num,forma,cantDigProc,primResultado,segResultado);
				}
				else
					printf("Error al ingresar el múltiplo de dígitos\n");
			}
			else
				printf("Error al ingresar la forma de extraer los dígitos del número\n");
		}
		else
			printf("Alguno de los dígitos del número es 0, no se pueden realizar las operaciones solicitadas\n");
	}
	else
		printf("Número incorrecto\n");
	return 0;
}

int calcularCantDigitos(int num){
	return (int)log10(num)+1;
}

int validarNumero(int num,int cantDigNum){
	int i=1,digito,errorNum=0;
	while(i<=cantDigNum){
		digito=num %10;
		num/=10;
		if (digito==0)
			errorNum=1;
		i++;
	}
	return !errorNum;
```

```
}

void operarDigitos(int num,char forma,int cantDigNum,int cantDigProc,double *primRes,double *segRes){
        int i=1,numDer=0,numIzq=0,digito;
        *primRes=0;
        *segRes=0;
        while(i<=cantDigNum){
                if (forma=='D' || forma=='d'){
                        //extracción de dígito y de parte derecha e izquierda si parte de la derecha
                        digito=num %10;
                        numIzq=num/10;
                        num/=10;
                        if (i %cantDigProc==0){
                                *primRes+=(numDer+numIzq)/(double)digito;
                                if (i==cantDigProc)
                                        *segRes=(numDer-numIzq)*digito;
                                else
                                        *segRes-=(numDer-numIzq)*digito;
                        }
                        numDer+=digito*(int)(pow(10,i-1));
                }
                //extracción de dígito y de parte derecha e izquierda si parte de la izquierda
                else{
                        digito=num/(int)pow(10,cantDigNum-i);
                        num %=(int)pow(10,cantDigNum-i);
                        numDer=num;
                        if (i %cantDigProc==0){
                                *primRes+=numDer*digito;
                                if (i==cantDigProc)
                                        *segRes=(numIzq-numDer)/(double)digito;
                                else
                                        *segRes-=(numIzq-numDer)/(double)digito;
                        }
                        numIzq=numIzq*10+digito;
                }
                i++;
        }
}

void escribirResultados(int num,char forma,int cantDigProc,double primResultado,
        double segResultado){
        printf("Las operaciones realizadas con el número  %d, extrayendo  %d dígitos desde la ",num,cantDigProc);
        if (forma=='D'|| forma=='d')
                printf("derecha son:\n");
        else
                printf("izquierda son:\n");
        printf("Primer resultado =  %lf\n",primResultado);
        printf("Segundo resultado =  %lf\n",segResultado);
}
```

No debe usar estructuras algorítmicas iterativas anidadas, iterativas con salida controlada o por centinela.