ESTUDIOS
GENERALES
CIENCIAS

PONTIFICIA
UNIVERSIDAD
CATÓLICA
DEL PERÚ

FUNDAMENTOS DE PROGRAMACIÓN
LABORATORIO 8
PROPUESTAS DE SOLUCIÓN
SEMESTRE ACADÉMICO 2021-2

Horarios: Todos los horarios

Elaborado por Mag. Silvia Vargas y Mag. Sergio Ponce

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 diseño estructurado.

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. Permutaciones

La permutación de un conjunto de números es la variación del orden de estos. Por ejemplo:

- Para los números: 7 y 9. Sus permutaciones son: 79 y 97.
- Para los números: 1, 3 y 5. Sus permutaciones son: 135 153 315 351 513 y 531.

Si en el caso de tener números tendríamos letras, por ejemplo las 24 permutaciones de A,B,C y D serían las que se muestra en el Cuadro 1.

ABCD	BACD	CABD	DABC
ABDC	BADC	CADB	DACB
ACBD	BCAD	CBAD	DBAC
ACDB	BCDA	CBDA	DBCA
ADBC	BDAC	CDAB	DCAB
ADCB	BDCA	CDBA	DCBA

Cuadro 1: Permutaciones A, B, C y D

Se pide que desarrolle un programa en Lenguaje C que reciba cuatro letras mayúsculas diferentes y muestre sus permutaciones numeradas, la cantidad de las mismas, calcule y muestre el mayor número formado por las permutaciones. Este número se formará multiplicando el valor ASCII de cada letra en su posición númerica.

El último caracter de la derecha sería la unidad, el siguiente caracter la decena, el siguiente la centena y el último (el primero empezando de la izquierda) el millar.

Por ejemplo para las permutaciones del cuadro 1 el mayor número formado sería: 75425, resultado de multiplicar 68 x 1000 + 67 x 100 + 66 x 10 + 65.

El valor ASCII del caracter A es 65, de B es 66, de C es 67 y de D 68.

Tome en cuenta que: la cantidad de permutaciones que se muestran se debe calcular, no es un valor fijo y que no puede usar dentro del programa los valores ASCII de las letras.

En el desarrollo del programa debe utilizar como mínimo 5 módulos incluído el principal. De estos módulos; por lo menos uno debe simular el uso de parámetros por referencia y no debe usarse para leer datos; por lo menos tres de ellos deben devolver un valor y por lo menos en uno de estos el valor a devolver debe ser un caracter. Además en uno de los módulos, diferente al principal, debe invocar a dos módulos diferentes y desarrollar una iterativa anidada.

Use los siguientes casos de prueba para verificar si su solución está correcta.

```
Ingrese 4 caracteres:rOsA
Las letras ingresadas no son mayúsculas o no son diferentes entre sí
```

```
Ingrese 4 caracteres:RAMA
Las letras ingresadas no son mayúsculas o no son diferentes entre sí
```

```
Ingrese 4 caracteres:RAMO
1) R A M O
2) R A O M
3) R M A O
4) R M O A
5) R O A M
6) R O M A
7) A R M O
8) A R O M
9) A M R O
10) A M O R
11) A O R M
12) A O M R
13) M R A O
14) M R O A
15) M A R O
16) M A O R
17) M O R A
18) M O A R
19) O R A M
20) O R M A
21) O A R M
22) O A M R
23) O M R A
24) O M A R
Cantidad de permutaciones: 24
El mayor número formado es: 90735
Es el resultado de la permutación 6 de las letras ROMA 82x1000+79x100+77x10+65 =
90735
```

```
Ingrese 4 caracteres:ROSA
1) R O S A
```

```
2) R O A S
3) R S O A
4) R S A O
5) R A O S
6) R A S O
7) O R S A
8) O R A S
9) O S R A
10) O S A R
11) O A R S
12) O A S R
13) S R O A
14) S R A O
15) S O R A
16) S O A R
17) S A R O
18) S A O R
19) A R O S
20) A R S O
21) A O R S
22) A O S R
23) A S R O
24) A S O R
Cantidad de permutaciones: 24
El mayor número formado es: 92055
Es el resultado de la permutación 13 de las letras SROA 83x1000+82x100+79x10+65 =
92055
```

```
Ingrese 4 caracteres:DILE
1) D I L E
2) D I E L
3) D L I E
4) D L E I
5) D E I L
6) D E L I
7) I D L E
8) I D E L
9) I L D E
10) I L E D
11) I E D L
12) I E L D
13) L D I E
14) L D E I
15) L I D E
16) L I E D
17) L E D I
18) L E I D
19) E D I L
20) E D L I
21) E I D L
22) E I L D
23) E L D I
24) E L I D
Cantidad de permutaciones: 24
El mayor número formado es: 84058
Es el resultado de la permutación 16 de las letras LIED 76x1000+73x100+69x10+68 =
84058
```

Programa 1: Propuesta de solución - Permutaciones

```c
#include <stdio.h>

int validarLetras(char car1,char car2,char car3,char car4);
int validarMayusc(char car);
void mostrarEncontrarPermutaciones(char car1,char car2,char car3,char car4,int *cantTotal,
        char *letraImp1Mayor,char *letraImp2Mayor,char *letraImp3Mayor,char *letraImp4Mayor,int *posicion,int *mayor);
char buscarLetra(int ,char ,char , char ,char);
int calcularMayorNumero(char letraImp1Mayor,char letraImp2Mayor,
        char letraImp3Mayor,char letraImp4Mayor);

int main(){
        int cantTotal=0,mayor=0,calculo,posicion;
        char car1,car2,car3,car4;
        char letraImp1Mayor,letraImp2Mayor,letraImp3Mayor,letraImp4Mayor;
        printf("Ingrese 4 caracteres:");
        scanf(" %c %c %c %c",&car1,&car2,&car3,&car4);
        if (validarLetras(car1,car2,car3,car4)){
                mostrarEncontrarPermutaciones(car1,car2,car3,car4,&cantTotal,&letraImp1Mayor,
                &letraImp2Mayor,&letraImp3Mayor,&letraImp4Mayor,&posicion,&mayor);
                calcularMayorNumero(letraImp1Mayor,letraImp2Mayor,letraImp3Mayor,letraImp4Mayor);
                printf("Cantidad de permutaciones:  %d\n",cantTotal);
                printf("El mayor número formado es:  %d\n",mayor);
                printf("Es el resultado de la permutación  %d de las letras  %c %c %c %c ",posicion,letraImp1Mayor,letraImp2Mayor,
                letraImp3Mayor,letraImp4Mayor);
                printf(" %dx1000+ %dx100+ %dx10+ %d =  %d\n",letraImp1Mayor,letraImp2Mayor,
                letraImp3Mayor,letraImp4Mayor,mayor);
        }
        else
                printf("Las letras ingresadas no son mayúsculas o no son diferentes entre sí\n");
        return 0;
}

int validarLetras(char car1,char car2,char car3,char car4){
        int mayusc,dif;
        mayusc=validarMayusc(car1) && validarMayusc(car2) && validarMayusc(car3) && validarMayusc(car4);
        dif=car1!=car2 && car1!=car3 && car1!=car4 && car2!=car3 && car2!=car4 && car3!=car4;
        return mayusc && dif;
}

int validarMayusc(char car){
        return car>='A' && car<='Z';
}

void mostrarEncontrarPermutaciones(char car1,char car2,char car3,char car4,int *cantTotal,
        char *letraImp1Mayor,char *letraImp2Mayor,char *letraImp3Mayor,char *letraImp4Mayor,int *posicion,int *mayor){
        int i,j,k,l,cant=0,calculo;
        char letraImp1,letraImp2,letraImp3,letraImp4;
        for(i=1;i<=4;i++){
                for (j=1;j<=4;j++){
                        for (k=1;k<=4;k++){
                                for (l=1;l<=4;l++) {
                                        if (i!=j && i!=k && i!=l && j!=k && j!=l && k!=l){
                                                letraImp1=buscarLetra(i,car1,car2,car3,car4);
                                                letraImp2=buscarLetra(j,car1,car2,car3,car4);
                                                letraImp3=buscarLetra(k,car1,car2,car3,car4);
                                                letraImp4=buscarLetra(l,car1,car2,car3,car4);
                                                cant++;
                                                printf("%d)  %c  %c  %c  %c\n",cant,letraImp1,letraImp2,letraImp3,letraImp4);
                                                calculo=calcularMayorNumero(letraImp1,letraImp2,letraImp3,letraImp4);
                                                if (calculo>*mayor){
                                                        *mayor=calculo;
                                                        *letraImp1Mayor=letraImp1;
                                                        *letraImp2Mayor=letraImp2;
                                                        *letraImp3Mayor=letraImp3;
                                                        *letraImp4Mayor=letraImp4;
                                                        *posicion=cant;
                                                }
```

```
                                        }
                                }
                        }
                }
        }
        *cantTotal=cant;
}

char buscarLetra(int num,char car1,char car2, char car3,char car4){
        if (num==1)
                return car1;
        else
                if (num==2)
                        return car2;
                else
                        if (num==3)
                                return car3;
                        else
                                return car4;
}

int calcularMayorNumero(char letraImp1Mayor,char letraImp2Mayor,
        char letraImp3Mayor,char letraImp4Mayor){
        return letraImp1Mayor*1000+letraImp2Mayor*100+letraImp3Mayor*10+letraImp4Mayor;
}
```

2. Número primo de Solimas

Un número primo de Solimas es aquel número primo que cumple con $2^a \pm 2^b \pm 1$, donde a>b>1 .

Se pide que desarrolle un programa en Lenguaje C que reciba sucesivos rangos de números, en cada rango calcule y muestre los números primos de Solimas formados por los exponentes que pertenecen al rango. Debe validar que los números ingresados formen un rango válido, el mínimo valor dentro del rango debe ser 2 y el máximo 20. Para terminar el ingreso de los rangos debe colocar en los dos valores 0.

Para cada par de exponentes a,b debe mostrar los números primos de Solimas encontrados, el cálculo realizado correspondiente y la cantidad de números primos de Solimas.

Al finalizar, debe mostrar la cantidad de operaciones que en el rango han encontrado números primos de Solimas.

Para el desarrollo del programa debe desarrollar como mínimo 6 módulos adicionales al main. De estos módulos, por lo menos uno debe simular el uso de parámetros por referencia y no debe usarse para leer datos. Además, debe desarrollar como mínimo tres módulos que devuelvan un valor. Solo en un módulo puede realizar la operación que determina un número primo de Solimas, este debe recibir como parámetros los exponentes y los signos (como caracteres). También, solo en un módulo puede imprimir una operación, este debe recibir como parámetros los exponentes, los signos (como caracteres) y el resultado de la operación.

Debe mostrar todos los mensajes que se muestran en los casos de prueba.

Use los siguientes casos de prueba para verificar si su solución está correcta.

```
--------------------
Ingrese el rango para los exponentes, 2<=rango<=20, ingrese 0 en ambos para
terminar: 2 5
Evaluación de los exponentes 3 y 2
13 = 2^3 + 2^2 + 1
11 = 2^3 + 2^2 - 1
5 = 2^3 - 2^2 + 1
3 = 2^3 - 2^2 - 1
Las cuatro combinaciones de operaciones con los exponentes 3 y 2 dan como
resultado números primos de Solimas
Evaluación de los exponentes 4 y 2
19 = 2^4 + 2^2 - 1
```

```
13 = 2^4 - 2^2 + 1
11 = 2^4 - 2^2 - 1
Solo 3 combinaciones de operaciones con los exponentes 4 y 2 dan como
resultado números primos de Solimas
Evaluación de los exponentes 4 y 3
23 = 2^4 + 2^3 - 1
7 = 2^4 - 2^3 - 1
Solo 2 combinaciones de operaciones con los exponentes 4 y 3 dan como
resultado números primos de Solimas
Evaluación de los exponentes 5 y 2
37 = 2^5 + 2^2 + 1
29 = 2^5 - 2^2 + 1
Solo 2 combinaciones de operaciones con los exponentes 5 y 2 dan como
resultado números primos de Solimas
Evaluación de los exponentes 5 y 3
41 = 2^5 + 2^3 + 1
23 = 2^5 - 2^3 - 1
Solo 2 combinaciones de operaciones con los exponentes 5 y 3 dan como
resultado números primos de Solimas
Evaluación de los exponentes 5 y 4
47 = 2^5 + 2^4 - 1
17 = 2^5 - 2^4 + 1
Solo 2 combinaciones de operaciones con los exponentes 5 y 4 dan como
resultado números primos de Solimas
--------------------
Se encontraron 15 operaciones para hallar números primos de Solimas con exponentes
en el rango [2,5]
--------------------
Ingrese el rango para los exponentes, 2<=rango<=20, ingrese 0 en ambos para
terminar: 6 7
Evaluación de los exponentes 7 y 6
193 = 2^7 + 2^6 + 1
191 = 2^7 + 2^6 - 1
Solo 2 combinaciones de operaciones con los exponentes 7 y 6 dan como
resultado números primos de Solimas
--------------------
Se encontraron 2 operaciones para hallar números primos de Solimas con exponentes
en el rango [6,7]
--------------------
Ingrese el rango para los exponentes, 2<=rango<=20, ingrese 0 en ambos para
terminar: 0 0
Fin de la evaluación
```

```
--------------------
Ingrese el rango para los exponentes, 2<=rango<=20, ingrese 0 en ambos para
terminar: 12 15
Evaluación de los exponentes 13 y 12
12289 = 2^13 + 2^12 + 1
Solo una combinación de operaciones con los exponentes 13 y 12 da como
resultado un número primo de Solimas
Evaluación de los exponentes 14 y 12
20479 = 2^14 + 2^12 - 1
12289 = 2^14 - 2^12 + 1
Solo 2 combinaciones de operaciones con los exponentes 14 y 12 dan como
resultado números primos de Solimas
Evaluación de los exponentes 14 y 13
8191 = 2^14 - 2^13 - 1
Solo una combinación de operaciones con los exponentes 14 y 13 da como
resultado un número primo de Solimas
Evaluación de los exponentes 15 y 12
Ninguna combinación de operaciones con los exponentes 15 y 12 da como
```

```
resultado un número primo de Solimas
Evaluación de los exponentes 15 y 13
40961 = 2^15 + 2^13 + 1
Solo una combinación de operaciones con los exponentes 15 y 13 da como
resultado un número primo de Solimas
Evaluación de los exponentes 15 y 14
Ninguna combinación de operaciones con los exponentes 15 y 14 da como
resultado un número primo de Solimas
--------------------
Se encontraron 5 operaciones para hallar números primos de Solimas con exponentes
en el rango [12,15]
--------------------
Ingrese el rango para los exponentes, 2<=rango<=20, ingrese 0 en ambos para
terminar: 18 20
Evaluación de los exponentes 19 y 18
786433 = 2^19 + 2^18 + 1
786431 = 2^19 + 2^18 - 1
Solo 2 combinaciones de operaciones con los exponentes 19 y 18 dan como
resultado números primos de Solimas
Evaluación de los exponentes 20 y 18
1310719 = 2^20 + 2^18 - 1
786433 = 2^20 - 2^18 + 1
786431 = 2^20 - 2^18 - 1
Solo 3 combinaciones de operaciones con los exponentes 20 y 18 dan como
resultado números primos de Solimas
Evaluación de los exponentes 20 y 19
524287 = 2^20 - 2^19 - 1
Solo una combinación de operaciones con los exponentes 20 y 19 da como
resultado un número primo de Solimas
--------------------
Se encontraron 6 operaciones para hallar números primos de Solimas con exponentes
en el rango [18,20]
--------------------
Ingrese el rango para los exponentes, 2<=rango<=20, ingrese 0 en ambos para
terminar: 0 0
Fin de la evaluación
```

```
--------------------
Ingrese el rango para los exponentes, 2<=rango<=20, ingrese 0 en ambos para
terminar: -1 10
Rango incorrecto
--------------------
Ingrese el rango para los exponentes, 2<=rango<=20, ingrese 0 en ambos para
terminar: 15 22
Rango incorrecto
--------------------
Ingrese el rango para los exponentes, 2<=rango<=20, ingrese 0 en ambos para
terminar: 0 0
Fin de la evaluación
```

Programa 2: Propuesta de solución - Número Primo de Solimas

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

void evaluarPrimoSolinas(int,int );
int calcularMostrarPrimos(int ,int );
void verificarImprimirPrimo(int ,int ,char ,char ,int *,int *);
int calcularOperacion(int ,int ,char ,char );
void imprimirOperacion(int ,int , char ,char ,int );
int verificarPrimo(int );
```

```
10
11  int main(){
12          int inicio,fin;
13          do{
14                  printf("--------------------\n");
15                  printf("Ingrese el rango para los exponentes, 2<=rango<=20, ingrese 0 en ambos para terminar: ");
16                  scanf(" %d  %d",&inicio,&fin);
17                  if (inicio>=2 && fin>inicio && fin<=20)
18                          evaluarPrimoSolinas(inicio,fin);
19                  else
20                          if (inicio==0 && fin==0){
21                                  printf("Fin de la evaluación\n");
22                                  break;
23                          }
24                          else
25                                  printf("Rango incorrecto\n");
26          }while(1);
27  }
28
29  void evaluarPrimoSolinas(int inicio,int fin){
30          int esPrimo,i,j,primo,cant,suma=0;
31          char signo1,signo2;
32          for (i=inicio;i<=fin;i++)
33                  for (j=inicio;j<=fin;j++)
34                          if (i>j){
35                                  printf("Evaluación de los exponentes  %d y  %d\n",i,j);
36                                  cant=calcularMostrarPrimos(i,j);
37                                  if (cant==4)
38                                          printf("Las cuatro combinaciones de operaciones con los exponentes  %d y  %d dan como
                                                resultado números primos de Solimas\n",i,j);
39                                  else
40                                          if (cant>1)
41                                                  printf("Solo  %d combinaciones de operaciones con los exponentes  %d y  %d dan
                                                        como resultado números primos de Solimas\n",cant,i,j);
42                                          else
43                                                  if (cant==1)
44                                                          printf("Solo una combinación de operaciones con los exponentes  %d y  %
                                                                d da como resultado un número primo de Solimas\n",i,j);
45                                                  else
46                                                          printf("Ninguna combinación de operaciones con los exponentes  %d y  %
                                                                d da como resultado un número primo de Solimas\n",i,j);
47                                  suma+=cant;
48                          }
49          printf("--------------------\n");
50          printf("Se encontraron  %d operaciones para hallar números primos de Solimas con exponentes en el rango [ %d, %d]\n",suma,
                inicio,fin);
51  }
52
53  int calcularMostrarPrimos(int a,int b){
54          int num1,num2,num3,num4,esPrimoCalc1,esPrimoCalc2,esPrimoCalc3,esPrimoCalc4;
55          int calculo1,calculo2,calculo3,calculo4;
56          verificarImprimirPrimo(a,b,'+','+',&esPrimoCalc1,&calculo1);
57          verificarImprimirPrimo(a,b,'+','-',&esPrimoCalc2,&calculo2);
58          verificarImprimirPrimo(a,b,'-','+',&esPrimoCalc3,&calculo3);
59          verificarImprimirPrimo(a,b,'-','-',&esPrimoCalc4,&calculo4);
60          if (esPrimoCalc1)
61                  imprimirOperacion(a,b,'+','+',calculo1);
62          if (esPrimoCalc2)
63                  imprimirOperacion(a,b,'+','-',calculo2);
64          if (esPrimoCalc3)
65                  imprimirOperacion(a,b,'-','+',calculo3);
66          if (esPrimoCalc4)
67                  imprimirOperacion(a,b,'-','-',calculo4);
68          return esPrimoCalc1+esPrimoCalc2+esPrimoCalc3+esPrimoCalc4;
69  }
70
71  void verificarImprimirPrimo(int a,int b,char signo1,char signo2,int *esPrimo,int *calculo){
```

```
72          *calculo=calcularOperacion(a,b,signo1,signo2);
73          *esPrimo=verificarPrimo(*calculo);
74  }
75
76  int calcularOperacion(int a,int b,char signo1,char signo2){
77          int num,calc;
78          calc=pow(2,a);
79          if (signo1=='+')
80                  calc+=pow(2,b);
81          else
82                  calc-=pow(2,b);
83          if (signo2=='+')
84                  calc+=1;
85          else
86                  calc-=1;
87          return calc;
88  }
89
90  void imprimirOperacion(int a,int b, char signo1,char signo2,int calculo){
91          printf(" %d = 2^ %d  %c 2^ %d  %c 1\n",calculo,a,signo1,b,signo2);
92  }
93
94  int verificarPrimo(int numero){
95          int cantDivisores=0,i=1;
96          while (i<=numero){
97                  if (numero %i==0)
98                          cantDivisores++;
99                  i++;
100         }
101         return cantDivisores==2;
102 }
```

3. Teorema de Zeckendorf

El teorema de Zeckendorf indica que todo número entero positivo se puede formar por la suma de números no consecutivos de la sucesión de Fibonacci.

La Sucesión de Fibonacci es una conocida sucesión que empieza con 0 y 1, y a partir de ese momento, cada término que sigue es la suma de los dos anteriores.

Los primeros números de la serie de Fibonacci son: 0,1,1,2,3,5,8,13,21,34,55,89, ...

Por ejemplo,

- Para el número 2, 2=0+2, 0 es el término 1 de la serie y 2 es el término 4
- Para el número 3, 3=1+2, 1 es el témino 2 de la serie y 2 es el término 4
- Para el número 4, 4=1+3, 1 es el témino 2 de la serie y 3 es el término 5
- Para el número 6, 6=1+5, 1 es el témino 3 de la serie y 5 es el término 6
- Para el número 7, 7=2+5, 2 es el témino 4 de la serie y 5 es el término 6
- Para el número 17, 17=1+3+13, 1 es el témino 2 de la serie, 3 es el término 5 y 13 es el término 8

Se pide que desarrolle un programa en Lenguaje C que reciba un rango de números enteros positivos y que para cada número en el rango encuentre todas las sumas de términos fibonacci no consecutivos posibles que formen el número. Cuando encuentre una suma que forme el número evaluado debe mostrar las posiciones de los términos Fibonacci que forman parte de la suma con sus respectivos valores. Para finalizar, debe mostrar si en el rango de números evaluados se cumple o no el teorema de Zeckendorf, siguiendo los pasos que se describen a continuación.

Para hallar la suma debe intercalar los términos de 2 en 2, 3 en 3, 4 en 4 y así sucesivamente hasta que la intercalación se realice con el número de la posición del término más cercano de Fibonacci al número evaluado.

Por ejemplo, para encontrar las sumas para el número 3, debe realizar lo siguiente:

- Iniciar sumando los términos de las posiciones 1 y 3. Esto es 0+1=1, pero 1 es aún menor que 3 así que continúe con el término 5, pero el término 5 es 3 y además 1+3= 4 que es mayor que 3; así que intercalando los términos de 1 en 1 empezando en el término 1 no encontrará 3.
- Continúe con una nueva suma, intercalando los términos desde el término 2, sumando los términos 2,4. Esto es 1+2=3, entonces encontró una suma que cumple con el teorema.
- Continúe con una nueva suma, intercalando los términos desde el término 3, sumando los términos 3,5. Pero el término 5 es 3 y además 1+3=4 que es mayor que 3; así que intercalando los términos de 1 en 1 empezando en el término 3 no encontrará 3.
- Continúe con una nueva suma, intercalando los términos desde el término 4, sumando los términos 4,6. Pero el término 6 es 5 que es mayor que 3, por lo cual ya no debe seguir con la suma; entonces intercalando los términos de 1 en 1 empezando en el término 4 no encontrará 3.
- Continúe con una nueva suma, intercalando los términos desde el término 5. Pero el término 5 es 3 que es igual que 3, por lo cual ya no debe seguir con la suma; entonces intercalando los términos de 1 en 1 empezando en el término 5 no encontrará 3.
- Como el término 5 que tiene el valor de 3 es igual al número buscado, ya no debe seguir intentando con la intercalación de 1 en 1; debe pasar a calcular las sumas intercalando los términos de 2 en 2. Para lo cual debe realizar los mismos pasos que los decritos anteriormente.
- Como el término de Fibonacci más cercano a 3 es el término 5 que en este caso es igual, como máximo debe intercalar los términos de 4 en 4. Esto es, en la primera suma intercalando 4 términos, empieza con el término 1 y suma el término 5; esto es 0+3=3 entonces encontró una suma que cumple con el teorema. Ya no puede seguir calculando más sumas, porque si al término 2, cuyo valor es 1 le suma el término 6, cuyo valor es 5; 5 es mayor que 3.
- Si está evaluando el número 7, el término de Fibonacci más cercano es el término 6 que tiene valor 5; por lo tanto como máximo debe intercalar los términos de 5 en 5.

Para el desarrollo del programa debe desarrollar como mínimo 4 módulos adicionales al main. De estos módulos, por lo menos 3 deben devolver un valor.

Uno de los módulos debe imprimir el detalle de la suma que forma el número evaluado según el teorema descrito, para lo cual debe recibir el número evaluado, el número del término de Fibonacci donde inicia la suma y el número que representa a la intercalación de términos. Por ejemplo, si este módulo se invoca con los parámetros 4,1,2; se muestra 4 = f(1) 0 + f(3) 1 + f(5) 3

Use los siguientes casos de prueba para verificar si su solución está correcta.

```
Ingrese el rango:1 4
Número 1
----------
1 =  f(1) 0  +  f(3) 1
Se encontraron 1 series de sumas para el número 1
Número 2
----------
2 =  f(1) 0  +  f(4) 2
Se encontraron 1 series de sumas para el número 2
Número 3
----------
3 =  f(1) 0  +  f(5) 3
3 =  f(2) 1  +  f(4) 2
Se encontraron 2 series de sumas para el número 3
Número 4
----------
4 =  f(1) 0  +  f(3) 1  +  f(5) 3
4 =  f(2) 1  +  f(5) 3
4 =  f(3) 1  +  f(5) 3
Se encontraron 3 series de sumas para el número 4
-----------------
En el rango [1,4] se cumple el Teorema de Zeckendorf
```

```
Ingrese el rango:8 11
Número 8
----------
8 =  f(1) 0  +  f(7) 8
8 =  f(2) 1  +  f(4) 2  +  f(6) 5
Se encontraron 2 series de sumas para el número 8
Número 9
----------
9 =  f(2) 1  +  f(7) 8
9 =  f(3) 1  +  f(7) 8
Se encontraron 2 series de sumas para el número 9
Número 10
----------
10 =  f(1) 0  +  f(4) 2  +  f(7) 8
10 =  f(4) 2  +  f(7) 8
Se encontraron 2 series de sumas para el número 10
Número 11
----------
11 =  f(5) 3  +  f(7) 8
Se encontraron 1 series de sumas para el numero 11
-----------------
En el rango [8,11] se cumple el Teorema de Zeckendorf
```

```
Ingrese el rango:19 21
Número 19
----------
No se encuentran series de sumas para el número 19
Número 20
----------
20 =  f(4) 2  +  f(6) 5  +  f(8) 13
Se encontraron 1 series de sumas para el número 20
Número 21
----------
21 =  f(1) 0  +  f(9) 21
21 =  f(2) 1  +  f(4) 2  +  f(6) 5  +  f(8) 13
Se encontraron 2 series de sumas para el número 21
-----------------
En el rango [19,21] no se cumple el Teorema de Zeckendorf
```

```
Ingrese el rango:0 5
Rango incorrecto
```

```
Ingrese el rango: 10 8
Rango incorrecto
```

Programa 3: Propuesta de solución - Teorema de Zeckendorf

```c
#include <stdio.h>

int calcularFibonacciPosicion(int );
int buscarPosFibonacciCercana(int );
int encontrarMostrarSeriesFibonacci(int );
void imprimirSeries(int ,int ,int );

int main(){
        int inicio,fin,i=1,fibo,cantSumas,cantNumerosZeck=0;
        printf("Ingrese el rango:");
```

```
11          scanf("%d %d",&inicio,&fin);
12          if (inicio>0 && inicio<fin){
13                  for (i=inicio;i<=fin;i++){
14                          cantSumas=encontrarMostrarSeriesFibonacci(i);
15                          if (cantSumas>0)
16                                  cantNumerosZeck++;
17                          if (cantSumas==0)
18                                  printf("No se encuentran series de sumas para el número %d\n",i);
19                          else
20                                  printf("Se encontraron %d series de sumas para el número %d\n",cantSumas,i);
21                  }
22                  printf("----------\n");
23                  if (fin-inicio+1==cantNumerosZeck)
24                          printf("En el rango [ %d, %d] se cumple el Teorema de Zeckendorf\n",inicio,fin);
25                  else
26                          printf("En el rango [ %d, %d] no se cumple el Teorema de Zeckendorf\n",inicio,fin);
27          }
28          else
29                  printf("Rango incorrecto\n");
30          return 0;
31  }
32
33  int calcularFibonacciPosicion(int posicion){
34          int pos=2,i,fn_2=0,fn_1=1,fn=1;
35          if (posicion==1)
36                  return fn_2;
37          else
38                  if (posicion==2)
39                          return fn_1;
40                  else{
41                          while(pos<posicion){
42                                  fn=fn_1+fn_2;
43                                  fn_2=fn_1;
44                                  fn_1=fn;
45                                  pos++;
46                          }
47                          return fn;
48                  }
49  }
50
51  int buscarPosFibonacciCercana(int numero){
52          int posicion=0,fibo;
53          while(1){
54                  posicion++;
55                  fibo=calcularFibonacciPosicion(posicion);
56                  if (fibo>=numero) break;
57          }
58          if (numero==1)
59                  return posicion+1;
60          else
61                  if (fibo==numero)
62                          return posicion;
63                  else
64                          return posicion-1;
65  }
66
67  int encontrarMostrarSeriesFibonacci(int numero){
68          int i,suma=0,cantSumas=0,fiboSumar,fiboSumarIni;
69          int posicion=1,posInicio=1,maxI;
70          maxI=buscarPosFibonacciCercana(numero);
71          printf("Número %d\n",numero);
72          printf("----------\n");
73          while(1){
74                  suma=0;
75                  fiboSumarIni=calcularFibonacciPosicion(posInicio);
76                  posicion=posInicio;
77                  if (fiboSumarIni>=numero) break;
```

```
78                  for (i=2;i<=maxI;i++){
79                          fiboSumar=fiboSumarIni;
80                          suma=0;
81                          posicion=posInicio;
82                          while(fiboSumar<=numero && suma<numero){
83                                  suma+=fiboSumar;
84                                  posicion+=i;
85                                  fiboSumar=calcularFibonacciPosicion(posicion);
86                          }
87                          if (suma==numero){
88                                  cantSumas++;
89                                  imprimirSeries(numero,posInicio,i);
90                          }
91                  }
92                  posInicio++;
93          }
94          return cantSumas;
95  }
96
97  void imprimirSeries(int numero,int posicion,int separacion){
98          int fibo,suma=0,i=0;
99          printf(" %d = ",numero);
100         while(suma<numero){
101                 fibo=calcularFibonacciPosicion(posicion);
102                 if (i>0)
103                         printf(" + ");
104                 printf(" f( %d)  %d ",posicion,fibo);
105                 suma+=fibo;
106                 posicion+=separacion;
107                 i++;
108         }
109         printf("\n");
110 }
```

4. Números de Fibonacci

Se pide que desarrolle un programa en lenguaje C que reciba el rango inicial y final de un intervalo de números positivos (el intervalo debe ser como máximo de 15 números incluyendo a y b) y muestre en un gráfico de barras los números que pertenecen a la sucesión de Fibonacci; en el eje x debe mostrar la posición de los términos y en el eje y debe mostrar el valor de los términos.

Este programa debe mostrar mensajes específicos ante las siguientes situaciones:

- Si el rango no cumple que $b > a$ y $a > 0$ y la cantidad de números del intervalo es como máximo 15, entonces debe mostrar el mensaje "El rango ingresado no es correcto".
- Si no se encontraron números Fibonacci dentro del rango, entonces debe mostrar el mensaje "No se encontraron números Fibonacci dentro del rango".

Para el desarrollo del programa debe considerar, por lo menos, la implementación de los siguientes módulos (además del programa principal):

- Un módulo que lea el rango inicial y final (a y b).
- Un módulo que determine si un número es de Fibonacci.
- Un módulo que devuelva la posición de un número dentro de la sucesión de Fibonacci.
- Un módulo que devuelva la cantidad de números de Fibonacci en un intervalo entre a y b.
- Un módulo que calcule si un número es un cuadrado perfecto. Para el desarrollo de este módulo debe utilizar estructuras iterativas y no podrá utilizar la función $sqrt$.
- Un módulo que imprima el gráfico solicitado.

Recordar que:

La sucesión de Fibonacci es la sucesión infinita de los siguientes número naturales:

0,1,1,2,3,5,8,13,21,34,55,89,144,...

Donde:

- El término en la posición 0 es 0
- El término en la posición 1 es 1
- El término en la posición 2 es 1
- El término en la posición 3 es 2

Notar que los términos en la posición 1 y 2 son iguales a 1.

En general los números de la sucesión de Fibonacci se pueden expresar con las siguientes fórmulas (n es la posición del término):

- $f_0 = 0$
- $f_1 = 1$
- $f_n = f_{n-1} + f_{n-2}$

Recordar que:

Un número (N) es de Fibonacci sí y solo sí la expresión $5N^2 + 4$ o $5N^2 - 4$ es un cuadrado perfecto.

Caso de prueba 1

```
Ingrese el inicio del rango (a): 10
Ingrese el final del rango (b): 1
El rango ingresado no es correcto.
```

Caso de prueba 2

```
Ingrese el inicio del rango (a): 22
Ingrese el final del rango (b): 30
No se encontraron numeros fibonacci dentro del rango.
```

Caso de prueba 3

```
Ingrese el inicio del rango (a): 1
Ingrese el final del rango (b): 7

En la siguiente grafica se muestran los numeros fibonacci del rango ingresado:

                                        Fibonacci(n)
                                    |   5
                                    |   4
                            |       |   3
                        |   |       |   2
                    |   |   |       |   1
posicion (n):   1 o 2   3       4       5
```

Caso de prueba 4

```
Ingrese el inicio del rango (a): 5
Ingrese el final del rango (b): 21
El rango ingresado no es correcto.
```

Caso de prueba 5

```
Ingrese el inicio del rango (a): 6
Ingrese el final del rango (b): 18

En la siguiente grafica se muestran los numeros fibonacci dentro del rango ingresado:

                              Fibonacci(n)
                         |    13
                         |    12
                         |    11
                         |    10
                         |    9
                    |    |    8
                    |    |    7
                    |    |    6
                    |    |    5
                    |    |    4
                    |    |    3
                    |    |    2
                    |    |    1
posicion (n):   6        7
```

Programa 4: Propuesta de solución - Números de Fibonacci

```
#include <stdio.h>
#include <math.h>
void leerDatos(int *,int *);
int obtenerCantFib(int, int);
int numeroFib(int);
int devolverPosicion(int );
int determinarCuadrado(int );
void imprimirCabecera(int);
void imprimirPieGrafica(int, int);
void dibujarBarras(int ,int );

int main() {
        int a, b, cantNumFib=0;
        leerDatos(&a, &b);
        if ((a>=0) && (a<b) && (b-a+1<=15)) {
                cantNumFib=obtenerCantFib(a,b);
                if (cantNumFib>=1) {
                        imprimirCabecera(cantNumFib);
                        dibujarBarras(a,b);
                        imprimirPieGrafica(a,b);
                } else
                        printf("No se encontraron numeros fibonacci dentro del rango.\n");
        } else
                printf("El rango ingresado no es correcto.\n");
        return 0;
}
void leerDatos(int *a, int *b) {
        printf("Ingrese el inicio del rango (a): ");
        scanf(" %d", a);
        printf("Ingrese el final del rango (b): ");
        scanf(" %d", b);
```

```
}
int obtenerCantFib(int a,int b) {
	int i=0, esFib=0, contador=0;
	for (i=a; i<=b; i++) {
		esFib=numeroFib(i);
		if (esFib!=-1)
			contador++;
	}
	return contador;
}
int numeroFib(int num) {
	int expresion1, expresion2, esCuadrado;
	expresion1=5*pow(num,2)+4;
	expresion2=5*pow(num,2)-4;
	esCuadrado=determinarCuadrado(expresion1) || determinarCuadrado(expresion2);
	if (esCuadrado)
		return num;
	return -1;
}
int determinarCuadrado(int num) {
	int i,j, esCuadrado=0;
	for(i=0; i<=num;i++) {
		for(j=2;j<=num/2;j=j+2) {
			if (pow(i,j)==num){
				esCuadrado=1;
				return esCuadrado;
			}
		}
	}
	return 0;
}
int devolverPosicion(int num) {
	int encontrado=0, i=2, esFib, posicion, t0=0, t1=1,tn;
	if (num==0) {
		posicion=0;
		encontrado=1;
	} else if (num==1) {
		posicion=1;
		encontrado=1;
	}
	while (!encontrado) {
		tn=t0+t1;
		t0=t1;
		t1=tn;
		if (tn==num) {
			posicion=i;
			encontrado=1;
		}
		i++;
	}
	return posicion;
}
void imprimirCabecera(int final) {
	int cabeceraFinal, i;
	cabeceraFinal=final+3; /*porque recorre desde a hasta b+3; es decir b+3 saltos*/
	printf("\nEn la siguiente gráfica se muestran los números fibonacci dentro del rango ingresado:\n\n");
	for (i=1;i<=cabeceraFinal;i++) {
		if (i==cabeceraFinal)
			printf("Fibonacci(n)\n");
		else
			printf("\t");
	}
}
void imprimirPieGrafica(int a,int b) {
	int i,numFib,pos;
	printf("posición (n):\t");
	for (i=a; i<=b;i++) {
```

```
99                      numFib=numeroFib(i);
100                     if (numFib!=-1) {
101                             pos=devolverPosicion(numFib);
102                             if (pos==1)
103                                     printf("1 o 2\t");
104                             else
105                                     printf(" %d\t",pos);
106                     }
107             }
108     }
109     void dibujarBarras(int a,int b) {
110             int i,filas=1, enBlanco,j, altura,k, numFib;
111             for (i=a; i<=b; i++) {
112                     altura=numeroFib(i);
113                     if (altura)
114                             if (altura>=filas)
115                                     filas=altura;
116             }
117             /*Con el número de filas empezamos a dibujar la gráfica*/
118             for (j=1; j<=filas;j++) {
119                     printf("\t\t");
120                     for (k=a;k<=b; k++) {
121                             numFib=numeroFib(k);
122                             if (numFib!=-1) {
123                                     altura=numeroFib(k);
124                                     enBlanco=filas-altura;
125                                     if (j<=enBlanco)
126                                             printf(" \t");
127                                     else
128                                             printf("|\t");
129                             }
130                     }
131                     printf(" %d\n",filas-j+1);
132             }
133     }
```

Puede usar cualquier estructura selectiva y cualquier estructura iterativa.