

FUNDAMENTOS DE PROGRAMACIÓN
LABORATORIO 6
PROPUESTAS DE SOLUCIÓN
SEMESTRE ACADÉMICO 2022-2

Horarios: Todos los horarios

Elaborado por Mag. Rosa Latorraca

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 comprenderá el funcionamiento de la estructura algorítmica selectiva múltiple.
- Al finalizar la sesión, el alumno construirá programas usando estructuras algorítmicas selectivas múltiple.
- Al finalizar la sesión, el alumno comprenderá el funcionamiento de la estructura algorítmica iterativa con salida controlada.
- Al finalizar la sesión, el alumno construirá programas usando la estructura algorítmica iterativa con salida controlada.

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 belgas y oblongos - Horarios: 0388, 0389, 0390

Un número es **k-belga** si se cumple la siguiente propiedad: si a partir de k se suma reiteradamente las cifras del número dado, se forma una sucesión que contiene a ese número.

Por ejemplo: el 18 es 0-belga, porque a partir del 0 vamos a ir sumando 1, 8, 1, 8 y así sucesivamente hasta llegar al número 18: $0+1 = 1$, $1+8 = 9$, $9+1 = 10$, $10+8 = 18$. Como se alcanza el 18, entonces es 0-belga.

El número 19 no es 1-belga, porque a partir del 1 vamos a ir sumando 1, 9, 1, 9 y así sucesivamente hasta llegar al número 19: $1+1 = 2$, $2+9 = 11$, $11+1 = 12$, $12+9 = 21$. Como se sobrepasó al número 19, entonces no es 1-belga.

Un **número oblongo** es un número que es el producto de dos números naturales consecutivos. Por ejemplo, 42 es un número oblongo porque $42 = 6 \times 7$.

Se pide desarrollar un programa en Lenguaje C que permita realizar lo siguiente:

- Muestre un menú de opciones, las cuales son O: Números oblongos. B: Números belgas.
- Leer una opción. Debe validar que la opción ingresada sea O o B. Si elige una opción distinta a las indicadas, se mostrará el mensaje: Opción inválida y el programa debe terminar.
- Leer un número.

- Debe validar que el número ingresado sea mayor a 0. En caso contrario, debe emitir el siguiente mensaje: El número ingresado debe ser mayor a 0 y el programa debe terminar.
- Si el usuario elige la opción B, debe solicitar k para la verificación del número k-belga. El valor de k debe ser mayor o igual a 0 y menor o igual a 9. En caso contrario, debe emitir el siguiente mensaje: El valor de k debe estar en el rango de [0,9] y el programa debe terminar.

La solución deberá contar como mínimo 3 módulos, incluido el módulo principal. Además, uno de los módulos debe devolver exactamente dos valores como parámetros que modifican su valor y no puede usarse para leer datos.

Consideraciones adicionales

Para la solución del problema debe tener en cuenta lo siguiente:

- La cantidad de cifras del número k-belga siempre será de dos cifras. Es decir, no debe validar que la cantidad de cifras sea 2.
- Considerar como opciones válidas solo la letra O y B en mayúsculas.

A continuación se presentan los siguientes ejemplos de ejecución:

Caso de prueba 1

Menú de opciones:

O: Números oblongos.

B: Números belgas.

Ingrese la opción: F

Opción inválida

Caso de prueba 2

Menú de opciones:

O: Números oblongos.

B: Números belgas.

Ingrese la opción: b

Opción inválida

Caso de prueba 3

Menú de opciones:

O: Números oblongos.

B: Números belgas.

Ingrese la opción: O

Ingrese un número: -1

El número ingresado debe ser mayor a 0.

Caso de prueba 4

Menú de opciones:

O: Números oblongos.

B: Números belgas.

Ingrese la opción: O

Ingrese un número: 30

El número ingresado es oblongo.

$$30 = 5 \times 6$$

Caso de prueba 5

Menú de opciones:

O: Números oblongos.

B: Números belgas.

Ingrese la opción: O

Ingrese un número: 40

El número ingresado no es oblongo.

Caso de prueba 6

Menú de opciones:

O: Números oblongos.

B: Números belgas.

Ingrese la opción: B

Ingrese un número: -10

El número ingresado debe ser mayor a 0.

Caso de prueba 7

Menú de opciones:

O: Números oblongos.

B: Números belgas.

Ingrese la opción: B

Ingrese un número: 20

Ingrese el k del número belga: -1

El valor de k debe estar en el rango de [0,9].

Caso de prueba 8

Menú de opciones:

O: Números oblongos.

B: Números belgas.

Ingrese la opción: B

Ingrese un número: 22

Ingrese el k del número belga: 2

El número ingresado es 2-belga.

Caso de prueba 9

Menú de opciones:

O: Números oblongos.

B: Números belgas.

Ingrese la opción: B

Ingrese un número: 25

Ingrese el k del número belga: 3

El número ingresado no es 3-belga.

Programa 1: Propuesta de solución - Números belgas y oblongos

```
1 #include <stdio.h>
2
3 /*Prototipos*/
4 void esNumeroOblongo(int numero,int *esOblongo,int *factor);
5 int siEsBelga(int numero, int k);
6
7 int main(){
8     /*Declaración de variables*/
9     int numero,k;
10    char opcion;
11    int esOblongo,factor;
12
13    printf("Menú de opciones:\n");
14    printf("O: Números oblongos.\n");
15    printf("B: Números belgas.\n");
16    printf("\nIngrese la opción: ");
17    scanf("\n %c",&opcion);
18    if(opcion=='O'||opcion=='B'){
19        printf("\nIngrese un número: ");
20        scanf("%d",&numero);
21        if(numero>0){
22            if(opcion=='O'){
23                esNumeroOblongo(numero,&esOblongo,&factor);
24                if(esOblongo){
25                    printf("El número ingresado es oblongo: \n");
26                    printf("%d= %d x %d\n",numero,factor,factor+1);
27                }
28                else{
29                    printf("El número ingresado no es oblongo.\n");
30                }
31            }
32        }
33        printf("Ingrese el k del número belga: ");
34        scanf("%d",&k);
```

```

35         if(k>=0 && k<=9){
36             if(siEsBelga(numero,k)){
37                 printf("\nEl número ingresado es %d-belga.\n",k);
38             }
39             else{
40                 printf("\nEl número ingresado no es %d-belga.\n",k);
41             }
42         }
43         else{
44             printf("El valor de k debe estar en el rango de [0,9].\n");
45         }
46     }
47     else{
48         printf("El número ingresado debe ser mayor a 0.\n");
49     }
50 }
51 else{
52     printf("Opción inválida.\n");
53 }
54
55
56
57     return 0;
58 }

59 /*Funciones*/
60 void esNumeroOblongo(int numero,int *esOblongo,int *factor){
61     int i=1,producto;
62
63     producto=1;
64
65     do{
66         producto=i*(i+1);
67         i++;
68     }while(numero>producto);
69
70     if(numero==producto){
71         *esOblongo=1;
72         *factor=i-1;
73     }
74     else{
75         *esOblongo=0;
76         *factor=0;
77     }
78 }
79
80 int siEsBelga(int numero, int k){
81     int suma,unidad,decena;
82
83     unidad=numero %10;
84     decena=numero/10;
85
86     suma=k;
87
88     do{
89         suma=suma+unidad+decena;
90     }while(numero>suma);
91
92
93     return suma==numero;
94 }
```

2. Multiplicación rusa e hindú - Horarios: B301, 0384, 0385, 0391

El método de **multiplicación rusa** consiste en multiplicar sucesivamente por 2 el multiplicando y dividir por 2 el multiplicador hasta que el multiplicador tome el valor 1. Luego, se suman todos los multiplicandos correspondientes a los multiplicadores impares. Dicha suma es el producto de los dos números.

Por ejemplo: en el siguiente cuadro 1, se muestra el cálculo realizado para multiplicar 37 por 12, cuyo resultado final es $12 + 48 + 384 = 444$.

Multiplicador	Multiplicando	Multiplicador	Suma
37	12	37	12
18	24	18	-
9	48	9	48
4	96	4	-
2	192	2	-
1	384	1	384

Cuadro 1: Multiplicación rusa

El método de la **multiplicación hindú** consiste en colocar dos números que se desean multiplicar: uno horizontalmente y otro verticalmente. Se repite el número que está de forma horizontal debajo del anterior pero desplazando un lugar hacia la derecha y así tantas veces la cantidad de cifras del número vertical. Se multiplica cada dígito vertical por cada número horizontal. El resultado de la multiplicación es la suma de las multiplicaciones parciales.

Por ejemplo: en la siguiente figura 1, se muestra el cálculo realizado para multiplicar 264 por 129 cuyo resultado final es 34056.

$$\begin{array}{r} & 2 & 1 & 2 & 9 \\ & 6 & 1 & 2 & 9 \\ & 4 & 1 & 2 & 9 \\ \hline & & 2 & 5 & 8 \\ & & 7 & 7 & 4 \\ & & 5 & 1 & 6 \\ \hline & & 3 & 4 & 0 & 5 & 6 \end{array}$$

Figura 1: Multiplicación hindú

Se pide desarrollar un programa en Lenguaje C que permita realizar lo siguiente:

- Leer dos números.

- Debe validar que los números ingresados sean mayores a 0. En caso contrario, debe emitir el siguiente mensaje: Los números ingresados deben ser mayores a 0 y el programa debe terminar.
- Muestre un menú de opciones, las cuales son R: Multiplicación rusa. H: Multiplicación hindú.
- Leer una opción. Debe validar que la opción ingresada sea R o H. Si elige una opción distinta a las indicadas, se mostrará el mensaje: Opción inválida y el programa debe terminar.

La solución deberá contar como mínimo 3 módulos, incluido el módulo principal. Además, uno de los módulos debe devolver exactamente dos valores como parámetros que modifican su valor y no puede usarse para leer datos.

Consideraciones adicionales

Para la solución del problema debe tener en cuenta lo siguiente:

- No puede utilizar la función log10 de la librería math.h por ningún motivo.
- En el caso de la multiplicación rusa, se debe cumplir que el multiplicando sea menor al multiplicador.
- Considerar como opciones válidas solo la letra R y H en mayúsculas.

A continuación se presentan los siguientes ejemplos de ejecución:

Caso de prueba 1

Ingrese dos números a multiplicar: 0 37
Los números ingresados deben ser mayores a 0.

Caso de prueba 2

Ingrese dos números a multiplicar: 37 12

Menú de opciones:
R: Multiplicación rusa
H: Multiplicación hindú

Ingrese la opción: F
Opción inválida.

Caso de prueba 3

Ingrese dos números a multiplicar: 380 125

Menú de opciones:
R: Multiplicación rusa
H: Multiplicación hindú

Ingrese la opción: r
Opción inválida.

Caso de prueba 4

Ingrese dos números a multiplicar: 58 62

Menú de opciones:

R: Multiplicación rusa

H: Multiplicación hindú

Ingrese la opción: R

El resultado de la multiplicación es: 3596

Caso de prueba 5

Ingrese dos números a multiplicar: 72 15

Menú de opciones:

R: Multiplicación rusa

H: Multiplicación hindú

Ingrese la opción: R

El resultado de la multiplicación es: 1080

Caso de prueba 6

Ingrese dos números a multiplicar: 121 239

Menú de opciones:

R: Multiplicación rusa

H: Multiplicación hindú

Ingrese la opción: H

El resultado de la multiplicación es: 28919

Caso de prueba 7

Ingrese dos números a multiplicar: 785 42

Menú de opciones:

R: Multiplicación rusa

H: Multiplicación hindú

Ingrese la opción: H

El resultado de la multiplicación es: 32970

Programa 2: Propuesta de solución - Multiplicación rusa e hindú

```
1 #include <stdio.h>
2 #include <math.h>
3
4 /*Prototipos*/
5 int multiplicacionRusa(int numero1,int numero2);
6 int multiplicacionHindu(int numero1,int numero2);
```

```

7 int evaluarCantidadCifras(int numero);
8 void MayorYMenor(int numero1,int numero2,int *mayor,int *menor);
9
10 int main(){
11
12     /*Declaración de variables*/
13     int numero1,numero2,resultado,mayor,menor;
14     char operador;
15
16     printf("Ingrese dos números a multiplicar: ");
17     scanf("%d %d",&numero1,&numero2);
18
19     if(numero1>0 && numero2>0){
20         printf("\nMenú de opciones:\n");
21         printf("R: Multiplicación rusa\n");
22         printf("H: Multiplicación hindú\n");
23         printf("\nIngrese la opción: ");
24         scanf("\n%c",&operador);
25         if(operador=='R'||operador=='H'){
26             if(operador=='R'){
27                 MayorYMenor(numero1,numero2,&mayor,&menor);
28                 resultado=multiplicacionRusa(mayor,menor);
29             }
30             else{
31                 resultado=multiplicacionHindu(numero1,numero2);
32             }
33             printf("\nEl resultado de la multiplicación es: %d\n",resultado);
34         }
35         else{
36             printf("Opción inválida.\n");
37         }
38     }
39     else{
40         printf("Los números ingresados deben ser mayores a 0.\n");
41     }
42
43     return 0;
44 }
45
46 /*Funciones*/
47 int multiplicacionRusa(int numero1,int numero2){
48     int suma=0;
49     do{
50         if(numero1 %2==1){
51             suma=suma+numero2;
52         }
53         numero1=numero1/2;
54         numero2=numero2*2;
55     }while(numero1>=1);
56
57     return suma;
58 }
59
60 int multiplicacionHindu(int numero1,int numero2){
61     int i=1,suma=0,dígito;
62     int cantCifras=evaluarCantidadCifras(numero1);
63
64     do{
65         dígito=numero1 %10;
66         numero1=numero1/10;
67         suma=suma+dígito*numero2*pow(10,i-1);
68         i++;
69     }while(i<=cantCifras);
70
71     return suma;
72 }
73

```

```

74
75 void MayorYMenor(int numero1,int numero2,int *mayor,int *menor){
76     if(numero1<numero2){
77         *mayor=numero2;
78         *menor=numero1;
79     }
80     else{
81         *mayor=numero1;
82         *menor=numero2;
83     }
84 }
85
86 int evaluarCantidadCifras(int numero){
87     int i=0;
88     do{
89         i++;
90         numero=numero/10;
91     }while(numero!=0);
92     return i;
93 }
```

3. Sucesión de números - Horarios: 0380, 0381, 0386, 0387

Las sucesiones de números son secuencias de números naturales que siguen unas reglas de formación determinadas. Existen algunas sucesiones que tienen utilidad práctica como la **sucesión de Padovan**. La sucesión de Padovan fue nombrada por el matemático Richard Padovan y tiene varios usos como por ejemplo el mostrado en la figura 2

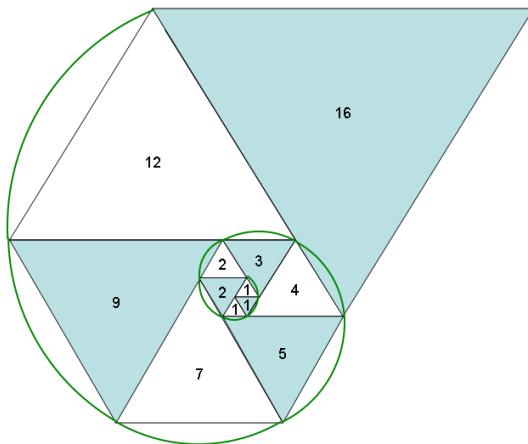


Figura 2: Sucesión de Padovan

Para calcular los números de la sucesión de Padovan se utiliza la función `padovan(n)`

$$padovan(n) = \begin{cases} 1 & n = 0 \\ 1 & n = 1 \\ 1 & n = 2 \\ padovan(n - 2) + padovan(n - 3) & n \geq 3 \end{cases}$$

Los primeros 20 términos de la sucesión de Padovan son: 1, 1, 1, 2, 2, 3, 4, 5, 7, 9, 12, 16, 21, 28, 37, 49, 65, 86, 114, 151.

También existe la **sucesión de Perrin** que es una sucesión con la misma regla de formación que la de Padovan, pero que utiliza diferentes valores iniciales, fue estudiada por el matemático francés Edouard Lucas en 1876. En 1899 sus ideas fueron desarrolladas por R. Perrin y por ello la sucesión se conoce como sucesión de Perrin.

Para calcular los números de la sucesión de Perrin se utiliza la función $perrin(n)$.

$$perrin(n) = \begin{cases} 3 & n = 0 \\ 0 & n = 1 \\ 2 & n = 2 \\ perrin(n - 2) + perrin(n - 3) & n \geq 3 \end{cases}$$

Los primeros 20 términos de la sucesión de Perrin son: 3, 0, 2, 3, 2, 5, 5, 7, 10, 12, 17, 22, 29, 39, 51, 68, 90, 119, 158, 209.

Además, la sucesión de Perrín se puede obtener a partir de la sucesión de Padovan mediante la siguiente fórmula:

$$perrin(n) = padovan(n + 1) + padovan(n - 10)$$

Se pide desarrollar un programa en Lenguaje C que permita realizar lo siguiente:

- Muestre un menú de opciones, las cuales son P: Sucesión de Padovan. R: Sucesión de Perrin.
- Leer una opción. Debe validar que la opción ingresada sea P o R. Si elige una opción distinta a las indicadas, se mostrará el mensaje: Opción inválida y el programa debe terminar.
- Leer el término de la sucesión.
- Debe validar que el término de la sucesión ingresado sea mayor a 0. En caso contrario, debe emitir el siguiente mensaje: El término de la sucesión ingresado debe ser mayor a 0 y el programa debe terminar.

La solución deberá contar como mínimo 3 módulos, incluido el módulo principal. Además, uno de los módulos debe devolver exactamente dos valores como parámetros que modifican su valor y no puede usarse para leer datos.

Consideraciones adicionales

Para la solución del problema debe tener en cuenta lo siguiente:

- Se debe usar iterativa para calcular la sucesión de números.
- Considerar como opciones válidas solo la letra P y R en mayúscula.

A continuación se presentan los siguientes ejemplos de ejecución:

Caso de prueba 1

Menú de opciones:

P: Sucesión de Padovan.

R: Sucesión de Perrin.

Ingrese una opción: F

Opción inválida.

Caso de prueba 2

Menú de opciones:

P: Sucesión de Padovan.

R: Sucesión de Perrin.

Ingrese una opción: r

Opción inválida.

Caso de prueba 3

Menú de opciones:

P: Sucesión de Padovan.

R: Sucesión de Perrin.

Ingrese una opción: P

Ingrese el término de la sucesión: -1

El término de la sucesión debe ser mayor o igual a 0.

Caso de prueba 4

Menú de opciones:

P: Sucesión de Padovan.

R: Sucesión de Perrin.

Ingrese una opción: R

Ingrese el término de la sucesión: -5

El término de la sucesión debe ser mayor o igual a 0.

Caso de prueba 5

Menú de opciones:

P: Sucesión de Padovan.

R: Sucesión de Perrin.

Ingrese una opción: P

Ingrese el término de la sucesión: 20

El término 20 de la sucesión de Padovan es 151.

Caso de prueba 6

Menú de opciones:

P: Sucesión de Padovan.

R: Sucesión de Perrin.

Ingrese una opción: R

Ingrese el término de la sucesión: 15

El término 15 de la sucesión de Perrin es 51.

El término 15 de la sucesión de Perrin mediante la fórmula de la sucesión de Padovan es 51.

Programa 3: Propuesta de solución - Sucesión de números

```
1 #include <stdio.h>
2
3 /*Prototipos*/
4 int calcularPadovan(int termino);
```

```

5 void calcularPerrin(int termino, int *sucesion, int *numeroPadovan);
6
7 int main(){
8
9     /*Declaración de variables*/
10    int termino,sucesion,numeroPerrin;
11    char opcion;
12
13    printf("Menú de opciones:\n");
14    printf("P: Sucesión de Padovan.\n");
15    printf("R: sucesión de Perrín.\n");
16    printf("\nIngrese una opción:\n");
17    scanf("\n %c",&opcion);
18
19    if(opcion=='P'||opcion=='R'){
20        printf("\nIngrese el término de la sucesión: ");
21        scanf("%d",&termino);
22        if(termino>0){
23            if(opcion=='P'){
24                sucesion=calcularPadovan(termino);
25                printf("El término %d de la sucesión de Padovan es %d.\n",termino,sucesion);
26            }
27            else{
28                calcularPerrin(termino,&sucesion,&numeroPerrin);
29                printf("El término %d de la sucesión de Perrin es %d.\n",termino,sucesion);
30                printf("El término %d de la sucesión de Perrin mediante la fórmula de la sucesión de Padovan es %d\n",termino,numeroPerrin);
31            }
32        }
33        else{
34            printf("El término de la sucesión debe ser mayor a 0.\n");
35        }
36    }
37    else{
38        printf("Opción inválida.\n");
39    }
40
41    return 0;
42 }
43
44 /*Funciones*/
45 int calcularPadovan(int termino){
46     int i=0;
47     int padovan,termino2,termino1,termino0;
48
49     do{
50         if(i>=0&&i<=2){
51             termino0=1;
52             termino1=1;
53             termino2=1;
54             padovan=1;
55         }
56         else{
57             padovan=termino0+termino1;
58             termino0=termino1;
59             termino1=termino2;
60             termino2=padovan;
61         }
62         i++;
63     }while(i<termino);
64
65     return padovan;
66 }
67
68 void calcularPerrin(int termino, int *sucesion, int *numeroPerrin){
69     int i=0;
70     int perrin,termino2,termino1,termino0;

```

```

71
72     do{
73         if(i==0){
74             termino0=3;
75             perrin=3;
76         }
77         else if(i==1){
78             termino1=0;
79             perrin=0;
80         }
81         else if(i==2){
82             termino2=2;
83             perrin=2;
84         }
85         else{
86             perrin=termino0+termino1;
87             termino0=termino1;
88             termino1=termino2;
89             termino2=perrin;
90         }
91         i++;
92     }while(i<termino);
93
94     *sucesion = perrin;
95     *numeroPerrin = calcularPadovan(termino+1) + calcularPadovan(termino-10);
96 }
```

4. Raíz cuadrada - Horarios: 0382, 0383

Hay varias formas de hallar la raíz cuadrada de los números. Una de ellas, es el **método de la resta sucesiva**. Este método consiste en tomar el número inicial y restar el primer número impar, es decir, uno. A este resultado se le resta el siguiente número impar y así sucesivamente hasta que el resultado de la resta sea menor o igual a cero. Si el resultado final es igual a cero se trata de un número con raíz entera y estará dada por la cantidad de veces que se hizo la resta, incluyendo el cero. Si el resultado es menor que cero, el número no tiene raíz perfecta y el resultado aproximado (truncado) estará dada por la cantidad de veces que se hizo la resta.

El otro método es el **cuadrado perfecto** es un número entero que es el cuadrado de algún otro; dicho de otro modo, es un número cuya raíz cuadrada es un número natural. Por ejemplo: el número 9 tiene raíz cuadrada entera debido a que se puede expresar como 3×3 .

Se pide desarrollar un programa en Lenguaje C que permita realizar lo siguiente:

- Leer un número.
- Debe validar que el número ingresado sea mayor a 0. En caso contrario, debe emitir el siguiente mensaje: El número ingresado debe ser mayor a 0 y el programa debe terminar.
- Muestre un menú de opciones, las cuales son R: Restas sucesivas. C: Cuadrado perfecto.
- Leer una opción. Debe validar que la opción ingresada sea R o C. Si elige una opción distinta a las indicadas, se mostrará el mensaje: Opción inválida y el programa debe terminar.
- En cualquiera de las opciones, se deberá imprimir si tiene raíz exacta y cuál es la raíz.

La solución deberá contar como mínimo 3 módulos, incluido el módulo principal. Además, uno de los módulos debe devolver exactamente dos valores como parámetros que modifican su valor y no puede usarse para leer datos.

Consideraciones adicionales

Para la solución del problema debe tener en cuenta lo siguiente:

- No puede utilizar la función sqrt de la librería math.h por ningún motivo.
- Considerar como opciones válidas solo la letra R y C en mayúscula.

A continuación se presentan algunos ejemplos de ejecución:

Caso de prueba 1

Ingrese un número: -1

El número ingresado debe ser mayor a 0.

Caso de prueba 2

Ingrese un número: 25

Menú de opciones:

R: Restas sucesivas.

C: Cuadrado perfecto.

Ingrese una opción: F

Opción inválida.

Caso de prueba 3

Ingrese un número: 49

Menú de opciones:

R: Restas sucesivas.

C: Cuadrado perfecto.

Ingrese una opción: r

Opción inválida.

Caso de prueba 4

Ingrese un número: 245

Menú de opciones:

R: Restas sucesivas.

C: Cuadrado perfecto.

Ingrese una opción: R

El número ingresado no tiene raíz exacta.

La raíz cuadrada aproximada es: 16

Caso de prueba 5

Ingrese un número: 625

Menú de opciones:

R: Restas sucesivas.

C: Cuadrado perfecto.

Ingrese una opción: R

El número ingresado tiene raíz exacta.

La raíz cuadrada de 625 es: 25

Caso de prueba 6

Ingrese un número: 181

Menú de opciones:

R: Restas sucesivas.

C: Cuadrado perfecto.

Ingrese una opción: C

El número ingresado no tiene raíz exacta.

La raíz cuadrada aproximada es: 14

Caso de prueba 7

Ingrese un número: 81

Menú de opciones:

R: Restas sucesivas.

C: Cuadrado perfecto.

Ingrese una opción: C

El número ingresado tiene raíz exacta.

La raíz cuadrada de 81 es: 9

Programa 4: Propuesta de solución - Raíz Cuadrada

```
1 #include <stdio.h>
2
3 /*Prototipos*/
4 void calcularRaizCuadrada(int numero, int *cantIteraciones, int *tieneRaiz);
5 void calcularRaizCuadradaIterativa(int numero, int *cantIteraciones, int *tieneRaiz);
6
7 int main(){
8
9     /*Declaración de variables*/
10    int numero, cantIteraciones, tieneRaiz;
11    char opcion;
12
13    printf("Ingrese un número:\n");
14    scanf("%d", &numero);
15
16    if(numero>0){
17        printf("\nMenú de opciones:\n");
18        printf("R: Restas sucesivas.\n");
```

```

19     printf("C: Cuadrado perfecto.\n");
20     printf("\nIngrese una opción: \n");
21     scanf("\n%c",&opcion);
22
23     if(opcion=='R'||opcion=='C'){
24         if(opcion=='R'){
25             calcularRaizCuadrada(numero,&cantIteraciones,&tieneRaiz);
26         }
27         else{
28             calcularRaizCuadradaIterativa(numero,&cantIteraciones,&tieneRaiz);
29         }
30
31         if(tieneRaiz){
32             printf("\nEl número ingresado tiene raíz exacta.\n");
33             printf("La raíz cuadrada de %d es: %d\n",numero,cantIteraciones);
34         }
35         else{
36             printf("\nEl número ingresado no tiene raíz exacta.\n");
37             printf("La raíz cuadrada aproximada es: %d\n",cantIteraciones);
38         }
39     }
40     else{
41         printf("Opción inválida.\n");
42     }
43 }
44
45     printf("El número ingresado debe ser mayor a 0.\n");
46 }
47
48 return 0;
}
49
50
51 /*Funciones*/
52 void calcularRaizCuadrada(int numero, int *cantIteraciones, int *tieneRaiz){
53     int impar=1;
54     int contador=0;
55     *tieneRaiz=0;
56
57     do{
58         if(numero==impar){
59             *tieneRaiz=1;
60         }
61         numero=numero-impar;
62         contador++;
63         impar=impar+2;
64     }while(numero>0);
65
66     *cantIteraciones=contador;
67 }
68
69 void calcularRaizCuadradaIterativa(int numero, int *cantIteraciones, int *tieneRaiz){
70     int i=1,nuevoNum;
71     *tieneRaiz=0;
72
73     do{
74         nuevoNum=i*i;
75         if(numero==nuevoNum){
76             *tieneRaiz=1;
77         }
78         i++;
79     }while(numero>nuevoNum);
80
81     *cantIteraciones=i-1;
82 }

```

Puede usar cualquier estructura selectiva por lo que el uso de la estructura selectiva múltiple queda a su

criterio

No puede usar estructuras iterativas de entrada controlada ni anidadas.