Fundamentos de Programación

Grado en Ing. Informática

Guión práctico nº 2

Tema 3.- Estructuras de Control

Dpto. de Tecnologías de la Información
Profesores Martín Ramos J. M., Polo Almohano M.P., Roche Beltrán F., Suárez Fábrega A.J.

Propuesta problemas tema 3 Alternativas

1. Diseñe un programa que lea una nota con decimales por teclado y muestre por pantalla la calificación que tiene.

NOTA	Mensaje
Menos de 0	Error en la nota
Entre 0 y 5(no incluido)	SUSPENSO
Entre 5 y 7(no incluido)	Aprobado
Entre 7 y 9(no incluido)	Notable
Entre 9 y 10(ambos incluidos)	Sobresaliente
Más de 10	Error en la nota

2. Diseñe un programa que lea dos números enteros por teclado y determine cuál es el mayor y cuál es el menor. También deberá considerar el caso en el que los dos números sean iguales.

3. Diseñe un programa que resuelva una ecuación de segundo grado $ax^2+bx+c = 0$. Debe pedir al usuario que introduzca los valores de a, b y c de tipo entero y el programa indicará las dos posibles raíces solución, o bien si sólo tiene una su valor, o bien un mensaje expresando que no tiene raíces reales.

4. Diseñe un programa que pida dos números enteros por teclado y muestre por pantalla el siguiente menú:

```
MENU

   1. Sumar
   2. Restar
   3. Multiplicar
   4. Dividir

Elija opción:
```

El usuario deberá elegir una opción y el programa deberá mostrar el resultado por pantalla, muestre el mensaje "**Opción inválida**" si el usuario pone una opción incorrecta.

5. Cree una clase llamada **Fecha** con el siguiente atributo:
   agno // Contiene el año indicado por teclado
   y los siguientes métodos:
   bool bisiesto (); /*Devuelve true si el atributo agno tiene un año bisiesto, en caso contrario devuelve false. Un año es bisiesto si es múltiplo de 400, o bien si es múltiplo de 4 y no de 100*/
   void leer(); //Pedirá por teclado el año y lo almacenará en el atributo agno

   Implemente un pequeño main que utilice esta clase.

6. Crear una clase que contenga dos atributos haciendo referencia a valores enteros. Esta clase contará con dos métodos públicos, que serán los siguientes:

   void informacion(); /* Solicita por teclado dos números enteros, y los almacena en los atributos de la clase */
   void operacion(); /* Solicita al usuario la operación a realizar con los dos números introducidos por teclado, de modo que si el usuario pulsa '+', 'S' o 's' el método mostrará un mensaje indicando el resultado de la suma de ambos números; pero si el usuario indica '-', 'R' o 'r' el método mostrará el resultado de la resta. Para cualquier otra pulsación se deberá indicar un mensaje de error, indicando que la operación solicitada no es válida */
   Diseñe un pequeño main que haga uso de esta clase.

7. Crear una clase llamada Billete con los siguientes atributos:
   distancia /* Valor de tipo int que contiene los kilómetros de que consta el viaje */
   dias /* Valor entero que almacena el número de días que dura un viaje*/
   edad /* Valor de tipo entero que guarda la edad del viajero. */

   Esta clase tendrá los siguientes métodos públicos:
   void informacion(); /* Pide por teclado la distancia del viaje a realizar, el número de días que dura dicho viaje y la edad del turista, almacenándolos en los atributos correspondientes*/
   float operacion(); /* Determina el precio de un billete de ida y vuelta en avión, conociendo la distancia a recorrer y sabiendo que si el número de días de estancia es superior a 7 y la distancia superior a 800km, o bien la persona es mayor de 55 años, el billete tiene una reducción del 25%. El precio por kilómetro es de 0.50€ */
   Implementar un pequeño programa que haga uso de esta clase.

8. Modificar el código del método **`void Leer()`** realizado para el ejercicio nº 24 del guión práctico nº 1, de modo que el usuario pueda introducir el valor de la carga en cualquiera de las siguientes unidades:
   a) ***Miliculombios*** ($mC = 10^{-3}C$)
   b) ***Microculombios*** ($\mu C = 10^{-6}C$)
   c) ***Nanoculombios*** ($nC = 10^{-9}C$)

   La distancia también podrá ser suministrada por parte del usuario en:
   a) ***Decímetros*** (*dm*)
   b) ***Centímetros*** (*cm*)
   c) ***Milímetros*** (*mm*)

   En dicho método se hará también la conversión a ***μC*** y a ***m***, necesaria para poder almacenar los valores en los atributos de la clase.

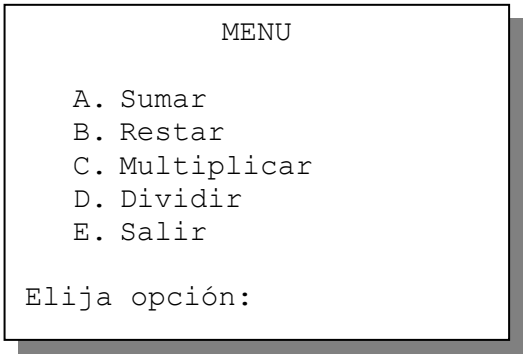
   Cuando el usuario indique el valor numérico de la carga, el programa le ofrecerá mediante un menú las diferentes opciones en cuanto a unidades, para que seleccione la correcta. Del mismo modo se trabajará con la distancia. Si el usuario pulsa una opción incorrecta se tomará por defecto la unidad de almacenamiento (***μC*** y ***m***) y se le avisará al usuario con un mensaje.

   La siguiente tabla puede ser utilizada por el alumno para comprobar los resultados obtenidos:

q = 1x10^{-6}mC	r = 20 mm	E = 2.25x10^{4} N/C
q = 5 nC	r = 30 cm	E = 499.5 N/C
q = 0.04μC	r = 20 dm	E = 89.9 N/C

Propuesta problemas tema 3 Repetitivas

1. Modifique el problema 4 de la relación (Tema 3 Alternativas), de tal forma que las opciones se escojan por letra y se añada una nueva opción E al menú que sea:



   El programa una vez realizada una opción deberá mostrar el menú de nuevo hasta que se elija la opción E. Salir.

2. Cree una clase con los atributos máximo, minimo y media, y con los siguientes métodos públicos:
   void estadistica(); /* Pide por teclado cuantos números enteros va a poner, a continuación pedirá por teclado tantos números como el usuario haya indicado, y almacenará el valor máximo, el mínimo y la media de todos los números introducidos en los atributos correspondientes. */
   int mostrarmax(); /* Devolverá el valor del atributo maximo */
   int mostrarmin(); /* Devolverá el valor del atributo minimo */
   float mostrarmedia(); /* Devolverá el valor del atributo media */

   Implemente un pequeño programa que utilice esta clase.

3. Crear una clase llamada **Estadistica** que contenga los siguientes atributos:
   susp /* Almacena el número de suspensos */
   apr /* Contiene el número de aprobados */
   notab /* Contiene el número de notables */
   sob /* Almacena el número de sobresalientes */

   Además dispondrá de los siguientes métodos públicos:
   void elaborarEstadistica(); /* Pide por teclado cuantos alumnos se han examinado, a continuación solicitará las notas de esos alumnos, que deben estar comprendidas en el rango [0,10], si esto no se cumple se volverá a solicitar esa nota hasta que sea correcta */
   void resultado(); /* Muestra por pantalla el número de suspensos, aprobados, notables y sobresalientes que ha habido en el examen */

   Implementar un pequeño programa que utilice esta clase.

4. Diseñe la siguiente clase para mostrar una tabla de multiplicar:

```
class TablaMultiplicar{
      int Tabla;
public:
      void PedirNoTabla();
      void MostrarTabla();
};
```

donde los métodos realizarán la siguiente función:

- **`PedirNoTabla()`**: Método para solicitar por teclado el número de la tabla de multiplicar a mostrar. Si el valor introducido es menor que uno o mayor que 10, el método volverá a solicitar otro número de tabla.
- **`MostrarTabla()`**: Método que muestra por pantalla la tabla de multiplicar con el formato:

  5 x 0 = 0
  5 x 1 = 5
  5 x 2 = 10
  5 x 3 = 15
  .....
  5 x 10 = 50

Implementar un programa principal que utilice la clase y muestre por pantalla la tabla solicitada.

5. Diseñe la siguiente clase para mostrar varias tablas a la vez de manera horizontal

```
class TablasMultiplicar{
      int TablaIni, TablaFin;
public:
      void PedirNoTablas();
      void MostrarTablas();
};
```

donde los métodos realizarán la siguiente función:

- **`PedirNoTablas()`**: Método para solicitar por teclado el número de la tabla de multiplicar inicial y final a ser mostradas. Si el valor introducido en ambos atributos (**`TablaIni`**, **`TablaFin`**) es menor que uno o mayor que 10, el método volverá a solicitar ambos números de tabla.
  En el caso que el valor de **`TablaIni`** sea menor que **`TablaFin`**, el método deberá intercambiarlos.

- **`MostrarTablas()`**: Método que muestra por pantalla las tabla de multiplicar con el formato:

5 x 0 = 0	6 x 0 = 0	7 x 0 = 0	8 x 0 = 0
5 x 1 = 5	6 x 1 = 6	7 x 1 = 7	8 x 1 = 8
5 x 2 = 10	6 x 2 = 12	7 x 2 = 14	8 x 2 = 16
5 x 3 = 15	6 x 3 = 18	7 x 3 = 21	8 x 3 = 24
.....	.....	.....	.....
5 x 10 = 50	6 x 10 = 60	7 x 10 = 70	8 x 10 = 80

Nota: Utilizar en la sentencia **`cout`** la constante **`\n`** para ir a una nueva línea y **`\t`** para introducir un tabulador.

Implementar un programa principal que utilice la clase y muestre por pantalla las tabla solicitadas.

6. Implemente la siguiente clase:

```
class primo {
   int n;
public:
    void cargar();
//Pedirá por teclado un número entero, almacenándolo en n
    bool esPrimo();
//Devolverá true si es primo el número n, false en caso contrario
};
```

Diseñe un pequeño main que haga uso de esta clase.

7. Modifique la clase del ejercicio anterior añadiéndole el método **`ListaPrimos`** para que muestre por pantalla los primos comprendidos en un intervalo. Este método deberá utilizar el método **`esPrimo`** ya implementado:

```
class primo {
   int n;
public:
    void cargar();
//Pedirá por teclado un número entero, almacenándolo en n
    bool esPrimo();
//Devolverá true si es primo el número n, false en caso contrario
    void ListaPrimos();
};
```

Diseñe un pequeño main que haga uso de esta clase.

8. Cree una clase con un atributo de tipo entero, y con los siguientes métodos públicos:

void informacion(); /* Pide por teclado un número entero que debe ser positivo menor que 20, si esto no se cumple se volverá a solicitar de nuevo hasta que se cumpla, para almacenarlo en el atributo de la clase */

long factorial(); /* devuelve el valor del factorial del número almacenado en el atributo de la clase */

Implemente un pequeño programa que utilice esta clase y muestre por pantalla el factorial del número solicitado por teclado.

9. Cree una clase llamada con un atributo de tipo entero, y con los siguientes métodos públicos:
   void informacion(); /* Pregunta al usuario cuántos números quiere sumar, almacenando dicho número en el atributo correspondiente (llamado N, por ejemplo) */
   int suma(); /* Devolverá la suma de los N primeros números enteros, exceptuando de dicha suma aquellos números que sean primos */

   Implemente un pequeño programa que utilice esta clase.

10. Diseñe la siguiente clase:

```
class Numero{
      int a;
      int b;
public:
      void informacion();
      int mcd();
};
```

- El método **información** pedirá dos valores enteros por teclado almacenándolos en a y b.
- El método **mcd** devolverá el máximo común divisor de a y b, aplicando el algoritmo sencillo de Euclides que dice que si a es mayor que b entonces a = a – b y si b es mayor que a entonces b = b – a y esto se repite hasta que a y b son iguales, momento en el que el máximo común divisor es a

Por ejemplo

a= 6, b=8	Como a< b, entonces b= b-a, b= 8 -6= 2
a= 6, b=2	Como a>b, entonces a= a–b, a= 6 -2= 4
a= 4, b=2	Como a>b, entonces a= a–b, a= 4 -2= 2
a= 2, b=2	Como a== b, entonces el mcd es 2

11. Diseñe la siguiente clase para mostrar una sucesión de Fibonacci

```
class Fibonacci{
      int NoElementos;
public:
      void PedirNoElementos();
      float MostrarElementos();
};
```

donde los métodos realizarán la siguiente función:

- **`PedirNoElementos()`**: Método para solicitar por teclado el número de elementos de la secuencia de fibonacci a ser mostrados. Si el valor introducido es menor que uno, el método volverá a solicitar el número de elementos.
- **`MostrarElementos()`**: Método que muestra los **`NoElementos`** primeros de la secuencia de Fibonacci. Además devolverá la media de todos ellos.

Implementar un programa principal que utilice la clase y muestre la media de los elementos mostrados.

Nota: La sucesión de Fibonacci es aquella cuyos dos primeros elementos son el 0 y el 1 y los siguientes se obtienen como suma de los dos elementos inmediatamente anteriores. por ejemplo:

Orden	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Valor	0	1	1	2	3	5	8	13	21	34	55	89	144	233	377	610

12. Modifique la clase del ejercicio anterior añadiéndole el método **`ComprobarElemento`** que solicita por teclado un número y comprobará si dicho número es un elemento de la secuencia de Fibonacci. En caso afirmativo devolverá la posición que ocupa dentro de la secuencia y en caso contrario -1. Por ejemplo, si introducimos el valor 55 en este método, nos devolverá el valor 10, pero si introducimos el valor 40 nos devolverá -1.

    Añadir al programa principal el código necesario para que muestre por pantalla la posición del valor introducido dentro de la sucesión o bien el error "**El número introducido no pertenece a la sucesión**".

13. Modificar el código realizado para el ejercicio nº 23 del guión práctico nº 1, de modo que desde el programa principal se nos indique si la *Fuerza* es de ***atracción*** (valor negativo para F) o de ***repulsión*** (valor positivo para F).

    Tras mostrar un resultado al usuario, el programa preguntará si se quiere volver a ejecutar con otros datos diferentes, de manera que:

    a) Si la respuesta es afirmativa se le pedirán nuevos valores para las cargas y la distancia entre ellas.
    b) Si la respuesta es negativa se le mostrará un mensaje de agradecimiento por el uso del software.

    La siguiente tabla puede ser utilizada por el alumno para comprobar los resultados obtenidos:

q_1 = 3μC	q_2 = -8μC	r = 2m	F = -0.054 N (Atracción)
q_1 = -0.5μC	q_2 = 644μC	r = 3.5m	F = -0.237 N (Atracción)
q_1 = 2.8μC	q_2 = 7.5μC	r = 0.1374m	F = 10 N (Repulsión)
q_1 = 4μC	q_2 = -8μC	r = 0.004m	F = -18000 N (Atracción)
q_1 = 1μC	q_2 = 2.5μC	r = 0.05m	F = 9 N (Repulsión)

14. Se pretende simular el funcionamiento de un **reloj digital** de manera que comience su funcionamiento a la hora indicada por el usuario (p. ej. las **`12:58:0`**) y realice la cantidad de **minutos de simulación** que se le haya indicado desde teclado (p.ej. 3 minutos de simulación implicarían acabar a las **`13:0:59`**).

    Para ello el alumno deberá implementar una clase **`Reloj`**, con las siguientes características:

    a. Tendrá los **atributos**:
        - **`Hi`** (Hora de inicio).
        - **`Mi`** (Minuto de inicio).
        - **`Si`** (Segundo de inicio).
        - **`TiempoSimulacion`** (Cantidad total de **segundos que deben ser simulados**).

    b. Sus **métodos públicos** serán los siguientes:
        - **`void Iniciar();`**

          Pedirá al usuario la hora y el minuto de inicio de la simulación y pondrá a 0 la cantidad de segundos de inicio.

          A continuación solicitará al usuario la cantidad de tiempo que deberá durar dicha simulación. El valor indicado por el usuario será un valor entero y hará referencia a **minutos de simulación**, habrá pues que tener en cuenta que el atributo **`TiempoSimulación`** lo que almacena son **segundos**.

          El tiempo de simulación indicado por el usuario del programa deberá ser **mayor o igual que cero**; de no ser así se le indicará esta circunstancia al usuario con un mensaje y se le volverá a solicitar otra cantidad de minutos.

        - **`void Simular();`**

          Mostrará el reloj por pantalla cambiando tal y como se ha indicado y realizará el tiempo de simulación solicitado.

          Comenzará con la hora de inicio, **`Hi:Mi:Si`**, y por cada segundo de simulación se volverá a mostrar de nuevo, borrando previamente el anterior.

          Para simular un tiempo de espera de un segundo, utilizaremos la función **`Sleep(milisegundos)`** que se encuentra en la librería **`windows.h`**. Se aconseja al alumno que para que la simulación no sea ni muy rápida ni excesivamente lenta, utilice la función Sleep con 500 milisegundos, **`Sleep(500);`** de manera que para la simulación 1 segundo serán realmente 500 milisegundos.

    Implementar un programa principal que utilice la clase **`Reloj`**.

15. Se quiere codificar un juego consistente en adivinar un número. Para ello se pide implementar la clase **`Adivinar`**, que constará de:

    a. Los siguientes **atributos privados**:
       - **`Nsecreto`** (número secreto a adivinar).
       - **`Puntos`** (puntos de partida para comenzar a jugar).
    b. Los siguientes **métodos públicos**:
       - **`void Inicio();`**

         Se establecen los parámetros iniciales del juego, es decir se determinará cuál será el número secreto y con qué cantidad de puntos comienza el jugador la partida.

         Para ello se solicita al jugador que indique un número. Dicho número será la ***semilla*** a partir de la cual se obtendrá el número secreto y los pasos a dar serán los siguientes:

         1) Si el valor obtenido de la siguiente operación **`(semilla%90)x123`** es un ***número abundante***, es decir el valor obtenido es menor que la suma de todos sus divisores, entonces el **número secreto será** el que se obtenga tras la ejecución de la siguiente expresión (**`(semilla%90)x123)%12459.`**
         2) En caso contrario (el ***número no*** es ***abundante***) el número secreto será (**`(semilla%90)x123)%5397.`**

         Además se preguntará cuál será la cantidad de puntos de partida (p.ej. 10).

       - **`void Jugar();`**

         Se implementará el juego, consistente en pedir al jugador que intente adivinar el número secreto.

         Si el número indicado no es el secreto y se ha quedado corto se le quita 1 punto, pero si se pasa se le quitan 2.

         El juego finalizará cuando se adivina el número o cuando el jugador se queda sin puntos. En cualquier caso se le mostrará mensaje que indique lo ocurrido, que será de enhorabuena con los puntos conseguidos, o lamentando que haya perdido.

    Además habrá que implementar un programa que utilice dicha clase para jugar tantas veces como se quiera.

16. Juego "**Te adivino el número que estás pensando**". En este caso se trata de programar un juego para que sea el ordenador el que te haga preguntas para intentar adivinar el número que has pensado previamente. Para ello se implementará los métodos de la clase Juego y un programa main que utilizará dichos métodos de la siguiente forma:

```
class Juego
{
    int ValorMin,ValorMax, //forman el rango [ValorMin,ValorMax]
     ValorMedio;

public:
    void PedirExtremos();
    char AdivinarNumero();
    bool CompruebaEsMenor();
    bool CompruebaEsMayor();
};
```

- **`PedirExtremos()`**: Método que solicitará por teclado los extremos [a,b] de un rango en el que el usuario deben pensar un número. Siempre el extremo a debe ser menor que b. Si no cumple esa opción mostrará por pantalla un error y volverá a preguntar dichos extremos a y b. En este método a y son **`ValorMin`** y **`ValorMax`** respectivamente.
- **`AdivinarNumero()`**: Método que calcula la media aritmética de los valores extremos del intervalo. Muestra por pantalla dicha media y pregunta por teclado si el valor pensado por el jugador es menor (**<**) mayor (**>**) o Igual (**=**) al propuesto. Si la entrada es correcta (**<**, **>** ó **=**) la devuelve , si no lo es mostrará un error y volverá a realizar la misma pregunta.
- **`CompruebaEsMenor()`**: Método que modificará el **`ValorMax`** con el **`ValorMedio`** menos 1. Además comprueba que los extremos verifican que el **`ValorMin`** es menor que **`ValorMax`**. Si no lo son devolverá false en caso contrario true.
- **`CompruebaEsMayor()`**: Método que modificará el **`ValorMin`** con el **`ValorMedio`** mas 1. Además comprueba que los extremos verifican que el **`ValorMin`** es menor que **`ValorMax`**. Si no lo son devolverá false en caso contrario true.

Implementar en el main el código de un Juego que utilice los métodos de la clase de la forma:

a) Pida los extremos.
b) Adivine un numero y pregunte al jugador si es <, > o =
c) Si la opción seleccionada es < y la comprobación de si es menor es falsa, muestra un error de que el jugador intenta engañar al programa y termina el juego. Si la comprobación es true no hace nada.
d) Si la opción seleccionada es > y la comprobación de si es mayor es falsa, muestra un error de que el jugador intenta engañar al programa y termina el juego. Si la comprobación es true no hace nada.
e) Si la opción seleccionado es = muestra un mensaje diciendo que lo ha conseguido y termina.

Repita los desde el punto b) al e) hasta que se seleccione un = o haya un error en las comprobaciones de si es mayor o menor.

17. Implementar un programa que simule un juego de acertar un número de tres cifras. El programa generará un número al azar y el usuario deberá intentar acertarlo con un máximo de 10 intentos. En cada intento el programa indicará si el número introducido por el usuario es mayor, menor o ha acertado.

    La clase a implementar es:

```
class juego{
    int numero;     // número aleatorio creado por el usuario
    int n_intento; // números de intentos realizados por el usuario
    int intento;   /* número introducido por el usuario para
                      intentar ganar */

    int pedir_numero();
public:
    void generar_numero();
    void jugada();
};
```

    donde los métodos realizarán la siguiente función:

    - El método **`pedir_numero()`** será privado y lo que hará será pedir un número al usuario dicho numero se utilizará en el método **`jugada()`**.
    - El método **`generar_numero()`** generará un número aleatorio entre 0 y 999, será el número a acertar, para ello utilizar el siguiente código:

```
//Inicializar los números aleatorios
srand(time(NULL));
// Número aleatorio entre 1 y 999
numero=1+rand()%(999-1);
```

    - El método **`jugada()`** pedirá un número al usuario mediante el método **`pedir_numero()`**, comprobará si dicho número es el que se generó aleatoriamente en el método **`generar_numero()`**, si es así se ha ganado el juego, si no es igual nos dirá si el número es menor o mayor que el generado aleatoriamente, esto se repetirá hasta que se acierte el número y se gane o hasta que se realicen 10 intentos.

    Implementar un programa principal que utilice la clase juego.

18. Realizar un programa que calcule el modulo de Young mediante la siguiente clase:

```
clase young{
    float seccion;
    float longitud;
    float carga;
    float deformacion;
public:
    void leer_datos();
    float calcular_esfuerzo();
    float calcular_deformacion();
    float modulo_young();
};
```

El modulo de Young se define como E = s/e, por lo que será necesario conocer el valor del esfuerzo actuante (s) y la deformación del material (e). Estos se calculan como:

Esfuerzo actuante:

$$s = \frac{F}{A_0} = + \frac{F(Newtons)}{\Pi\left(\frac{Sección(mm)}{2}\right)^2}$$

Deformación del material:

$$e = \frac{\Delta l}{l_0}$$

donde los métodos realizarán la siguiente función:

- El método **`leer_datos`** será el encargado de la introducción de los datos de la carga, sección, longitud inicial y el incremento de la longitud. Además cada vez que se introduzca una dato, el método deberá preguntar si el valor introducido ha sido en **Newtons** o **KiloNewtons** para la carga, y en metros o **milímetros** para las longitudes y sección.

  Hay que tener en cuenta que los cálculos se deben hacer en **Newtons** y en **milímetros**, por lo que es posible tener que hacer una conversión de unidades. El resultado del módulo de Young hay que darlo en MegaPascales.

- El método **`calcular_esfuerzo`** es el encargado de calcular el esfuerzo.
- el método **`calcular_deformacion`** es el encargado de calcular la deformación.
- El método **`modulo_young`** es el encargado de calcular y devolver el módulo de Young.

Implementar un programa principal que utilice la clase young y muestre el valor del módulo calculado.

Veámoslo con un ejemplo.

Suponga un objeto metálico de sección circular de 10 mm de diámetro y 60 mm de longitud que, al aplicarle un carga de 50 kN, se alarga elásticamente 0.62 mm. Calcule el módulo de Young, que se utiliza para saber el tipo de material que se ha utilizado en la construcción.

El módulo de Young se define como E = s/e, por lo que será necesario conocer el valor del esfuerzo actuante (s) y el índice de deformación del material (e). Estos se calculan como:

$$s = \frac{F}{A_0} = \frac{50000N}{\Pi\left(\frac{10mm}{2}\right)^2} = 636.62MPa \qquad e = \frac{\Delta l}{l_0} = \frac{(0.62mm)}{(60mm)} = 0.0103$$

Por tanto:

$$E = \frac{(636.62MPa)}{(0.0103)} = 61807.77MPa = 61.8GPa$$