GUÍA DE TRABAJOS PRACTICOS

PROGRAMACIÓN II
2020

MARÍA ALEJANDRA BOSIO

ucc | FACULTAD DE INGENIERÍA

Contenido

I. ENCUADRAMIENTO DE LA FORMACIÓN PRÁCTICA 3
II. CONSIGNAS GENERALES 3
III. TRABAJO PRACTICO Nº 1 - Programación en C++ 4
A. Objetivos de Aprendizaje 4
B. Unidad temática que incluye este trabajo práctico 4
C. Consignas a desarrollar en el trabajo práctico: 4
IV. TRABAJO PRÁCTICO Nº 2 - Punteros 6
A. Objetivos de Aprendizaje 6
B. Unidad temática que incluye este trabajo práctico 6
C. Consignas a desarrollar en el trabajo práctico: 6
V. TRABAJO PRÁCTICO Nº 3 - Programación Orientada a objetos 8
A. Objetivos de Aprendizaje 8
B. Unidad temática que incluye este trabajo práctico 8
C. Consignas a desarrollar en el trabajo práctico: 8
VI. TRABAJO PRÁCTICO Nº 4 - Herencia y polimorfismo 11
A. Objetivos de Aprendizaje 11
B. Unidad temática que incluye este trabajo práctico 11
C. Consignas a desarrollar en el trabajo práctico: 11
VII. TRABAJO PRÁCTICO Nº 5 - Clases parametrizadas y Manejo de Excepciones 15
A. Objetivos de Aprendizaje 15
B. Unidad temática que incluye este trabajo práctico 15
C. Consignas a desarrollar en el trabajo práctico: 15

I. ENCUADRAMIENTO DE LA FORMACIÓN PRÁCTICA

La formación práctica que se desarrolla en esta GTP incluye la resolución de problemas tipo o rutinarios y de problemas abiertos de Ingeniería.

II. CONSIGNAS GENERALES

- Los problemas propuestos en la presente GTP pueden ser resueltos en forma individual o grupal.
- La resolución de los problemas se realizará' en lenguaje C++.

III. TRABAJO PRACTICO Nº 1 - Programación en C++

A. Objetivos de Aprendizaje

a) Repasar los conceptos de programación ya adquiridos en asignaturas anteriores.
b) Adquirir habilidad en la resolución de problemas y optimización del código.

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la Unidad 2 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico:

1. Un grillo se encuentra de un lado de la calle y necesita llegar hasta el otro lado.
   Suponiendo que se encuentra actualmente en la posición X y que quiere llegar a la posición Y tal que, Y es mayor o igual que X, determine la cantidad mínima de saltos que el grillo debe realizar para alcanzar su objetivo. Considere que el grillo siempre salta una distancia fija, D.
   El programa debe solicitar el ingreso de tres enteros X, Y y D y mediante una función calcular y posteriormente mostrar el mínimo número de saltos necesarios.

2. Dados tres valores enteros A, B y K, determine, mediante una función la cantidad de números divisibles por K que se encuentran en el entorno A, B,

3. Obtener una función recursiva que calcule el Máximo Común Divisor de dos números M y N.

4. Obtener una función recursiva que calcule el factorial de un numero N.

5. Un número primo es un número entero positivo X que tiene solamente dos divisores distintos: 1 y X. Los primeros números enteros primos son 2, 3, 5, 7, 11 y 13.
   Un numero primo D se llama *divisor primo* de un entero positivo P si existe un entero positivo K tal que D * K = P. Por ejemplo, 2 y 5 son divisores primos de 20.
   Suponiendo que se dan dos enteros positivos N y M, obtener una función de retorno boolean que permita verificar N y M tienen el mismo conjunto de divisores primos.
   Por ejemplo:
   N = 15 y M = 75, los divisores primos son los mismos: {3, 5}.
   N = 10 y M = 30, los divisores primos no son los mismos: {2, 5} no es igual a {2, 3, 5}.

6. Un *semi-primo* es un número natural que es el producto de dos números primos no necesariamente distintos. Por ejemplo, los números 4, 6, 9, 10, 14, 15, 21, 22, 25, 26, son semi-primos. Dado un numero entero X determinar mediante una función de retorno boolean si se trata de un número semi-primo.

7. Escribir una función que intercambie el valor de dos variables, es decir si X=5 e Y=7 tras aplicar la función, por ejemplo haciendo "intercambiar(X,Y)" se tiene que X=7 e Y=5.

8. Representar un polinomio completo de grado n mediante un arreglo. Ingresar el grado del polinomio y sus coeficientes. A continuación, ingresar un valor x y calcular el polinomio evaluado en ese valor. Utilizar funciones en la resolución.

9. Una permutación es una secuencia que contiene cada elemento de 1 a N una vez, y solo una vez. Suponiendo que se ingresa un arreglo unidimensional de tamaño N, determinar mediante una función si dicho arreglo representa o no una permutación.
   El programa debe solicitar el tamaño del arreglo, a continuación, sus elementos e invocar una función de retorno boolean que devuelva true en caso de tratarse de una permutación o false en caso contrario.

10. Leer la cantidad de elementos de un vector y los datos en él. A continuación, reorganizar el arreglo en modo que los valores impares queden en el primer tramo del arreglo y los pares detrás.

11. Leer la cantidad de elementos de un vector y los datos en él. Determinar mediante una función la cantidad de valores distintos presentes en el arreglo.

12. Leer los n elementos de un vector y ordenarlo de mayor a menor.

13. Dado un arreglo y un número k determinar si el número está presente dentro del arreglo, en ese caso dar su ubicación.

14. Suponiendo un arreglo unidimensional de N, se dice que existe un triplete triangular (P, Q, R) si 0 ≤ P <Q <R <N y:

    A [P] + A [Q]> A [R],
    A [Q] + A [R]> A [P],
    A [R] + A [P]> A [Q]

    Obtener un programa que permita ingresar la cantidad de elementos de un vector y los datos en él. Definir una función de retorno boolean que devuelva true si existe un triplete triangular para el arreglo y false en caso contrario

15. Implementar un programa que solicite la introducción de un vector de N elementos, invierta su contenido (intercambie el primer elemento con el último, el segundo con el penúltimo, etc.) y lo muestre.

16. Implementar una función que, dada una matriz m y una posición (i,j) de dicha matriz, calcule la suma de elementos adyacentes al elemento m(i,j)

IV. TRABAJO PRÁCTICO Nº 2 - Punteros

A. Objetivos de Aprendizaje

a) Comprender el significado de los punteros y su relación con arreglos y cadenas.
b) Desarrollar programas utilizando gestión dinámica de memoria.

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad 3 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico:

1. Escribir un programa que declare e inicialice dos variables enteras y un puntero a entero. A continuación, asigne al puntero la dirección de la primera variable e imprima la dirección de memoria apuntada y su contenido. Repetir la operación para la segunda variable.

2. Escribir un programa que declare un arreglo de 5 enteros, y un puntero a entero. Comprobar que los elementos del arreglo ocupan posiciones sucesivas en memoria, escribiendo sus direcciones.

3. Para el arreglo del ejercicio anterior, declare dos punteros a entero y asigneles las direcciones del primer y último elemento del arreglo. Imprima la diferencia entre ambos punteros.

4. Realizar un programa que rellene de forma aleatoria con los primeros 100 números un arreglo de 15 elementos. Mostrar por medio de punteros los valores en el vector y la dirección de memoria de cada uno.

5. Ingresar una cadena de caracteres de longitud máxima 10, enviarla como parámetro a una función que maneje punteros de modo que retorne la misma cadena de caracteres, pero en mayúsculas.

6. Realizar un programa que ingrese una cadena de caracteres de máximo 50 elementos y la envíe como parámetro a una función que maneje punteros de modo que la función invierta la cadena.

7. Obtener una función copia que utilice punteros para copiar una cadena en otra.

8. Realice un programa que permita ingresar una cadena de caracteres de máximo 50 elementos, la envíe como parámetro a una función que utilizando punteros y retorne el número de vocales minúsculas que contiene la cadena.

9. Definir un arreglo unidimensional de N componentes enteras de modo que la gestión de memoria sea dinámica utilizando new. Ingresar los datos en el vector. Mostrar el contenido del vector en dos modos: usando notación vectorial y luego usando aritmética de punteros.

10. Se pide crear un programa que haciendo uso de la reserva dinámica de memoria almacene un número determinado de valores (n) obtenidos de forma aleatoria, entre 0 y 100 y los ordene de mayor a menor

11. Se pide crear un programa que pida una serie de números al usuario y halle el máximo, el mínimo y la media aritmética de ellos. Para ello se debe crear una

variable puntero tipo float, pedir al usuario que introduzca el número de datos, y sucesivamente los datos a cargar en el arreglo. Recordar que se debe reservar memoria de forma dinámica.

12. Obtener un programa que permita reservar memoria dinámica para un arreglo bidimensional de datos reales realizando los siguientes pasos:
    a. Crear un puntero a punteros del tipo de datos correspondiente.
    b. Reservar memoria para el arreglo de filas.
    c. Hacer un bucle para reservar memoria para las columnas de cada fila:
    d. Ingresar los datos y mostrar su contenido usando índices y el operador de indexación [].
    e. Liberar la memoria asignada.

V. TRABAJO PRÁCTICO Nº 3 - Programación Orientada a objetos

A. Objetivos de Aprendizaje

a) Comprender el concepto de Orientación a Objetos y tipos abstractos.
b) Construir programas en C++ utilizando definiciones de clases.

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a las unidades 4 y 5 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico:

1. Realizar una clase que permita representar una fecha.
   - Definir los datos miembros de la clase.
   - Definir si fuera necesario funciones de carga y muestra de los datos miembro.
   - Definir un constructor que inicializa la fecha a una fecha dada.
   - Definir un constructor que inicializa la fecha en 01/01/1900.
   - Definir sobrecargas de los siguientes operadores:
     - ++ y – incrementa o decrementa la fecha en 1 día.
     - + y – suma a una fecha un cierto número de días.
   - Realizar un programa principal que haga uso de la clase.

2. Realizar una clase que permita representar un número racional.
   - Definir los datos miembros de la clase.
   - Definir si fuera necesario funciones de carga y muestra de los datos miembro.
   - Definir uno o más constructores para inicializar los datos miembro.
   - Definir sobrecargas de los siguientes operadores:
     - + suma 2 racionales generando otro como resultado.
     - – resta 2 racionales generando otro como resultado.
     - ++ incrementa en 1 un racional
     - –– decrementa en 1 un racional.
     - +=suma un entero a un racional.
     - –=suma un entero a un racional
   - Definir además una función miembro que implemente la simplificación del racional.
   - Realizar un programa principal que haga uso de la clase.

3. Realizar una clase que permita representar un número complejo.
   - Definir los datos miembros de la clase.
   - Definir si fuera necesario funciones de carga y muestra de los datos miembro.
   - Definir uno o más constructores para inicializar los datos miembro.
   - Definir la sobrecarga de los operadores +, - y * para efectuar estas operaciones entre complejos.
   - Definir una función miembro para calcular el conjugado de un complejo.
   - Realizar un programa principal que haga uso de la clase.

4. Realizar una clase que permita representar un cronómetro. Dicho cronómetro consta de horas, minutos y segundos. Las horas no tienen límite en valor mientras que los minutos y segundos llegan al máximo hasta 59.
   - Definir los datos miembros de la clase.
   - Definir si fuera necesario funciones de carga y muestra de los datos miembro.

- Definir un constructor que inicializa a cero el cronómetro.
- Definir un método Reset que permita llevar a cero el cronómetro.
- Definir la sobrecarga del operador ++ para producir el incremento del cronómetro en un segundo.
- Realizar un programa principal que haga uso de la clase.

5. Realizar una clase que permita representar un punto en coordenadas cartesianas.
   - Definir los datos miembros de la clase.
   - Definir si fuera necesario funciones de carga y muestra de los datos miembro.
   - Definir un constructor que inicializa el punto en el origen de coordenadas.
   - Definir la sobrecarga del operador ++ que incrementa en 1 ambas coordenadas.
   - Idem para el operador --.
   - Definir 2 sobrecargas para el operados +, una para sumar 2 puntos entre si y otra para sumar a un punto un valor entero.
   - Idem para el operador -.
   - Definir la/las función/es miembro para obtener la conversión en coordenadas polares.
   - Realizar un programa principal que haga uso de la clase.

6. Crear una clase llamada Password que siga las siguientes condiciones:
   - Los atributos son longitud y contraseña. Por defecto, la longitud será de 8 caracteres

   Generar los siguientes constructores:
   - Un constructor por defecto.
   - Un constructor con la longitud pasada como parámetro.

   Generar los siguientes métodos:
   - generarPassword(): genera la contraseña del objeto con la longitud que corresponda.
   - esFuerte(): devuelve si es fuerte o no. Para que sea fuerte debe tener más de 2 mayúsculas, más de 1 minúscula y más de 5 números.
   - Método para visualización de la contraseña y la longitud.
   - Método para cargar la longitud.

   Realizar un programa principal que haga uso de la clase.

7. Obtener una clase que represente una cuenta bancaria. Para la misma se tiene como información:
   - Número de cuenta.
   - Saldo.
   - Tasa de interés

   Definir los datos miembro de la clase.
   Obtener los métodos para:
   - Crear una cuenta nueva (constructor) a partir de la asignación del número de cuenta y una tasa de interés. Toda nueva cuenta se crea con saldo 0.
   - Carga y visualización de los datos miembro.
   - Realizar un depósito.
   - Realizar una extracción.
   - Acreditar intereses.

   Considerar que algunas operaciones requieren comprobación antes de ser realizadas
   Definir un programa principal que gestione dos cuentas distintas.

8. Generar una clase Persona que siga las siguientes condiciones:
   Los atributos son:
   - Nombre
   - Fecha de nacimiento
   - DNI
   - Sexo (H hombre, M mujer)
   - Peso
   - Altura

   Generar los siguientes constructores:
   - Un constructor por defecto.
   - Un constructor con el nombre, edad y sexo, el resto por defecto.
   - Un constructor con todos los atributos como parámetro.

   Definir los métodos para:
   - Carga y muestra de los datos miembro.
   - calcularIMC: calcula el índice de masa corporal como (peso en kg/(altura^2 en m)),
   - calcularEdad: calcula la edad según la fecha de nacimiento.
   - esMayorDeEdad: indica si es mayor de edad.

   Realizar un programa principal que haga uso de la clase.

9. Construir una clase que permita representar un arreglo unidimensional de enteros. Obtener las siguientes sobrecargas:
   - Operador + para sumar dos arreglos.
   - Operador – para restar dos arreglos.
   - Operador * para calcular el producto escalar de 2 arreglos.
   - Operador [] para subindicar los elementos del arreglo.

   Realizar un programa principal que haga uso de la clase.

10. Construir la clase Matriz que permita representar un arreglo bidimensional de enteros. Obtener las siguientes sobrecargas:
    - Operador + para sumar dos Matrices.
    - Operador – para restar dos Matrices.
    - Operador * para calcular el producto de 2 Matrices.
    - Operador () para subindicar cada elemento de la matriz.
    - Operador [] para subindicar los elementos de la diagonal principal

    Realizar un programa principal que haga uso de la clase.

11. Crear una clase conjunto que represente un conjunto de números enteros. El conjunto tiene un tamaño máximo y un cierto número de elementos. Considerar que en un conjunto no puede haber elementos repetidos. Definir los constructores que considere necesario, destructor y sobrecarga del operador de asignación.
    Definir los siguientes métodos y sobrecargas:
    - Agregar: inserta un elemento al conjunto.
    - Eliminar: elimina un elemento del conjunto.
    - Mostrar: Imprimir los elementos del conjunto.
    - Sobrecargar el operador + para realizar la Union de conjuntos.
    - Sobrecargar el operador * para realizar la intersección de conjuntos
    - Sobrecargar el operador - para realizar la diferencia de conjuntos.

VI. TRABAJO PRÁCTICO Nº 4 - Herencia y polimorfismo

A. Objetivos de Aprendizaje

a) Comprender el concepto de Herencia y polimorfismo.
b) Construir programas en C++ según el modelo de generalización-especialización.
c) Comprender el concepto de agregación.
d) Construir programas en C++ según el modelo de agregación y composición.

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a la unidad 6 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico:

1. Se desea obtener un sistema de representación de figuras geométricas. Toda figura geométrica posee un centro (coordenada x y coordenada y) y posee además una fórmula para calcular su perímetro y su área.
   Es posible definir también la razón superficie/perímetro.
   Teniendo en cuenta estas consideraciones obtenga un mecanismo de herencia que a partir de una clase abstracta FIG_GEO permita generar las clases derivadas para representar un punto, un círculo y un polígono regular.
   Realice pruebas de las clases creadas mediante un programa principal.

2. Obtener un programa para gestionar los empleados de un comercio.
   Los empleados se definen por:
   - Nombre
   - Edad
   - Salario

   Existen dos tipos de empleados: repartidor y comercial.
   El comercial, aparte de los atributos anteriores, tiene uno más llamado comisión (número real).
   El repartidor, aparte de los atributos de empleado, tiene otro llamado zona (número entero).
   Se debe generar un mecanismo de Herencia con Empleado (clase padre abstracta) y Comercial y Repartidor (clases hijas).
   Crear los constructores, métodos para cargar y visualizar atributos y un método llamado toString para mostrar la información completa del individuo.
   Definir también un método llamado plus, que aumenta el salario del empleado en 3000$ según las siguientes condiciones:
   - En comercial, si tiene más de 30 años y cobra una comisión de más del 12% se le aplicara el plus.
   - En repartidor, si tiene menos de 25 años y reparte en la zona 3

   Realizar un programa principal que haga uso de las clases.

3. Generar un mecanismo de herencia para gestionar una serie de productos.
   Los productos tienen los siguientes atributos:
   - Nombre
   - Precio

   Hay dos tipos de productos:
   - Perecedero: tiene un atributo llamado días a caducar
   - No perecedero: tiene un atributo llamado tipo

Definir los constructores necesarios y los métodos de carga y visualización de los atributos
Definir el método calcularMonto con un parámetro entero que corresponde a la cantidad de productos:

- En Producto, se calcula como el precio por la cantidad.
- En Perecedero, además de lo anterior, el precio se reducirá según los días a caducar:
  - Si le queda 1 día para caducar, se reducirá 4 veces el precio final.
  - Si le quedan 2 días para caducar, se reducirá 3 veces el precio final.
  - Si le quedan 3 días para caducar, se reducirá a la mitad de su precio final.
- En NoPerecedero, el cálculo es igual que en Producto

Realizar un programa principal que haga uso de las clases.

4. Se desea realizar una aplicación para gestionar la información sobre las personas vinculadas a una facultad.
   Las personas se pueden clasificar en tres tipos:
   - Estudiantes
   - Profesores
   - Personal de servicio.

   A continuación, se detalla qué tipo de información debe gestionarse:
   Por cada persona, se tiene:
   - Nombre y Apellido
   - DNI
   - Estado civil.

   Con respecto a los empleados, sin importar de que tipo que sean, se tiene el año de incorporación a la facultad y el número de oficina asignado.
   Para los estudiantes, se requiere almacenar el curso en el que están matriculados.
   Para los profesores, es necesario gestionar a qué departamento pertenecen (lenguajes, matemáticas, arquitectura, etc.).
   Para el personal de servicio, se debe registrar en que sección están asignados (biblioteca, decanato, secretaría, etc.)
   Se pide:
   Definir la jerarquía de clases.
   Generar las clases definidas en las que, además de los constructores, hay que desarrollar los métodos para:
   - Cambio del estado civil de una persona.
   - Reasignación de despacho a un empleado.
   - Matriculación de un estudiante en un nuevo curso.
   - Cambio de departamento de un profesor.
   - Traslado de sección de un empleado del personal de servicio.
   - Imprimir toda la información de cada tipo de individuo.

   Generar un programa de prueba que instancie objetos de los distintos tipos y pruebe los métodos desarrollados.

5. Para un juego de rol, es necesario definir el esquema de personajes según lo siguiente:
   - Hay 2 tipos de personajes, los Magos y los Clérigos.
   - Todos los personajes cuentan con los siguientes datos:
   - Nombre: una cadena.
   - Raza: una cadena que puede tomar los valores “humano”, “elfo”, “enano” u “orco”.

- Fuerza: un entero entre 0 y 20
- Inteligencia: un entero entre 0 y 20
- Puntos de vida máximos: un entero entre 0 y 100
- Puntos de vida actuales: un entero entre 0 y puntos de vida máximos

Además cada tipo de personaje tiene atributos y restricciones específicos que se Detallaran más adelante.
Se pide:

- Generar una clase Personaje que reúna los atributos mencionados anteriormente. Dicha clase debe incluir:
    - Un constructor para poder inicializar los atributos (se supone que los puntos de vida actuales son iguales a los máximos al inicializarse)
    - Métodos de carga y visualización para todos los atributos de la clase
    - Método imprime que muestre por pantalla los datos del personaje

- Generar la clase Mago teniendo en cuenta las siguientes restricciones:
    - Al crearse, un mago no puede tener en inteligencia un valor menor que 17 ni en fuerza un valor mayor que 15.
    - Además un mago almacena los nombres de los hechizos que ha memorizado y sólo puede memorizar a la vez un máximo de 4 hechizos.
    - Generar los siguientes métodos:
        - AprendeHechizo: que tiene un parámetro de tipo cadena y añade un hechizo a los memorizados.
        - LanzaHechizo: que tiene como parámetro un objeto de tipo Personaje. Este será el personaje sobre el que recae el hechizo. Al lanzar el hechizo, el Personaje que lo recibe pierde 10 de los puntos de vida y el Mago lo olvida.
        - Generar el constructor de la clase sabiendo que en el momento de su creación, el Mago no conoce ningún hechizo.
        - Adecuar el método imprime para que se muestren además de los datos del personaje la lista de hechizos.
- Generar la clase Clérigo teniendo en cuenta las siguientes restricciones:
    - Al crearse, un clérigo no puede tener una fuerza con un valor menor que 18 y una inteligencia con un valor menor que 12 ni mayor que 16 ambos incluidos
    - El clérigo reza a un dios para obtener el don de la curación. Por lo tanto el constructor debe aceptar el nombre del dios del cual el clérigo es devoto y almacenarlo.
    - Un clérigo tiene el don de curar.
    - Generar los siguientes métodos:
        - Curar que recibe como parámetro un objeto de tipo Personaje sobre el que recae la cura. La vida de ese personaje aumenta en 10 puntos.
        - Adecuar el método imprime para que se muestren además de los datos del personaje se muestre el nombre del Dios.

- Generar un programa de prueba en el que se creen 2 magos (A y B) y un clérigo (C) y el que tengan que realizar las siguientes acciones:
- Imprimir los datos de los tres personajes
- El mago A aprende 2 hechizos.
- El mago B aprende 1 hechizo.
- Imprimir los datos de los magos
- El mago A lanza un hechizo sobre el mago B

- El mago B lanza un hechizo sobre el mago A
- El clérigo cura al mago B
- El mago A lanza un hechizo sobre el mago B
- Imprimir los datos de los tres personajes

VII. TRABAJO PRÁCTICO Nº 5 - Clases parametrizadas y Manejo de Excepciones

A. Objetivos de Aprendizaje

a) Comprender el concepto de clases parametrizadas y su utilidad en reutilización de código.
b) Construir programas en C++ con uso de templates.
c) Construir programas con manejo de excepciones.

B. Unidad temática que incluye este trabajo práctico

Este trabajo práctico corresponde a las unidades 7 y 8 de la programación de la asignatura.

C. Consignas a desarrollar en el trabajo práctico:

1. Repetir el ejercicio 9 del práctico 3 de modo de tener una clase parametrizada. Utilizar la clase usando como tipo de dato entero, float y donde sea posible los tipos creados en el practico 3.

2. Repetir el ejercicio 10 del práctico 3 de modo de tener una clase parametrizada. Utilizar la clase usando como tipo de dato entero, float y donde sea posible los tipos creados en el practico 3.

3. Obtener una clase que permita trabajar sobre un archivo de texto. La misma tendrá un dato miembro inicializado por el constructor que corresponde al nombre completo del archivo de texto a trabajar (path, nombre del archivo y extensión).
   Determinar los datos y funciones miembro necesarios para proveer a la clase de las siguientes funcionalidades:
   - Lectura del texto.
   - Conversión a mayúsculas del texto.
   - Calculo de la cantidad de caracteres del texto.
   - Calculo de la cantidad de palabras del texto. (los espacios repetidos no deben ser considerados como palabras distintas).
   - Calculo de la longitud promedio de las palabras del texto.

4. Obtener programas que haciendo uso de las excepciones (try cath) prevengan los siguientes errores de ejecución:
   - Memoria insuficiente al intentar la operación de new.
   - División por un denominador nulo.