GUÍA DE TRABAJOS PRACTICOS

PROGRAMACIÓN II
2019

MARÍA ALEJANDRA BOSIO

ucc | FACULTAD DE INGENIERÍA

ENCUADRAMIENTO DE LA FORMACIÓN PRÁCTICA	3
CONSIGNAS GENERALES	3
INFORME DE TRABAJO PRÁCTICO	3
TRABAJO PRACTICO Nº 1 - Programación en C++. Funciones. Arreglos	4
TRABAJO PRÁCTICO Nº 2 - Punteros	8
TRABAJO PRÁCTICO Nº 3 - Programación Orientada a objetos	10
TRABAJO PRÁCTICO Nº 4 - Herencia y polimorfismo	14
TRABAJO PRÁCTICO Nº 5 - Clases parametrizadas y Manejo de Excepciones	18

1. 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.

2. CONSIGNAS GENERALES

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

3. INFORME DE TRABAJO PRÁCTICO

- Deberá presentarse la carpeta de trabajos prácticos con la resolución de los ejercicios que indique el docente

TRABAJO PRACTICO Nº 1 - Programación en C++

1. 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.

2. Unidad temática que incluye este trabajo práctico

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

3. Consignas a desarrollar en el trabajo práctico:

1. Dada una medida de tiempo expresada en horas, minutos y segundos con valores arbitrarios, obtenga un programa que transforme dicha medida en una expresión correcta. Por ejemplo, dada la medida 3h 118m 195s, el programa deberá obtener como resultado 5h 1m 15s.

2. Validar el día, mes y el año para una fecha suministrada sabiendo que los años van desde 1960 hasta 2021.

3. Obtener un programa que lea un número natural y diga si es o no es triangular.
   A saber: un número N es triangular si, y solamente si, es la suma de los primeros M números naturales, para algún valor de M.
   Ejemplo: 6 es triangular pues 6 = 1 + 2 + 3.

4. Desarrollar un programa en el cual introduzca un número entero positivo e invierta los dígitos del número. Mostrar en pantalla el número invertido.

5. Solicitar al usuario un valor entero y decir si es capicúa.

6. Escribir un programa que sume las cifras de un número entero positivo.

7. Realizar un programa que simule un juego de dados con las siguientes reglas:
   - El jugador tira dos dados. Cada dato tiene seis caras. Las caras contienen 1, 2, 3, 4, 5 y 6 puntos.
   - Una vez que los dados se hayan detenido, se calcula la suma de los puntos en las dos caras superiores.
   - Si la suma es 7 u 11, el jugador gana y acaba el juego
   - Si la suma es 2, 3 o 12 el jugador pierde y acaba el juego
   - Si la suma es 4, 5, 6, 8, 9 ó 10 se repite la jugada hasta que gane o pierda

8. Observe la siguiente secuencia:
   22 11 34 17 52 26 13 40 20 10 5 16 8 4 2 1
   Cada número de la misma se ha obtenido como la mitad del anterior si éste era par o el triple más uno si era impar. Por ejemplo, el 34 es par, luego le sigue el 17 que es su mitad; el 5 es impar, luego le sigue su triple 15 aumentado en 1 o sea 16.
   El número que encabeza la secuencia se denomina semilla, en este caso el 22. Independientemente de cual sea la semilla la secuencia siempre llega a 1 y esto se considera el fin de la misma. Dada una semilla se desea conocer:
   - Cuan larga es la secuencia
   - Cuál es el número más grande que contiene

9. Dos amigas necesitan enviarse por correo electrónico mensajes muy importantes y extremadamente reservados. Ante el temor de que alguien pueda leer los correos, deciden codificar los mensajes. La estrategia que utilizan es la siguiente: ciertos trozos del texto los escriben en orden inverso y los encierran entre paréntesis, de manera tal de no olvidar que

esos trozos deben ser leídos al revés. Obtener un programa al cual se ingrese el texto codificado y lo devuelva decodificado.
Ejemplo;

```
Hoy (.sh 22 sal a) (ed asac ne sominuer son) Marcelo.
```

```
Hoy a las 22 hs. nos reunimos en casa de Marcelo.
```

10. Obtener una función que calcule el M.C.D. de dos números. Generar dos soluciones, una por repetición y la otra por recursión.

11. Obtener una función que dado un numero entero calcule la cantidad de dígitos que posee.

12. Obtener una función que dados dos enteros devuelva 1 si son números amigos y 0 en caso contrario. Dos números amigos son dos enteros positivos a y b tales que a es la suma de los divisores propios de b, y b es la suma de los divisores propios de a. Por ejemplo (220, 284) son números amigos, ya que:
Los divisores propios de 220 son 1, 2, 4, 5, 10, 11, 20, 22, 44, 55 y 110, que suman 284.
Los divisores propios de 284 son 1, 2, 4, 71 y 142, que suman 220.

13. Obtener una función que calcule el factorial de un numero n. Generar dos soluciones, una por repetición y la otra por recursión.

14. 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.

15. 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.

16. Una empresa de colectivos tiene 3 líneas de 12 coches cada una. Por cada viaje el chofer entrega al llegar a la terminal una planilla con el número de coche (de 1 a 12), número de línea (de 1 a 3) y la recaudación del viaje. Las planillas se entregan sin ningún orden. Se pide informar:
    - La recaudación total por línea de colectivo
    - La recaudación total por coche
    - La recaudación total general.

17. Dado un arreglo de N valores enteros, se desea eliminar los repetidos.

18. Leer la cantidad de elementos de un vector y los datos en él. A continuación leer un numero k y reorganizar el arreglo en modo que los datos menores a K queden delante y los iguales o mayores detrás. No se debe utilizar un algoritmo de ordenación.

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

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

21. 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.

22. 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)

23. Una matriz mágica es una matriz cuadrada (tiene igual número de filas que de columnas) que tiene como propiedad especial que la suma de las filas, las columnas y las diagonales es igual. Construir un algoritmo que compruebe si una matriz de datos enteros es mágica o no.

24. En un terreno ondulado se quiere instalar antenas y es necesario que estas ocupen un lugar prominente. Para facilitar la elección se ha cuadriculado el terreno a los efectos de hacer un modelo digital del mismo. A cada una de las cuadrículas se le asigna un valor numérico entero representativo de la altura media del terreno dentro de esa cuadrícula. Definimos que una celda es una loma de prominencia k si dentro de un cuadrado de tamaño 2k+1 del cual la celda loma es centro, esta celda posee la mayor altura respecto de las celdas vecinas. Suponiendo que se ingresan las altura medias en una matriz A(MxN) y el valor k donde 0<k<10, determinar la posición de todas las lomas .
Ejemplo:

M = 5; N =10; k =1

t:

3	4	6	2	4	6	4	2	4	6
6	9	2	8	5	9	5	2	5	8
1	8	2	2	3	8	4	6	2	3
4	9	4	5	6	7	3	2	8	1
3	6	3	8	4	1	3	7	9	5

25. Escribir un programa que defina una estructura (Equipo) que permita contener los datos relativos a equipos de football.
Los datos a considerar son:
    - Nombre del equipo (tira de caracteres de longitud 20)
    - Cantidad de goles realizados (entero)
    - Cantidad de goles recibidos (entero)

    Ingresar la información correspondiente a un grupo de equipos y memorizarla en un arreglo.
    Mostrar como resultado los datos completos de los equipos cuya cantidad de goles realizados supere a los recibidos.

26. Un comercio tiene a la venta una serie de productos. Para cada producto se dispone de:
    - Descripción (tira de caracteres de longitud 20)
    - Stock del producto (entero)
    - Año de vencimiento (entero)

    El programa debe leer la información anterior mediante una estructura y generar un arreglo conteniendo los datos de todos los productos del comercio.
    Luego procesar los datos de modo de eliminar los productos vencidos.

27. En un concurso los participantes son sometidos a 10 pruebas diferentes, Cada prueba otorga un puntaje entre 0 y 10. Esta información debe ser representada mediante una estructura que contiene los siguientes campos:
    - Nombre del participante (tira de caracteres de longitud 20)
    - Una lista de 10 números enteros.

    El programa debe solicitar el ingreso de los datos de todos los concursantes y determinar:
    - El ganador de cada prueba
    - El ganador del concurso

28. Se dispone de la información correspondiente a un grupo de N atletas (máximo 100). Para cada uno se tiene:
    - Deporte (tira de caracteres de longitud 20)
    - Numero de medallas obtenidas (entero)
    - Datos personales
        - Nombre (tira de caracteres de longitud 20)
        - Fecha de nacimiento.(día, mes y año)
        - País de origen (tira de caracteres de longitud 20)

    Obtener un programa que luego de solicitar el ingreso de la información antes detallada, muestre los datos del atleta que ha ganado mayor número de medallas.

TRABAJO PRÁCTICO Nº 2 - Punteros

1. 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.

2. Unidad temática que incluye este trabajo práctico

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

3. 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.

TRABAJO PRÁCTICO Nº 3 - Programación Orientada a objetos

1. Objetivos de Aprendizaje

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

2. Unidad temática que incluye este trabajo práctico

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

3. 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.

12. Una empresa dedicada a la construcción de barrios cerrados necesita un sistema para gestionar la información de los mismos.
    Para cada barrio se tiene:
    - Nombre del barrio
    - Precio del metro cuadrado
    - Cantidad de lotes.

    Para cada lote se tiene:
    - Número del lote
    - Cantidad de metros cuadrados
    - Estado: 0=en venta, 1=vendido

    Generar una clase que permita representar la información del barrio.
    Considerar lo siguiente:
    - El constructor recibe el nombre del barrio, el número de lotes del mismo y el precio del metro cuadrado. Establece los datos del barrio y realiza la gestión dinámica de memoria para representar los lotes del mismo.
      - Generar los siguientes métodos:
      - Agrega_lote, que recibe el nuero del lote y los metros cuadrados del mismo y anexa la información del lote definiendo su estado como en venta.
      - Precio_lote que recibe el número del lote y devuelve el precio del mismo según sus metros cuadrados.
      - Vender, que recibe el número del lote y modifica el estado del lote señalándolo como vendido.
      - Vendidos, que devuelve la cantidad de lotes ya vendidos
      - EnVenta que devuelve la cantidad de lotes aun sin vender.

13. Crear una clase que permita representar y gestionar una tarjeta recargable.
    La información de la tarjeta es su número de identificación, el saldo y las 10 últimas operaciones.
    La información de cada operación es fecha (día-mes-año), cantidad (float) y tipo de operación (1 =cargar, 2=extraer, 3=consultar).
    Generar los métodos para las siguientes operaciones:
    - Cargar una cantidad.
    - Extraer una cierta cantidad, si hay saldo disponible.
    - Consultar el saldo
    - Consultar todas las últimas operaciones.

    El constructor debe crear una tarjeta vacía (con saldo 0) asignándole el número de identificación.

TRABAJO PRÁCTICO Nº 4 - Herencia y polimorfismo

1. 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.

2. Unidad temática que incluye este trabajo práctico

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

3. 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 puede ser mostrada (dibujada) u ocultada (borrada). Además una figura geométrica puede moverse especificando el desplazamiento (en X y en Y) de su centro.
   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. Obtener una clase base CUENTA que represente una cuenta bancaria. Para la misma se tiene la siguiente información:
   - Número de cuenta.
   - Saldo.

- Tasa de interés

El constructor genera una nueva cuenta a partir de la asignación del número de cuenta y de la tasa de interés. Toda nueva cuenta se crea con saldo 0.
Generar los siguientes métodos:

- Deposito: recibe el monto a depositar y registra dicho depósito
- Extracion: recibe el monto a extraer y registra dicho extracción. La operación puede realizarse solo si el saldo lo permite.
- Acreditacio: realiza la acreditación de intereses.

Generar una clase derivada de la anterior CUENTACC que representa una cuenta corriente.
La misma además de las características ya enunciadas para la clase base, dispone de un cierto valor permitido de descubierto, es decir admite un saldo negativo acorde a ese descubierto.
Determinar los atributos o métodos a agregar o redefinir para la clase derivada implementarla convenientemente.

4. 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.

5. Hacer una jerarquía de clases para modelar una obra de arte.
La raíz de la jerarquía es clase abstracta Obra_de_arte y sus clases derivadas serán Cuadro y Escultura.
La clase base Obra_de_arte tiene los siguientes atributos:
   - Título
   - Autor

   Definir los métodos que estime necesarios.
   Definir el método mostrar_caracteristicas() que debe mostrar lo siguiente:

   Nombre de la obra: xxx
   Autor: xxx
   Espacio necesario para la muestra: xxx

   El valor del espacio necesario se calcula mediante un método espacio() que dependiendo del tipo de obra de arte devolverá:
   - El cálculo de la superficie si se trata de un cuadro
   - El cálculo del volumen si se trata de una escultura.

   La subclase cuadro tiene dos atributos adicionales: Altura y Ancho
   La subclase escultura tiene tres atributos adicionales Altura, Ancho y Profundidad.

6. 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.

7. 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

TRABAJO PRÁCTICO Nº 5 - Clases parametrizadas y Manejo de Excepciones

2. 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.

2. Unidad temática que incluye este trabajo práctico

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

3. 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.