Metodología de la Programación II

Relación de Problemas

1. Problemas de Clases en C++.

Problema 1.1 *Haz una clase "Racional" para trabajar con números fraccionarios de la forma a/b (siendo a y b números enteros). Debes implementar, al menos, los siguientes métodos:*

- *Constructores para:*
  - *Crear un nuevo racional con valor cero.*
  - *Crear un racional a partir de un número entero (se entiende que el número fraccionario tendrá denomidador igual a uno).*
  - *Crear un racional a partir de dos números enteros (el primero será el numerador y el segundo el denominador).*
- *Destructor.*
- *Métodos para obtener y asignar el numerador y el denominador.*

Nota: No es necesario implementar ni el constructor de copias ni la sobrecarga del operador de asignación ya que no se hace uso de memoria dinámica y por tanto las operaciones por defecto son válidas.

Problema 1.2 *Haz una clase que permita almacenar coordenadas en un espacio bidimensional (parejas de números reales de la forma x,y). La clase proveerá métodos, al menos, para obtener o asignar dichas coordenadas en formato cartesiano o polar.*

Recuerde que las coordenadas polares corresponden a una pareja de valores (α,r), donde α hace referencia a un ángulo y r a una distancia positiva.

Problema 1.3 *Implementa una clase para trabajar con un vector dinámico de números enteros. Tenga en cuenta que la clase (*VectorDinamico*) debe incluir operaciones para:*

- *Cambiar el tamaño del vector.*
- *Modificar el elemento almacenado en una posición.*
- *Consultar el elemento de una posición.*
- *Consultar el tamaño del vector.*

Problema 1.4 *Implementa una clase para construir el TDA "pila". Piensa como hacerlo de forma que hagas uso de una representación basada en la clase "Vector dinámico" construida en el ejercicio 1.3.*

Problema 1.5 *Implementa una clase "Conjunto" en C++ que permita manipular el TDA conjunto de enteros.Para la representación interna de los datos usa una lista de celdas enlazadas. La clase debe contener, al menos, las siguientes operaciones:*

- *Constructor y destructor.*
- *Constructor de copia.*
- *Sobrecarga del operador de asignación.*
- *Método para añadir un nuevo elemento al conjunto.*
- *Método para borrar un elemento del conjunto.*
- *Método que reciba un número entero y devuelva true o false dependiendo de si el número pertenece o no al conjunto.*

- *Método que nos diga cuantos elementos tiene el conjunto.*
- *Método que devuelva un vector de enteros con todos los elementos del conjunto.*

Una vez hecha la clase, escriba un programa que lea de la entrada estándar un conjuntos de valores (un grupo de enteros hasta el valor cero), y escriba en la salida estándar todos los números introducidos sin repetir.

Problema 1.6 *Completa el ejercicio 1.1 sobrecargando los siguientes operadores: +, -, *, /, == y !=. Sobrecarga dichos operadores para poder operar racionales con racionales y racionales con enteros.*

Problema 1.7 *Completa el ejercicio 1.5 con los siguientes métodos o funciones:*

- *Sobrecarga el operador* `<<` *para mostrar en pantalla los elementos de un conjunto. Por ejemplo, si A es un conjunto entonces* `cout << A;` *muestra los elementos en pantalla.*
- *Sobrecarga el operador* `>>` *para leer desde teclado los elementos de un conjunto. Por ejemplo, si A es un conjunto entonces* `cin >> A` *lee por teclado números enteros y los mete en A.*
- *Sobrecarga el operador + para:*
  - *Hacer la unión de conjuntos. Por ejemplo, si A y B son dos conjuntos entonces A+B devolverá la unión de ambos.*
  - *Añadir un elemento a un conjunto. Por ejemplo, si A es un conjunto entonces A+4 devolverá el conjunto A tras haberle añadido el elemento 4. Sobrecarga el operador + para que también sea válido escribir 4+A.*
- *Sobrecarga el operador - para eliminar elementos de un conjunto:*
  - *Por ejemplo, A-4 devolverá el conjunto A sin el elemento 4.*
  - *Por ejemplo, A-B devolverá un conjunto donde están todos los elementos de A salvo aquellos que están en B.*
- *Sobrecarga el operador * para implementar un método que devuelva la intersección de dos conjuntos.*

Problema 1.8 *Implementa una clase que permita manipular un grupo de cartas de la baraja española. Dicha clase (*MontonCartas*) deberá disponer* únicamente *de los siguientes métodos:*

- *Constructor y destructor. Dispondremos de dos alternativas al construir un montón: que esté vacío (sin cartas) y que contenga todas las cartas de la baraja ordenadas por palos y números.*
- *Un método que mezcle las cartas que hay en el montón (de forma aleatoria).*
- *Un método que saque una carta del tope del montón (y la elimine de la baraja).*
- *Un método que nos diga cual es la carta del tope del montón (pero que no la saque).*
- *Un método que inserte una carta en la parte inferior del montón. No se permiten cartas repetidas en un mismo montón.*
- *Un método que nos informe de cuantas cartas quedan en el montón.*
- *Sobrecarga el operador* `<<` *para mostrar el montón en pantalla.*

Observe que puede ser conveniente definir previamente un tipo Carta*.*

Problema 1.9 *Escriba un programa para jugar a las 7 y 1/2 usando la clase del ejercicio 1.8. El juego deberá permitir jugar a un máximo de 10 jugadores tantas rondas como deseen. Al final de cada partida se mostrarán los resultados de todos los jugadores y se preguntará si se desea jugar una nueva ronda. Al finalizar el juego se imprimirá una estadística global de todos los jugadores.*

En este juego la banca tiene la baraja y le va dando cartas a cada jugador hasta que, o bien el jugador decida que no quiere mas, o bien el jugador se pase de 7 puntos y medio (en este último caso el jugador habrá perdido la partida). En una ronda ganará quien mas se acerque a 7 y medio sin pasarse. Por simplicidad, la banca no juega (sólo reparte cartas). El valor de las cartas es el que marque su número (del 1 al 7); las figuras valen todas 1/2 punto.