Programación Avanzada

PRÁCTICO 1

Ejercicio 1

Implementar en C++ una clase que represente un punto en un plano utilizando coordenadas cartesianas. Proveer operaciones para obtener y modificar la posición del punto en el plano.

A su vez, implementar en C++ una clase que represente un segmento en un plano. Proveer una operación que calcule el largo del segmento.

Ejercicio 2

a) Implementar en C++ una clase llamada Racional que permita definir y manipular números racionales, representados como un par de enteros. La clase deberá proveer las siguientes operaciones.

```
suma       : Racional x Racional -> Racional
diferencia : Racional x Racional -> Racional
producto   : Racional x Racional -> Racional
cociente   : Racional x Racional -> Racional
igualdad   : Racional x Racional -> bool
```

b) Agregar a la implementación de la clase Racional las operaciones habituales entre racionales y enteros. Sobrecargar los operadores que se consideren necesarios. Como ejemplo, la clase Racional debería soportar la siguiente aplicación:

```
void main() {
    Racional r1(2,3), r2;
    Racional r3(4), r4 = r1;
    if (r1 == r2)
        r1 = r1 + r2;
    else
        r1 = r1 - r2;

    if (r3 != r4)
        r3 = r3 * r4;
    else
        r3 = r3 / r4;

    if (r1 == 2)
        r1 = r1 + 1;
    else
        r1 = r1 - 1;
    if (r3 != 3)
        r3 = r3 * 2;
    else
        r3 = r3 / 2;
}
```

c) ¿Qué modificaciones habría que realizarle a la implementación para que soporte además la siguiente aplicación?

```
r3 = 3 * r2;
if (4 == (2 + r2))
    r3 = 1 / r2;
```

Ejercicio 3

a) Implementar en C++ una clase que represente a los conjuntos de enteros, utilizando como estructura de datos arreglos dinámicos. La clase deberá proveer al menos las siguientes operaciones:

```
agregar      : SetInt x int -> SetInt
remover      : SetInt x int -> SetInt
union        : SetInt x SetInt -> SetInt
diferencia   : SetInt x SetInt -> SetInt
interseccion : SetInt x SetInt -> SetInt
pertenece    : SetInt x int -> bool
esVacio      : SetInt -> bool
cantidadElem : SetInt -> int
esIgual      : SetInt x SetInt -> bool
```

b) Sobrecargar los operadores `+`, `-`, `==` y `!=` para que realicen la unión, diferencia, igualdad y desigualdad de conjuntos.

c) Ejecutar manualmente y en una computadora el siguiente programa comparando los resultados.

```
SetInt a,b,c,d,e;
a.agregar(0);
a.agregar(1);
b.agregar(1);
b.agregar(2);
c.agregar(0);
c.agregar(1);
c.agregar(2);
d = a + b;
e = c - b;

if (a == d)
    cout << "a == d\n";
else
    cout << "a != d\n";

if (e == a)
    cout << "e == a\n";
else
    cout << "e != a\n";
```

Ejercicio 4

a) Implementar en C++, utilizando la estructura de datos lista enlazada, una clase llamada SecInt que permita definir y manipular secuencias de enteros. La clase deberá proveer las siguientes operaciones. Sobrecargar los operadores que se consideren necesarios.

```
largo              : SecInt -> int
insertarEnPosicion : SecInt x int x int -> SecInt
eliminarPosicion   : SecInt x int -> SecInt
obtenerPosicion    : SecInt x int -> int
contiene           : SecInt x int -> bool
concatenar         : SecInt x SecInt -> SecInt
igualdad           : SecInt x SecInt -> bool
```

b) Ejecutar manualmente y en una computadora el siguiente programa comparando los resultados.

```
void main() {
    SecInt s1, s2;
    int pos = s1.largo();

    s1.insertarEnPosicion(pos, 3);
    s2.insertarEnPosicion(pos, 4);

    SecInt s3 = s2;

    s1.insertarEnPosicion(pos+1, s2.obtenerPosicion(pos));

    if (s1.contiene(4))
        s2 = s3 + s1;

    if (s3 == s2)
        for (int i = s3.largo()-1; i>=0; i--)
            s3.eliminarPosicion(i);
}
```

Ejercicio 5

a) Implementar en C++ una clase llamada MatrizInt que represente matrices de enteros de tamaño fijo. La clase deberá proveer al menos las siguientes funcionalidades.

```
setPosicion    : MatrizInt x int x int x int -> MatrizInt
getPosicion    : MatrizInt x int x int -> int
getNumFilas    : MatrizInt -> int
getNumColumnas : MatrizInt -> int
transpuesta    : MatrizInt -> MatrizInt
```

b) Implementar un procedimiento `main()` que permita ingresar una matriz, calcule su transpuesta y la imprima en pantalla.

Ejercicio 6

Implementar una clase genérica en C++ que represente una Pila Genérica utilizando una estructura de datos basada en listas enlazadas. La pila deberá proveer al menos las siguientes operaciones:

- **push**: agrega un elemento a la pila.
- **pop**: elimina el elemento en la cima de la pila.
- **top**: devuelve el elemento en la cima de la pila sin eliminarlo.
- **empty**: verifica si la pila está vacía.
- **size**: retorna la cantidad de elementos en la pila.