Государственный Университет Молдовы
Факультет Математики и Информатики
Департамент Информатики

Лабораторная работа №#4

по курсу “Объектно ориентированное программирование”
тема: Шаблоны (Templates)

Выполнил студент группы I2302:
Ciobanu
Stanislav.,
Проверил преподаватель:
G.Latul

Кишинэу, 2023

Условия работы

Вариант 1 : Создайте класс Строка.

1. Разработать шаблонный класс Матрица.
2. Перегрузить операции +, -. *.
3. В классе должен быть конструктор.
4. Предусмотреть член-функцию для печати элементов класса.

Задние 1

Создаём класс Matrix. Задаём ему 2 поля и создаём геттеры, сеттеры и деструктор.

```
// Matrix.h

#pragma once
#include <iostream>

using namespace std;

template<typename T>
class Matrix
{
private:
	T** _values;
	int _sizeX;
	int _sizeY;

public:

    int GetSizeX()
    {
        return _sizeX;
    }

        . . .


    void SetValues(T* values [])
    {

        for (int i = 0; i < _sizeX; i++)
        {
            for (int j = 0; j < _sizeY; j++)
            {
                _values[i][j] = values[i][j];
            }
        }
    }

    ~Matrix()
    {
        cout << "DESTRUCTOR IS WORKING\n";
        if (_values != nullptr)
        {
            for (int i = 0; i < _sizeX; ++i)
            {
                delete[] _values[i];
            }
            delete[] _values;
        }
    }

    . . .

};
```

Задние 2

Перегружаем операции +,-,*

```
// Matrix.h

Matrix<T>& operator+(Matrix<T>& matrix)
{
    if (_sizeX == matrix.GetSizeX() && _sizeY == matrix.GetSizeY())
    {
        Matrix<T>* mat = new Matrix<T>(_sizeX, _sizeY);

        for (int i = 0; i < _sizeX; i++)
        {
            for (int j = 0; j < _sizeY; j++)
            {
                mat->_values[i][j] = _values[i][j] + matrix.GetValues()[i][j];
            }
        }

        return *mat;

    }
    else
    {
        throw 0;
    }
}

Matrix<T>& operator-(Matrix<T>& matrix)
{
        . . .
}

Matrix<T>& operator*(Matrix<T>& matrix)
{
        . . .
}
```

Задние 3

Создаём конструкторы

```
// Matrix.h

Matrix(int sizeX, int sizeY) : _sizeX(sizeX), _sizeY(sizeY)
{
    _values = new T * [_sizeX];
    for (int i = 0; i < _sizeX; ++i)
    {
        _values[i] = new T[_sizeY];
    }
}

Matrix(int sizeX, int sizeY, T** values) : _sizeX(sizeX), _sizeY(sizeY),
_values(values) {}
```

Задние 4

Создаём функцию для печати

```
// Matrix.h

friend ostream& operator << (ostream& os, Matrix<T>&& matrix);


// Main.cpp


template<typename T>
ostream& operator<<(ostream& os, Matrix<T>& matrix)
{
    for (int i = 0; i < matrix.GetSizeX(); i++) {
        for (int j = 0; j < matrix.GetSizeY(); j++) {
            os << matrix.GetValues()[i][j] << " ";
        }
        os << "\n";
    }

    return os;
}
```

Используемые библиотеки

```
<iostream>          // Console
<fstream>           // Потоки ввода и вывода
<string>            // Строки

"Matrix.h"          // Headers
```