Министр науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«Национальный исследовательский университет ИТМО»

Факультет информационных технологий и программирования

Лабораторная работа №3

Перегрузка операторов

Выполнил студент группы № M3119
Самигуллин Руслан Рустамович
Подпись:

Проверил:
Повышев Владислав Вячеславович

Санкт-Петербург
2024

Задание:
Описать указанные типы данных и поместить их в отдельный заголовочный файл, в нем же описать операторы, указанные в варианте. Реализацию функций поместить с отдельный cpp файл. Написать программу, проверяющую правильность работы – для наглядности и лучшего усвоения материала использовать как явный, так и не явный метод вызова функций операторов (см. пример в конце задания).

1. Матрица 3x3
2. Стек целых чисел глубиной не более 100

1. Создание заголовочного файла для класса Matrix (Matrix.h):
- Объявлен класс Matrix.
- Объявлены конструкторы: без параметров и с массивом.
- Объявлены перегруженные операторы для математических операций и сравнения.
- Объявлена дружественная функция для вывода матрицы на экран.

```
#ifndef RS03_MATRIX_H
#define RS03_MATRIX_H
#include <iostream>

class Matrix {
private:
    double data[3][3];

public:
    // Конструкторы
    Matrix();
    Matrix(const double arr[3][3]);

    // Перегрузка операторов
    Matrix operator*(const Matrix& other) const; // Перемножение двух матриц
    Matrix operator*(double scalar) const; // Умножение матрицы на вещественное число
    Matrix operator+(const Matrix& other) const; // Сложение матриц
    Matrix operator-(const Matrix& other) const; // Вычитание матриц
    bool operator==(const Matrix& other) const; // Сравнение матриц на равенство
    bool operator!=(const Matrix& other) const; // Сравнение матриц на неравенство
    bool operator>(const Matrix& other) const; // Сравнение матриц (по сумме элементов)
    bool operator<(const Matrix& other) const; // Сравнение матриц (по сумме элементов)

    // Дружественная функция для вывода матрицы на экран
    friend std::ostream& operator<<(std::ostream& os, const Matrix& matrix);
};

#endif //RS03_MATRIX_H
```

2. Создание заголовочного файла для класса Stack (Stack.h):
- Класс Stack
- Конструкторы
- Перегруженные операторы для добавления и изъятия чисел из стека
- Функция для вывода стека на экран

```
#ifndef RS03_STACK_H
#define RS03_STACK_H

#include <iostream>

class Stack {
private:
    int data[100];
    int top;

public:
    // Конструктор
    Stack();

    // Перегрузка операторов
    Stack& operator<<(int value); // Добавление числа в стек
    Stack& operator>>(int& value); // Изъятие числа из стека

    // Функция для вывода стека на экран
    void printStack() const;
};

#endif //RS03_STACK_H
```

3. Реализация класса Matrix (Matrix.cpp):

- Перегруженные операторы для математических операций и сравнения
- Дружественная функция для вывода матрицы

1.

```cpp
#include "Matrix.h"

Matrix::Matrix() {
    // Заполним матрицу нулями
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            data[i][j] = 0;
        }
    }
}

Matrix::Matrix(const double arr[3][3]) {
    // Копируем переданную матрицу
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            data[i][j] = arr[i][j];
        }
    }
}

Matrix Matrix::operator*(const Matrix& other) const {
    Matrix result;
    // Перемножение матриц
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            for (int k = 0; k < 3; ++k) {
                result.data[i][j] += data[i][k] * other.data[k][j];
            }
        }
    }
    return result;
}

Matrix Matrix::operator*(double scalar) const {
    Matrix result;
    // Умножение матрицы на вещественное число
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            result.data[i][j] = data[i][j] * scalar;
        }
    }
    return result;
}

Matrix Matrix::operator+(const Matrix& other) const {
    Matrix result;
    // Сложение матриц
    for (int i = 0; i < 3; ++i) {
```

2.

```cpp
        for (int j = 0; j < 3; ++j) {
            result.data[i][j] = data[i][j] + other.data[i][j];
        }
    }
    return result;
}

Matrix Matrix::operator-(const Matrix& other) const {
    Matrix result;
    // Вычитание матриц
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            result.data[i][j] = data[i][j] - other.data[i][j];
        }
    }
    return result;
}

bool Matrix::operator==(const Matrix& other) const {
    // Сравнение матриц на равенство
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            if (data[i][j] != other.data[i][j]) {
                return false;
            }
        }
    }
    return true;
}

bool Matrix::operator!=(const Matrix& other) const {
    // Сравнение матриц на неравенство
    return !(*this == other);
}

bool Matrix::operator>(const Matrix& other) const {
    // Сравнение матриц (по сумме элементов)
    int sum1 = 0, sum2 = 0;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            sum1 += data[i][j];
            sum2 += other.data[i][j];
        }
    }
    return sum1 > sum2;
}
```

3.

```cpp
bool Matrix::operator<(const Matrix& other) const {
    // Сравнение матриц (по сумме элементов)
    int sum1 = 0, sum2 = 0;
    for (int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            sum1 += data[i][j];
            sum2 += other.data[i][j];
        }
    }
    return sum1 < sum2;
}

std::ostream& operator<<(std::ostream& os, const Matrix& matrix) {
    // Вывод матрицы на экран
    for(int i = 0; i < 3; ++i) {
        for(int j = 0; j < 3; ++j) {
            os << matrix.data[i][j] << " ";
        }
        os << std::endl;
    }
    return os;
}
```

4. Реализация класса Stack (Stack.cpp):
   - Перегруженные операторы для добавления и изъятия чисел из стека.
   - Функция для вывода стека на экран.

```cpp
// Stack.cpp
#include "Stack.h"
#include <iostream>

using namespace std;

Stack::Stack() : top(-1) {}

Stack& Stack::operator<<(int value) {
    // Добавление числа в стек
    if (top < 99 ) {
        data[++top] = value;
    } else {
        cout << "Stack overflow!" << endl;
    }
    return *this;
}

Stack& Stack::operator>>(int& value) {
    // Изъятие числа из стека
    if (top >= 0) {
        value = data[top--];
    } else {
        cout << "Stack underflow!" << endl;
    }
    return *this;
}

void Stack::printStack() const {
    // Вывод стека на экран
    for (int i = top; i >= 0; --i) {
        cout << data[i] << " ";
    }
    cout << endl;
}
```

5.1 Основной файл (main.cpp) - Матрица

1. Подключение заголовочных файлов и использование пространств имен.
2. Создание и использование объектов класса Matrix:
   - Ввод данных для двух матриц
   - Создание объектов матриц
   - Вывод матриц на экран
   - Демонстрация операций умножения сложения, вычитания и сравнения

5.2 Основной файл (main.cpp) – Стек

1. Ввод количества чисел для добавления в стек.
2. Добавление чисел в стек с помощью перегруженного оператора <<
3. Вывод стека на экран.
4. Ввод количества чисел для извлечения из стека.
5. Извлечение чисел из стека с помощью перегруженного оператора >>
6. Вывод стека на экран после извлечения.

```cpp
#include "Matrix.h"
#include "Stack.h"
using namespace std;

int main() {
    // Пример использования класса Matrix
    double arr1[3][3];
    double arr2[3][3];

    cout << "Matrix 1: ";
    for(int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cin >> arr1[i][j];
        }
    }

    cout << "Matrix 2: ";
    for(int i = 0; i < 3; ++i) {
        for (int j = 0; j < 3; ++j) {
            cin >> arr2[i][j];
        }
    }

    Matrix matrix1(arr1);
    Matrix matrix2(arr2);

    cout << "Matrix 1:" << endl;
    cout << matrix1 << endl;

    cout << "Matrix 2:" << endl;
    cout << matrix2 << endl;

    double value1;
    cout << "Введите вещественное число для умножения матрицы: "; cin >> value1;
    cout << matrix1*value1 << endl;
    cout << "Результат умножения первой матрицы и второй матрицы:" << endl;
    cout << matrix1*matrix2 << endl;
    cout << "Результат сложение первой матрицы и второй матрицы:" << endl;
    cout << "Результат вычитания первой матрицы из второй матрицы:" << endl;
    cout << matrix1-matrix2 << endl;
    cout << "Сравнение первой матрицы и второй матрицы:" << endl;
    if (matrix1==matrix2) {
        cout << "Матрицы равны" << endl;
    } else if (matrix1!=matrix2) {
        cout << "Матрицы не равны. ";
        if (matrix1 > matrix2) {
            cout << "Первая матрица больше, чем вторая матрица" << endl;
        } else if (matrix2 > matrix1) {
            cout << "Вторая матрица больше, чем первая матрица" << endl;
        }
    }
    cout << endl;
    // Создание объекта стека
    Stack stack;
    cout << "Сколько вы хотите добавить в стек?: ";
    int count1; cin >> count1; int value2;
    if (0 <= count1 and count1 <= 100) {
        cout << "Напишите все " << count1 << " числа: ";
        // Добавление чисел в стек с помощью оператора <<
        for (int i = 0; i < count1; i++) {
            cin >> value2;
            stack << value2;
        }
        // Вывод стека на экран с помощью простой функции printStack
        cout << "Stack: ";
        stack.printStack();

        cout << "Сколько вы хотите убрать из стека?:";

        int count2; cin >> count2; int value3;
        // Извлечение чисел из стека с помощью оператора >>
        for(int i = 0; i < count2; i++) {
            stack >> value3;
        }
        // Вывод стека на экран после извлечения
        cout << "Stack after delete: ";
        stack.printStack();

    } else if (count1 > 100) {
        cout << "Стек не превышает 100 чисел!" << endl;
    } else if (count1 < 0) {
        cout << "Стек должен иметь положительное количество!" << endl;
    }
    return 0;
}
```

Результат работы:

Matrix 1:
1 2 3
4 5 6
7 8 9
Matrix 2:
1 2 3
4 5 6
7 8 9
Введите вещественное число для умножения матрцицы: 1,5
 1,5 3 4,5
 6 7,5 9
10,5 12 13,5
Результат умножения первой матрицы и второй матрицы:
30 36 42
66 81 96
102 126 150
Результат сложение первой матрицы и второй матрицы:
2 4 6
8 10 12
14 16 18

Результат вычитания первой матрицы из второй матрицы:
0 0 0
0 0 0
0 0 0
Сравнение первой матрицы и второй матрицы:
Матрицы равны
#тест1
Сколько вы хотите добавить в стек?: 102
Стек не превышает 100 чисел!
#тест2
Сколько вы хотите добавить в стек?: -2
Стек должен иметь положительное количество!
#тест3
Сколько вы хотите добавить в стек?: 3
Напишите 3 числа: 1 2 3
Сколько хотите убрать?: 2
Стек после удаления: 1

1. Что такое перегрузка функций? Как понять, что функция перегружена (или по-другому – как должны отличаться функции, чтобы это можно было назвать перегрузкой)?

Ответ: Перегрузка функций – это возможность определения нескольких функций с одинаковым именем в одном области видимости, но с различными наборами параметров. Функции считается перегруженными, если они имеют разные типы или количество параметров. Например, функция принимает int, а другая double.

2. Аналогично вопросу 1 но про перегрузку операторов. Дополнительно: какие ещё ограничения на перегрузку операторов добавляются?

Ответ: Перегрузка операторов позволяет переопределить поведение операторов для пользовательских типов данных. Это делает код удобным для использования. Оператор считается перегруженным, если он применяется к пользовательским типам данных, а не только к встроенным типам, и имеет переопределение поведения для таких типов данных. Нельзя изменить приоритет операторов, нельзя создавать новые операторы, нельзя изменить количество операндов (всегда должен иметь 2).

3. Продемонстрировать явный и неявный метод вызова оператора в своём коде.

```
stack << value2;  // Явный вызов оператора << для добавления числа в стек
stack >> value3;  // Явный вызов оператора >> для извлечения числа из стека

matrix1 * value1; matrix1 * matrix2;  matrix1 + matrix2; matrix1 - matrix2; matrix1 == matrix2;
matrix1 != matrix2; matrix1 > matrix2; matrix1 < matrix2; // Явные вызова операторов
```