Министерство 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 № M3111
Гаврилов Алексей Евгеньевич
Подпись:

Проверил:
Повышев Владислав Вячеславович

Санкт-Петербург
2022

Текст задания

Согласно варианту 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).

Решение с комментариями

```
//fun.cpp
// Created by Volirvag on 25.02.2022.
//
#include <iostream>
#include "fun.h"

using namespace std;

namespace fun{
    Matrix::Matrix() {
        matrix = new int * [size_l];
        for (int i = 0; i < size_l; i++)
            matrix[i] = new int [size_h];

    }

    Matrix::~Matrix() {
        for (int i = 0; i < size_l; i++)
            delete matrix[i];
        delete matrix;
    }

    Matrix::Matrix(int a1, int a2, int a3, int a4, int a5, int a6, int a7,
int a8, int a9) {
        matrix = new int * [size_l];
        for (int i = 0; i < size_l; i++)
            matrix[i] = new int [size_h];
        matrix[0][0] = a1;
        matrix[0][1] = a2;
        matrix[0][2] = a3;
        matrix[1][0] = a4;
        matrix[1][1] = a5;
        matrix[1][2] = a6;
        matrix[2][0] = a7;
        matrix[2][1] = a8;
        matrix[2][2] = a9;
    }

    Matrix::Matrix(const Matrix &temp) {
        size_l = temp.size_l;
        size_h = temp.size_h;
        matrix = temp.matrix;
    }
    void Matrix::multiply_count(int value) {
        for (int i = 0; i < size_l; i++)
            for (int j = 0; j < size_h; j++)
                matrix[i][j] *= value;
    }

    void Matrix::print_matrix() {
        for (int i = 0; i < 3; i++) {
```

```
            for (int j = 0; j < 3; j++) {
                cout << matrix[i][j] << " ";
            }
            cout << endl;
        }
        cout << endl;
    }

    /*void  Matrix::plus_minus(struct Matrix &b, char signt) {
        if (signt == '+')
            for (int i = 0; i < size_l; i++)
                for (int j = 0; j < size_h; j++)
                    matrix[i][j] += b.matrix[i][j];
        else
            for (int i = 0; i < size_l; i++)
                for (int j = 0; j < size_h; j++)
                    matrix[i][j] -= b.matrix[i][j];
    }*/

    Matrix operator * (struct Matrix &a, struct Matrix &b)
    {
        Matrix temp;
        /*int **temp_matrix = new int * [3];
        for (int i = 0; i < 3; i++)
            temp_matrix[i] = new int [3];*/
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                temp.matrix[i][j] = 0;
                for (int k = 0; k < 3; k++)
                    temp.matrix[i][j] += a.matrix[i][k] * b.matrix[k][j];
            }
        }
        return temp;
    }

    /*void operator * (struct Matrix &a, int &t)
    {
        //Matrix temp;
        int **temp_matrix = new int * [3];
        for (int i = 0; i < 3; i++)
            temp_matrix[i] = new int [3];
        for (int i = 0; i < 3; i++)
        {
            for (int j = 0; j < 3; j++)
            {
                temp_matrix[i][j] = 0;
                for (int k = 0; k < 3; k++)
                    temp_matrix[i][j] += a.matrix[j][k] * t;
            }
        }
        a.matrix = temp_matrix;
    }*/

    Matrix operator + (struct Matrix &a, struct  Matrix &b){
        Matrix temp;

        for (int i = 0; i < a.size_l; i++)
            for (int j = 0; j < a.size_h; j++)
                temp.matrix[i][j] = a.matrix[i][j] + b.matrix[i][j];
        return temp;
    }
```

```
Matrix operator - (struct Matrix &a, struct  Matrix &b){
    Matrix temp;
    for (int i = 0; i < a.size_l; i++)
        for (int j = 0; j < a.size_h; j++)
            temp.matrix[i][j] = a.matrix[i][j] - b.matrix[i][j];
        return temp;
}

bool operator != (class Matrix &a, class Matrix &b) {
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            if (a.matrix[i][j] != b.matrix[i][j])
                return true;
    return false;
}

bool operator == (class Matrix &a, class Matrix &b) {
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            if (a.matrix[i][j] != b.matrix[i][j])
                return false;
    return true;
}

bool operator < (struct Matrix &a, struct Matrix &b) {
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            if (a.matrix[i][j] >= b.matrix[i][j])
                return false;
    return true;
}

bool operator > (struct Matrix &a, struct Matrix &b) {
    for (int i = 0; i < 3; i++)
        for (int j = 0; j < 3; j++)
            if (a.matrix[i][j] <= b.matrix[i][j])
                return false;
    return true;
}

FIFO::FIFO() {
    Q.first=new Node;
    Q.first->next=NULL;
    Q.last=Q.first;
    Q.size=0;
}

void FIFO::Add(int value) {
    {
        Q.last->next = new Node;
        Q.last = Q.last->next;
        Q.last->data = value;
        Q.last->next = NULL;
        Q.size++;
    }
}

void FIFO::Delete() {
    Q.first=Q.first->next;
    Q.size--;
}

int  FIFO::Top() {
     return Q.first->next->data;
```

```
    }
    void operator << (struct FIFO &A, int &value) {
        A.Add(value);
    }

    void operator >> (struct FIFO &A, int &value){
        value = A.Top();
        A.Delete();
    }

}
```

```
//fun.h
// Created by Volirvag on 25.02.2022.
//

#ifndef LAB_3_FUN_H
#define LAB_3_FUN_H

namespace fun
{
    class Matrix{
    private:
        int size_h = 3;
        int size_l = 3;
        int **matrix;

    public:
        Matrix();
        ~Matrix();
        Matrix(int a1, int a2, int a3, int a4, int a5, int a6, int a7, int
a8, int a9);
        Matrix(const Matrix &temp);
        void multiply_count(int value);
        void print_matrix();
        friend bool operator == (struct Matrix &a, struct  Matrix &b);
        friend bool operator != (class Matrix &a, class  Matrix &b);
        friend bool operator > (struct Matrix &a, struct Matrix &b);
        friend bool operator < (struct Matrix &a, struct Matrix &b);
        friend Matrix operator * (struct Matrix &a, struct Matrix &b);
        friend Matrix operator + (struct Matrix &a, struct Matrix &b);
        friend Matrix operator - (struct Matrix &a, struct Matrix &b);

    };

    class FIFO
    {
    private:
        struct Node
        {
            int data;
            Node *next;
        };
        struct Queue{
            int size;
            Node *first;
            Node *last;
        };
        Queue Q;
    public:
        FIFO();
        void Add(int value);
        void Delete();
```

```
        int Top();
        friend void operator << (struct FIFO &A, int &value);
        friend void operator >> (struct FIFO &A, int &value);
    };


};
#endif //LAB_3_FUN_H
```

//main

```
#include <iostream>
#include "fun.h"

using namespace std;
using namespace fun;

int main() {
    Matrix m(1,2,3,4,5,6,7,8,9);
    Matrix b(1,2,3,4,5,6,7,8,9);
    Matrix c(1,1,1,1,1,1,1,1,1);
    Matrix d;
    Matrix s;
    if (m != c)
        cout << "Yes" << endl;
    else
        cout << "No" << endl;
    if (m == b)
        cout << "Yes" << endl;
    else
        cout << "No" << endl;
    if (m == c)
        cout << "Yes" << endl;
    else
        cout << "No" << endl;
    if (m > c)
        cout << "Yes" << endl;
    else
        cout << "No" << endl;
    if (m < c)
        cout << "Yes" << endl;
    else
        cout << "No" << endl;
    cout << endl;
    m.print_matrix();
    b.print_matrix();
    c.print_matrix();
    s = m + c;
    s.print_matrix();
    s = b - c;
    s.print_matrix();
    d = m * b;
    d.print_matrix();
    c.multiply_count(5);
    c.print_matrix();

    FIFO fifo;
    int value;
    for (int i = 1; i < 7; i++){
        fifo << i;
    }

   for (int i = 1; i < 7; i++)
    {
```

```
        fifo >> value;
        cout << value << " ";
    }
    cout << endl;
    return 0;
}
```