МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Тихоокеанский государственный университет»

Кафедра «Программное обеспечение вычислительной техники и автоматизированных систем»

Перегрузка операторов

Лабораторная работа №4

по дисциплине «Объекно-ориентированное программирование»

Выполнил студент Чекулаев В. Ю.

Факультет, группа ФКФН, ПО(аб)-81

Проверил Федосеев А. А.

Хабаровск – 2020г.

1. Постановка задачи

На основе лабораторной работы № 3, перегрузить операторы для улучшения работы с созданной структурой данных. Согласно варианта реализовать набор перегруженных операторов. Для обработки ошибочных ситуаций использовать механизм исключений.

Вариант 6 : правый верхний треугольник квадратной матрицы. Размеры указываются в конструкторе. Операторы:

- [] - чтение и запись элемента по индексу

2. Содержание заголовочного файла класса-шаблона

```
#ifndef RIGHTTRIANGLE_H
#define RIGHTTRIANGLE_H

#include<cstddef>
#include<iostream>

template<class T>
class RightTriangle{
    private:
    class SubClass{
        public:
            SubClass(T* pointer, size_t i, size_t n);
            ~SubClass(){};

            T& operator[](size_t index) const;

        private:
            T* m_ptr;
            size_t m_index;
            size_t m_n;
    };

    public:
        RightTriangle(size_t n = 0);
        ~RightTriangle(){ if(m_triangle){ delete[] m_triangle; } }

        T get_at(size_t i, size_t j) const;
        void set_at(size_t i, size_t j,const T& elem);

        RightTriangle<T>::SubClass& operator[](size_t index) const;

    private:
        size_t m_n;
```

```
        T* m_triangle;
};

template<class T>
RightTriangle<T>::RightTriangle(size_t n) : m_n(n), m_triangle(nullptr){
    try{
        if(n <= 1) throw(1);

        u_int amount(0);
        while(n){
            amount += n;
            --n;
        }

        m_triangle = new T[amount]();
    }
    catch(int){
        std::cout << "RightTriangle<T>::RightTriangle(int n)::Uncorrect
size" << "\n";
        exit(1);
    }
    catch(std::bad_alloc){
        std::cout << "RightTriangle<T>::RightTriangle(int n)::Failed to
allocate memory" << "\n";
        exit(1);
    }
}

template<class T>
RightTriangle<T>::SubClass::SubClass(T* pointer, size_t i, size_t n) :
m_ptr(pointer), m_index(i), m_n(n){}

template<class T>
T& RightTriangle<T>::SubClass::operator[](size_t index) const{
    try {
        if(m_index > index || (m_index >= m_n && index >= m_n)) throw(1);

        return m_ptr[index];
    }
    catch (int) {
        std::cout << "RightTriangle<T>::operator[]::Index out of range\n";
        exit(1);
    }
}

template<class T>
typename RightTriangle<T>::SubClass& RightTriangle<T>::operator[](size_t
index)const{
    T* ptr = m_triangle;
    size_t i = index;
    size_t n(m_n);
    --n;

    while(i){
        ptr += n;
```

```
        --n; --i;
    }

    SubClass* obj = new SubClass(ptr, index, m_n);
    return *obj;
}

#endif // RIGHTTRIANGLE_H
```

3. Список исключительных ситуаций при работе класса-шаблона:

- Неверное задание размеров матрицы в конструкторе
- Невозможность выделения достаточного количества памяти для хранения матрицы
- Неверное задание индексов при чтении матрицы
- Неверное задание индексов при записи значения в матрицу

4. Текст одного из методов класса-шаблона, реализующего поставленную задачу

```
template<class T>
typename RightTriangle<T>::SubClass& RightTriangle<T>::operator[](size_t
index)const{
    T* ptr = m_triangle;
    size_t i = index;
    size_t n(m_n);
    --n;

    while(i){
        ptr += n;
        --n; --i;
    }

    SubClass* obj = new SubClass(ptr, index, m_n);
    return *obj;
}
```

5. Пример работы класса-шаблона для двух разных типов

Содержание файла main.cpp:

```
#include<iostream>

#include<cstddef>
#include"righttriangle.h"

class TestClass{
    public:
        TestClass(int num = 0) : m_num(num){}
        ~TestClass(){}

        int get() const{
            return m_num;
        }

    private:
        int m_num;
};

int main(){
    size_t n = 5;
    RightTriangle<double> obj1(n);
    obj1[2][3] = 8;
    obj1[0][2] = 2;
    obj1[4][4] = 5;
    obj1[0][0] = 3;
    obj1[2][3] = 7;

    std::cout << "obj1:" << "\n";
```

```
    for(size_t i = 0; i < n; ++i){
        for(size_t j = 0; j < n; ++j){
            if(i <= j){
                std::cout << obj1[i][j];
            } else{
                std::cout << " ";
            }
        }
        std::cout << "\n";
    }

    n = 8;
    RightTriangle<TestClass> obj2(n);
    obj2[2][6] = TestClass(8);
    obj2[0][0] = TestClass(4);
    obj2[3][5] = TestClass(1);
    obj2[6][6] = TestClass(7);
    obj2[5][6] = TestClass(3);
    obj2[2][3] = TestClass(2);

    std::cout << "\nobj2:" << "\n";
    for(size_t i = 0; i < n; ++i){
        for(size_t j = 0; j < n; ++j){
            if(i <= j){
                std::cout << obj2[i][j].get();
            } else{
                std::cout << " ";
            }
        }
        std::cout << "\n";
    }
}
```

Вывод программы:

```
obj1:
30200
 0000
  070
   00
    5

obj2:
40000000
 0000000
  020080
   00100
    0000
     030
      70
       0
```

6. Пример работы класса-шаблона по обработке исключительных ситуаций

Изменение в файле main.cpp:

Строка 24: «obj1[0][0] = 3;» изменено на «obj1[2][0] = 3;»

Вывод программы:

```
RightTriangle<T>::operator[]::Index out of range
```