Правительство Российской Федерации

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Отчёт к лабораторной работе №11

по дисциплине

«Языки программирования»

Работу выполнила

Студент группы СКБ222 _____ М. Х. Халимов

подпись, дата

Работу проверил _____ С. А. Булгаков

подпись, дата

Москва 2023

Содержание

Постановка задачи

Разработать шаблон класса *Stack* описывающий контейнер типа стек, параметр шаблона - тип данных для хранения значений. Интерфейс и реализацию разместить в файле *stack.hpp*. Класс должен содержать однонаправленный итератор (при реализации, класс *std::iterator* не использовать). Класс должен обладать интерфейсом, аналогичным классу *std::stack*.

В основной функции, размещенной в файле *main.cpp*, продемонстрировать применение разработанного класса и его методов.

1. Описание шаблона классов *stack* и *iterator*

Объявляются классы *stack* и *iterator*. Класс *stack* является совокупностью поля *std::vector<T> elems* и методов *empty(), T top ()* (где *T* – это тип возвращаемого значения), *push(), pop(), size()*. Класс *iterator* является совокупностью полей *std::vector<T>::reverse_iterator it, value_type, reference, pointer, difference_type, iterator_category* и методов *operator++(), operator==(), operator!=(), operator*(), end(), begin()*. В результате получаются контейнер и итератор.

2. Описание методов класса

Методы класса *stack: empty()* – определяет пустой ли контейнер, *top()* – возвращает *“верхний”* (последний) элемент контейнера, *push()* – добавляет элемент в контейнер, *pop()* – удаляет элемент из контейнера, size() – возвращает длину контейнера. Методы класса *iterator: operator++(), operator==(), operator!=(), operator*()* – перегрузка операторов *“++”, “==”, “!=”* и *“*”, end()* и *begin()* – отвечают за перемещение.

3. Функция main

Функция *main* включает в себя создание объектов класса *main* и вызов методов этого же класса. В результате работы функции в консоль выводится пример выполнения методов, описанных выше.

4. Результаты тестирования программы

Листинг

```
Stack1 elements: 5 4 3 2 1
Popped element from stack1: 5
Popped element from stack1: 4
Popped element from stack1: 3
Popped element from stack1: 2
Popped element from stack1: 1
Trying to access top element of an empty stack
Exception caught: Stack::top(): empty stack
```

Приложение А

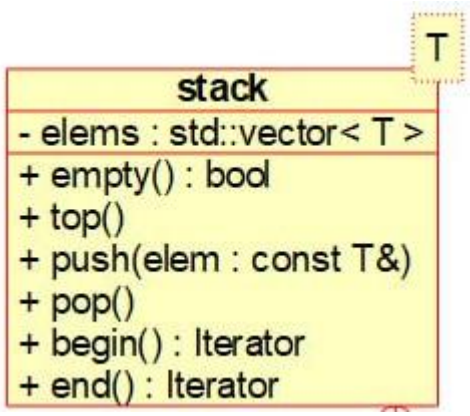
Исходный код

```
#ifndef STACK_HPP
#define STACK_HPP

#include <stdexcept>
#include <vector>

template<typename T>
class stack {
private:
    std::vector<T> elems;

public:
    bool empty() const {
        return elems.empty();
    }

    T top() {
        if (elems.empty()) {
            throw std::out_of_range("Stack::top(): empty stack");
        }
        return elems.back();
    }

    void push(const T& elem) {
        elems.push_back(elem);
    }

    void pop() {
        if (elems.empty()) {
            throw std::out_of_range("Stack::pop(): empty stack");
        }
        elems.pop_back();
    }

    class Iterator {
    private:
        typename std::vector<T>::reverse_iterator it;

    public:
        using value_type = typename std::vector<T>::value_type;
        using reference = typename std::vector<T>::reference;
        using pointer = typename std::vector<T>::pointer;
        using           difference_type           =           typename
std::vector<T>::difference_type;
        using iterator_category = std::forward_iterator_tag;
```

```
        Iterator(typename   std::vector<T>::reverse_iterator   rit)   :
it(rit) {}

        Iterator& operator++() {
            ++it;
            return *this;
        }

        Iterator operator++(int) {
            Iterator tmp(*this);
            operator++();
            return tmp;
        }

        bool operator==(const Iterator& other) const {
            return it == other.it;
        }

        bool operator!=(const Iterator& other) const {
            return it != other.it;
        }

        reference operator*() {
            return *it;
        }
    };

    Iterator begin() {
        return Iterator(elems.rbegin());
    }

    Iterator end() {
        return Iterator(elems.rend());
    }
};

#endif // STACK_HPP
```

Приложение Б

UML-диаграмма класса *stack*



UML-диаграмма класса *iterator*