**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и системы управления»

Кафедра ИУ5 «Системы обработки информации и управления»

Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №4

«Шаблоны классов»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-24б |  | преподаватель каф. ИУ5 |
| Байдаков В.М. |  |  |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

**Постановка задачи**

Дано описание класса MyStack (Приложение 1, файл MyStack.h), который реализует на основе односвязного списка динамическую структуру данных типа стек.

Разработайте реализацию класса MyStack.

Разработайте функцию (глобальную), которая выполняет разложение на простые множители целого числа N. Для хранения множителей функция должна использовать класс MyStack. Прототип функции: void Multipliers(int n, MyStack<DATA> &stack).

В функции main() распечатайте множители, которые функция Multipliers() записывает в стек, сначала по убыванию, а потом по возрастанию.

Классы ListNode и MyStack необходимо упаковать в самостоятельную статическую библиотеку.

**Разработка алгоритма**

**Класс ListNode:**

Члены класса:

INF d - хранимые данные узла

ListNode\* next - указатель на следующий узел в списке

Методы класса:

INF getValue() - возвращает хранимые узлом данные

Конструкторы:

ListNode() - инициализирует узел с nullptr в качестве следующего элемента

ListNode(INF a) - инициализирует узел с заданным значением a и nullptr в качестве следующего элемент

**Класс MyStack:**

Члены класса:

Node\* top - указатель на вершину стека

Методы класса:

bool empty() - проверяет, пуст ли стек

bool push(INF n) - добавляет элемент n на вершину стека

bool pop() - удаляет элемент с вершины стека

INF top\_inf() const - возвращает значение на вершине стека

MyStack() - конструктор

~MyStack() – деструктор, удаляет полностью стек

MyStack(const MyStack &s) – конструктор копирования

MyStack &operator=(const MyStack &s) – оператор присваивания

Main.cpp

void Multipliers(int n, MyStack<int> &stack) – заполняет стек множителями числа n.

**Текст программы**

CMakeLists.txt для проекта

cmake\_minimum\_required(VERSION 3.23)

set(project "lab4")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(SOURCES

    main.cpp

)

add\_subdirectory(MyStack)

add\_executable(${project} ${SOURCES})

target\_link\_libraries(${project} MyStack)

target\_include\_directories(

    ${PROJECT\_NAME} PUBLIC

    ${CMAKE\_SOURCE\_DIR}/MyStack

)

Main.cpp

#include "MyStack/MyStack.h"

#include <iostream>

void Multipliers(int n, MyStack<int> &stack)

{

    int i = 2, currentN = 1;

    if (n < 1)

    {

        return;

    }

    if (n == 1)

    {

        stack.push(1);

        return;

    }

    while (currentN < n)

    {

        if ((n / currentN) % i == 0)

        {

            currentN \*= i;

            stack.push(i);

        }

        else

        {

            i += 1;

        }

    }

}

int main()

{

    MyStack<int> stack;

    MyStack<int> stackReversed;

    int value = 3960;

    Multipliers(value, stack);

    std::cout << value << " = ";

    while (!stack.empty())

    {

        std::cout << stack.top\_inf();

        stackReversed.push(stack.top\_inf());

        stack.pop();

        if (!stack.empty())

            std::cout << " \* ";

    }

    std::cout << "\n"

              << value << " = ";

    while (!stackReversed.empty())

    {

        std::cout << stackReversed.top\_inf();

        stackReversed.pop();

        if (!stackReversed.empty())

            std::cout << " \* ";

    }

    return 0;

}

Библиотека MyStack

mystack.h

#pragma once

template <class INF, class FRIEND>

class ListNode

{

private:

    INF d;

    ListNode \*next;

    ListNode(void) { next = nullptr; }

    friend FRIEND;

    ListNode(INF a)

    {

        d = a;

        next = nullptr;

    }

    INF getValue() { return d; }

};

template <class INF>

class MyStack

{

    typedef class ListNode<INF, MyStack<INF>> Node;

    Node \*top;

public:

    bool empty(void)

    {

        if (top == nullptr)

            return true;

        else

            return false;

    }

    MyStack(void)

    {

        top = nullptr;

    }

    ~MyStack(void)

    {

        Node \*temp;

        while (top)

        {

            temp = top->next;

            delete top;

            top = temp;

        }

    }

    bool push(INF n)

    {

        Node \*new\_top = new Node(n);

        new\_top->next = top;

        top = new\_top;

        return 0;

    }

    bool pop(void)

    {

        if (empty())

        {

            return false;

        }

        Node \*temp = top->next;

        delete top;

        top = temp;

        return true;

    }

    INF top\_inf(void) const

    {

        return top->getValue();

    }

    MyStack(const MyStack &s)

    {

        Node \*newNode = s.top;

        MyStack rightStack;

        while (newNode)

        {

            rightStack.push(newNode->getValue());

            newNode = newNode->next;

        }

        top = nullptr;

        while (!rightStack.empty())

        {

            push(rightStack.top\_inf());

            rightStack.pop();

        }

    }

    MyStack &operator=(const MyStack &s)

    {

        if (this != &s)

        {

            Node \*temp;*//очищение стэка, то же самое что и в деструкторе*

            while (top)

            {

                temp = top->next;

                delete top;

                top = temp;

            }

            Node \*newNode = s.top;

            MyStack rightStack;

            while (newNode)

            {

                rightStack.push(newNode->getValue());

                newNode = newNode->next;

            }

            top = nullptr;

            while (!rightStack.empty())

            {

                pop();

                push(rightStack.top\_inf());

                rightStack.pop();

            }

        }

        return \*this;

    }

};

MyStack.cpp

#include "MyStack.h"

CMakeLists.txt для библиотеки MyStack

cmake\_minimum\_required(VERSION 3.23)

set(project "MyStack")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(SOURCES

    MyStack.cpp

)

set(HEADERS

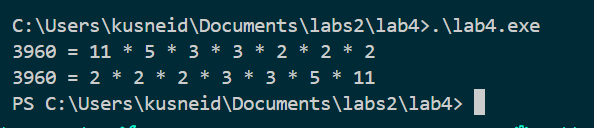
    MyStack.h

)

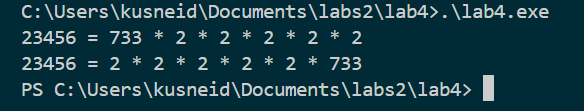
add\_library(${project} STATIC ${SOURCES} ${HEADERS})

Результаты выполнения

При value=3960



При value=23456



При value=467803133

