**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 ИУ5-23Б |  | преподаватель каф. ИУ5 |
| Лачина Екатерина Андреевна |  | Аладин Д.В. |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2024 г.

1. **Задача**
2. Разработайте реализацию класса MyStack.
3. 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).
4. В функции main() распечатайте множители, которые функция Multipliers() записывает в стек, сначала по убыванию, а потом по возрастанию. Например, для N=3960 программа должна вывести:

3960=11 \* 5 \* 3 \* 3 \* 2 \* 2 \* 2  
3960=2 \* 2 \* 2 \* 3 \* 3 \* 5 \* 11

1. Классы ListNode и MyStack необходимо упаковать в самостоятельную статическую библиотеку.
2. **Листинг**

**Папка Laba2sem4**

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.2)

set(project "Laba2sem4")

project(**${project}**)

set(CMAKE\_CXX\_STANDARD 17)

set(**${project}**\_SOURCES

Laba2sem4.cpp)

set(**${project}**\_HEADERS

Laba2sem4.h)

set(**${project}**\_SOURCE\_LIST

${**${project}**\_SOURCES}

${**${project}**\_HEADERS})

add\_library(**${project}**

STATIC

${**${project}**\_SOURCE\_LIST})

**Laba2sem4.cpp**

// Laba2sem4.cpp: определяет точку входа для приложения.

//

#include "Laba2sem4.h"

**Laba2sem4.h**

#pragma once

// Файл MyStack.h

// Шаблонный класс MyStack на основе односвязного списка.

#ifndef MyStack\_h // защита от повторной компиляции

#define MyStack\_h // модуль подключен

// Шаблонный класс ListNode (узел односвязного списка)

template<class INF, class FRIEND>

class ListNode // узел списка

{

private:

INF d; // информационная часть узла

ListNode\* next; // указатель на следующий узел списка

ListNode(void) :d{} { next = nullptr; } //конструктор

friend FRIEND;

};

// Шаблонный класс MyStack на основе односвязного списка.

template<class INF>

class MyStack {

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

Node\* top;

public:

MyStack(void); // конструктор

MyStack(const MyStack& other);

~MyStack(void); // освободить динамическую память

bool empty(void); // стек пустой?

bool push(INF n); // добавить узел в вершину стека

bool pop(void); // удалить узел из вершины стека

INF top\_inf(void); // считать информацию из вершины стека

void Clear();

MyStack& operator = (const MyStack& other);

};

template<class INF>

MyStack<INF>::MyStack(void) {

top = nullptr;

}

template<class INF>

MyStack<INF>::~MyStack(void) {

while (!empty()) {

pop();

}

}

template<class INF>

bool MyStack<INF>::empty(void) {

return top == nullptr;

}

template<class INF>

bool MyStack<INF>::push(INF n) {

Node\* temp = new Node;

if (temp == nullptr) {

return false;

}

temp->d = n;

temp->next = top;

top = temp;

return true;

}

template<class INF>

bool MyStack<INF>::pop(void) {

if (empty()) {

return false;

}

Node\* temp = top;

top = top->next;

delete temp;

return true;

}

template<class INF>

void MyStack<INF>::Clear() {

while (pop());

}

template<class INF>

INF MyStack<INF>::top\_inf(void) {

if (empty()) {

throw "Stack is empty";

}

return top->d;

}

template class MyStack<int>;

template <class INF>

MyStack<INF>::MyStack(const MyStack& other) {

if (other.top) {

top = new Node(\*other.top);

Node\* current = top;

Node\* otherCurrent = other.top->next;

while (otherCurrent) {

current->next = new Node(\*otherCurrent);

current = current->next;

otherCurrent = otherCurrent->next;

}

}

}

template<class INF>

MyStack<INF>& MyStack<INF>::operator = (const MyStack& other) {

if (this != &other) {

Clear();

if (other.top) {

top = new Node(\*other.top);

Node\* current = top;

Node\* otherCurrent = other.top->next;

while (otherCurrent) {

current->next = new Node(\*otherCurrent);

current = current->next;

otherCurrent = otherCurrent->next;

}

}

}

return \*this;

}

#endif

**Главные файлы**

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.2)

set(project "Laba4")

project(**${project}**)

set(CMAKE\_CXX\_STANDARD 17)

set(**${project}**\_SOURCES

main.cpp)

add\_subdirectory(Laba2sem4)

set(**${project}**\_SOURCE\_LIST

${**${project}**\_SOURCES})

add\_executable(**${project}**

${**${project}**\_SOURCE\_LIST})

target\_link\_libraries(**${project}** Laba2sem4)

**main.cpp**

#include <iostream>

#include "Laba2sem4/Laba2sem4.h"

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

int temp\_n = n;

int\* arr = new int[30];

int count = 0;

while (n % 2 == 0) {

stack.push(2);

arr[count++] = 2;

n = n / 2;

}

for (int i = 3; i <= sqrt(n); i = i + 2) {

while (n % i == 0) {

stack.push(i);

arr[count++] = i;

n = n / i;

}

}

if (n > 2) {

stack.push(n);

arr[count++] = n;

}

std::cout << temp\_n << "=";

while (!stack.empty()) {

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

stack.pop();

if (!stack.empty()) {

std::cout << " \* ";

}

}

std::cout << std::endl;

for (int i = count - 1; i > -1; i--) {

stack.push(arr[i]);

}

std::cout << temp\_n << "=";

while (!stack.empty()) {

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

stack.pop();

if (!stack.empty()) {

std::cout << " \* ";

}

}

std::cout << std::endl;

delete[] arr;

}

int main() {

setlocale(LC\_ALL, "Russian");

MyStack<int> stack;

stack.push(5);

stack.push(19);

MyStack<int> stack1;

stack1 = stack;

MyStack<int> stack2{ stack1 };

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

std::cout << stack1.top\_inf() << " ";

std::cout << stack2.top\_inf() << std::endl;

stack.pop(); stack1.pop(); stack2.pop();

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

std::cout << stack1.top\_inf() << " ";

std::cout << stack2.top\_inf() << std::endl;

int value;

std::cout << "Введите целое число:" << std::endl;

std::cin >> value;

MyStack<int> stack\_;

Multipliers(value, stack\_);

return 0;

}

1. **Тестирование**







