**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 |
| Чернев Н.А. |  |  |
| Подпись и дата: 19.03.2024 |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

## Задание

Дано описание класса MyStack ([Приложение 1](https://iu5edu.ru/wiki/cpp2/docs/labs/lab4/Instructions/SupplementMaterial1/), файл MyStack.h), который реализует на основе односвязного списка динамическую структуру данных типа стек.

1. Разработайте реализацию класса MyStack.
2. 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).
3. 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.

**Разработка алгоритма**

Библиотека MyStack:

* Класс ListNode<INF, FRIEND>:
  + Поля:
    - INF value; - данные узла (указывается в <>)
    - ListNode \*next; - указатель на следующий узел
  + Методы:
    - ListNode() - конструктор без параметров, создает узел
    - ListNode(INF \_value) - конструктор с параметром, создает узел и записывает в value передаваемое значение
  + Дружественные классы:
    - friend FRIEND;
* Класс MyStack<INF>:
  + typedef class ListNode<INF, MyStack<INF>> Node; - добавление “синонима” к ListNode
  + Поля:
    - Node \*top; - указатель на вершину стека
  + Методы
    - MyStack() - конструктор без параметров, создает пустой стек
    - MyStack(const MyStack &st) - конструктор копирования, создает стек со значениями из передаваемого стека
    - MyStack& **operator** = (**const** MyStack &st) - перегрузка оператора = для MyStack = MyStack
    - ~MyStack() - деструктор, удаляет динамическую память, выделенную под узлы стека
    - bool empty() - возвращает true, если стек пустой, иначе false
    - bool push(INF n) - добавляет элемент на вершину стека, возвращает true при удачном добавлении
    - bool pop() - удаляет верхний элемент стека и возвращает true при успешном удалении, false в противном случае
    - INF top\_inf() - возвращает значение из вершины стека

Функции в main.cpp:

* **void** multipliers(**int** n, MyStack<**int**> &stack) - принимает число n и копию стека, раскладывает n на множители и добавляет их в стек

**Текст программы**

Main.cpp

#include "MyStack/MyStack.h"

#include <iostream>

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

**int** i = 2, cur = 1;

**if** (n < 1) **return**;

**if** (n == 1) {

stack.push(1);

**return**;

}

**while** (cur < n) {

**if** ((n / cur) % i == 0) {

cur \*= i;

stack.push(i);

}

**else** {

i += 1;

}

}

}

**int** main() {

MyStack<**int**> stack, stack2;

**int** n = 3960;

multipliers(n, stack);

MyStack<**int**> a;

a = stack;

**while** (!a.empty()) {

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

a.pop();

**if** (!a.empty()) std::cout << ' ';

}

std::cout << '\n';

std::cout << n << " = ";

**while** (!stack.empty()) {

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

stack2.push(stack.top\_inf());

stack.pop();

**if** (!stack.empty()) std::cout << " \* ";

}

std::cout << '\n' << n << " = ";

**while** (!stack2.empty()) {

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

stack2.pop();

**if** (!stack2.empty()) std::cout << " \* ";

}

**return** 0;

}

CMakeLists.txt проекта

cmake\_minimum\_required(VERSION 3.23)

set(project "lab04")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(${project}\_SOURCES

main.cpp)

add\_subdirectory(MyStack)

set(${project}\_SOURCE\_LIST

${${project}\_SOURCES})

add\_executable(${project}

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

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

MyStack.h

#pragma once

**template**<**class** INF, **class** FRIEND>

**class** ListNode

{

**private**:

INF value;

ListNode \*next;

ListNode() { next = **nullptr**; }

ListNode(INF \_value) {

value = \_value;

next = **nullptr**;

}

**friend** FRIEND;

};

**template**<**class** INF>

**class** MyStack {

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

Node \*top;

**public**:

MyStack() {

top = **nullptr**;

}

MyStack(**const** MyStack &st) {

Node \*tmp = st.top;

MyStack tmp\_st;

**while** (tmp) {

tmp\_st.push(tmp->value);

tmp = tmp->next;

}

top = **nullptr**;

**while** (!tmp\_st.empty()) {

push(tmp\_st.top\_inf());

tmp\_st.pop();

}

}

MyStack& **operator** = (**const** MyStack &st) {

Node \*tmp = st.top;

MyStack tmp\_st;

**while** (tmp) {

tmp\_st.push(tmp->value);

tmp = tmp->next;

}

top = **nullptr**;

**while** (!tmp\_st.empty()) {

push(tmp\_st.top\_inf());

tmp\_st.pop();

}

**return** \***this**;

}

~MyStack() {

Node \*tmp;

**while** (top) {

tmp = top->next;

**delete** top;

top = tmp;

}

}

**bool** empty() {

**return** (!top);

}

**bool** push(INF n) {

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

tmp->next = top;

top = tmp;

**return** **true**;

}

**bool** pop() {

**if** (empty()) **return** **false**;

Node \*tmp = top->next;

**delete** top;

top = tmp;

**return** **true**;

}

INF top\_inf() {

**return** top->value;

}

};

MyStack.cpp

#include "MyStack.h"

CMakeLists.txt библиотеки MyStack

cmake\_minimum\_required(VERSION 3.23)

set(project "MyStack")

project(${project})

set(CMAKE\_CXX\_STANDARD 17)

set(${project}\_SOURCES

MyStack.cpp)

set(${project}\_HEADERS

MyStack.h)

set(${project}\_SOURCE\_LIST

${${project}\_SOURCES}

${${project}\_HEADERS})

add\_library(${project}

STATIC

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

**Анализ результатов**

