**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 ИУ5-22Б |  | Преподаватель каф. ИУ5 |
| Поляков Л. С. |  | Бурмистрова М.В. |
| Подпись и дата: |  | Подпись и дата: |
|  |  |  |

Москва, 2024 г.

# Постановка задачи

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).

1. 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

**Требования:**

* Классы ListNode и MyStack необходимо упаковать в самостоятельную статическую библиотеку.
* Не забудьте реализовать конструкторы копирования и перегрузки оператора = для ListNode и MyStack.

# Разработка алгоритма

## Структура проекта

.

├── CMakeLists.txt

├── MyStack

│   ├── CMakeLists.txt

│   ├── MyStack.cpp

│   └── MyStack.hpp

└── main.cpp

## Текст программы

**MyStack.hpp**

#ifndef LAB\_4\_MYSTACK\_HPP  
#define LAB\_4\_MYSTACK\_HPP  
  
#include <iostream>  
  
  
template <class INF, class FRIEND>  
class ListNode  
{  
private:  
 INF d;  
 ListNode \*next;  
 ListNode() { next = nullptr; d = INF{};}  
 ListNode(ListNode &tmp);  
 friend FRIEND;  
};  
  
  
template <class INF>  
class MyStack  
{  
 typedef class ListNode<INF, MyStack<INF>> Node;  
 Node \*top;  
  
public:  
 MyStack();  
 ~MyStack();  
 MyStack(const MyStack &tmp);  
 bool empty();  
 bool push(INF n);  
 bool pop();  
 void clear();  
 INF top\_inf();  
 MyStack<INF>& operator=(const MyStack<INF>& tmp);  
 void show();  
 void showReverse();  
};  
  
template<class INF>  
void MyStack<INF>::clear() {  
 while (!this->empty()) {  
 this->pop();  
 }  
}  
  
template <class INF>  
bool MyStack<INF>::empty() {  
 return top == nullptr;  
}  
  
template <class INF>  
MyStack<INF>::MyStack() {  
 top = nullptr;  
}  
  
template <class INF>  
MyStack<INF>::~MyStack() {  
 this->clear();  
}  
  
template <class INF>  
inline MyStack<INF>::MyStack(const MyStack &tmp) {  
 top = nullptr;  
 top = new Node{};  
 top->d = tmp.top->d;  
  
 Node \*cur {tmp.top}, \*toCur{top};  
  
 while (cur->next != nullptr) {  
 toCur->next = new Node{};  
 toCur->next->d = cur->next->d;  
 toCur = toCur->next;  
 cur = cur->next;  
 }  
}  
  
template <class INF>  
bool MyStack<INF>::push(INF n) {  
 if (empty()) {  
 top = new Node{};  
 top->d = n;  
 return false;  
 }  
 Node \*cur{top};  
 while (cur->next != nullptr) {  
 cur = cur->next;  
 }  
 cur->next = new Node{};  
 cur->next->d = n;  
 return true;  
}  
  
template <class INF>  
bool MyStack<INF>::pop() {  
 if (top->next == nullptr) {  
 delete top;  
 top = nullptr;  
 return false;  
 }  
  
 Node \*cur{top};  
 while (cur->next->next != nullptr) {  
 cur = cur->next;  
 }  
 delete cur->next;  
 cur->next = nullptr;  
 return true;  
}  
  
template <class INF>  
INF MyStack<INF>::top\_inf() {  
 return top->d;  
}  
template <class INF>  
inline MyStack<INF>& MyStack<INF>::operator=(const MyStack<INF>& tmp) {  
 if (&tmp == this) {  
 return \*this;  
 }  
 this->clear();  
 top = nullptr;  
 top = new Node{};  
 top->d = tmp.top->d;  
  
 Node \*cur {tmp.top}, \*toCur{top};  
  
 while (cur->next != nullptr) {  
 toCur->next = new Node{};  
 toCur->next->d = cur->next->d;  
 toCur = toCur->next;  
 cur = cur->next;  
 }  
 return \*this;  
}  
template <class INF>  
inline void MyStack<INF>::showReverse() {  
 const Node\* cur{top};  
 bool firstElem{};  
 while (cur != nullptr) {  
 if (firstElem) {  
 std::cout << "\* ";  
 }  
 std::cout << cur->d << " ";  
 cur = cur->next;  
 firstElem = true;  
 }  
 std::cout << std::endl;  
}  
template <class INF>  
inline void MyStack<INF>::show() {  
 const Node \*last{top};  
 while (last->next != nullptr) {  
 last = last->next;  
 }  
 std::cout << last->d;  
 if (last != top) {  
 for (;;) {  
 Node \*preLast = top;  
 while (preLast->next != last) {  
 preLast = preLast->next;  
 }  
 std::cout << " \* " << preLast->d;  
 last = preLast;  
  
 if (last == top) {  
 break;  
 }  
 }  
 }  
 std::cout << std::endl;  
}  
  
void Multipliers(int num, MyStack<int> &stack);  
  
  
#endif //LAB\_4\_MYSTACK\_HPP

**MyStack.cpp**

#include "MyStack.hpp"  
  
void Multipliers(int num, MyStack<int> &stack)  
{  
 while (!stack.empty()) {  
 stack.pop();  
 }  
 int sign = num < 0 ? -1 : 1;  
 num = abs(num);  
 if (num == 0) {  
 stack.push(0);  
 }  
 if (num == 1) {  
 stack.push(1 \* sign);  
 }  
 int i{2}, count{};  
 while ((num > 2) and (i <= num)) {  
 if (num % i == 0) {  
 if (count == 0) {  
 stack.push(i \* sign);  
 } else {  
 stack.push(i);  
 }  
 count++;  
 num /= i;  
 i = 1;  
 }  
 i++;  
 }  
 if (num == 2)  
 stack.push(2);  
}

**MyStack/CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.26)  
  
set(PROJECT\_NAME MyStack)  
project(${PROJECT\_NAME})  
  
set(CMAKE\_CXX\_STANDARD 20)  
  
set(${PROJECT\_NAME}\_HEADERS  
 MyStack.hpp  
)  
  
set(${PROJECT\_NAME}\_SOURCES  
 MyStack.cpp  
)  
  
set(${PROJECT\_NAME}\_SOURCE\_LIST  
 ${${PROJECT\_NAME}\_SOURCES}  
 ${${PROJECT\_NAME}\_HEADERS}  
)  
  
add\_library(${PROJECT\_NAME} STATIC ${${PROJECT\_NAME}\_SOURCE\_LIST})

**main.cpp**

#include <iostream>  
#include "MyStack/MyStack.hpp"  
  
  
int main() {  
 int num{};  
 std::cin >> num;  
 MyStack<int> stack;  
 Multipliers(num, stack);  
 MyStack<int> stackCopy(stack);  
 stackCopy.show();  
 stack.showReverse();  
  
 std::cout << std::endl;  
  
 MyStack<char> stack1;  
 stack1.push('a');  
 stack1.push('b');  
 stack1.push('c');  
 stack1.show();  
 MyStack<char> stack2(stack1);  
 stack2.show();  
 MyStack<char> stack3;  
 stack3 = stack2;  
 stack3.show();  
 return 0;  
}

**CMakeLists.txt**

cmake\_minimum\_required(VERSION 3.26)  
  
set(PROJECT\_NAME Lab\_4)  
project(${PROJECT\_NAME})  
  
set(CMAKE\_CXX\_STANDARD 20)  
  
set(${PROJECT\_NAME}\_HEADERS  
  
)  
  
set(${PROJECT\_NAME}\_SOURCES  
 main.cpp  
)  
  
add\_subdirectory(MyStack)  
  
set(${PROJECT\_NAME}\_SOURCE\_LIST  
 ${${PROJECT\_NAME}\_SOURCES}  
 ${${PROJECT\_NAME}\_HEADERS}  
)  
  
add\_executable(${PROJECT\_NAME} ${${PROJECT\_NAME}\_SOURCE\_LIST})  
  
target\_link\_libraries(${PROJECT\_NAME} MyStack)

# Анализ Результатов

Изображение выглядит как текст, снимок экрана, Шрифт, дизайн

Автоматически созданное описание

# Вывод

Я научился

* Работать с шаблонами классов в C++
* Работать со статическими библиотеками с использованием Cmake

Познакомился с такими структурами данных как связанный список и стек, понял принцип их работы и научился реализовывать их в коде.