**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. Создание класса Stack.»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-23Б |  | преподаватель каф. ИУ5 |
| Поляков Данила |  | Правдина А.Д. |
| Подпись и дата: 28.02.2022 |  | Подпись и дата: 28.02.2022 |

Москва, 2022 г.

**Постановка задачи**

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

1. Разработайте реализацию интерфейса класса в виде файла MyStack.cpp.

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

**Текст программы**

**Листинг кода программы:**

**MyStack.hpp:**

*// Copyright 2021 qqq <polyakovdd@student.bmstu.ru>  
  
  
#ifndef* CODE\_MY\_STACK\_HPP  
*#define* CODE\_MY\_STACK\_HPP  
*//Шаблонный класс MyStack на основе односвязного списка.  
// 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() { 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();  
 ~MyStack();  
 *bool* empty();  
 *void* push(INF n);  
 *bool* pop();  
 INF top\_inf();  
 *void* reverse();  
 *void* print();  
};  
  
  
  
*#endif //CODE\_MY\_STACK\_HPP*

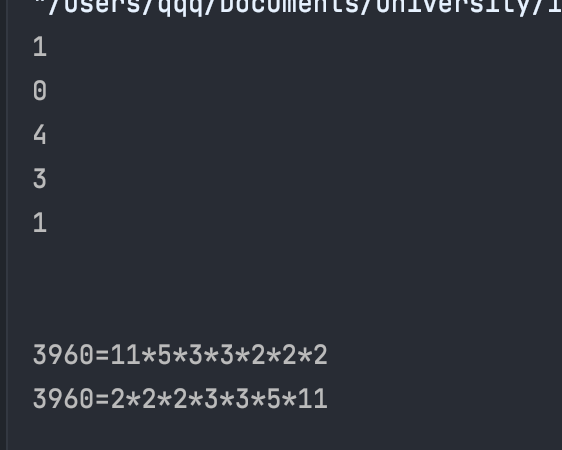
**MyStack.cpp**

*// Copyright 2021 qqq <polyakovdd@student.bmstu.ru>  
  
#include* "MyStack.hpp"  
*#include* <iostream>  
  
*template*<*class* INF>  
*bool* MyStack<INF>::empty() {  
 *if* (top == *nullptr*)  
 *return true*;  
 *else  
 return false*;  
}  
  
*template*<*class* INF>  
MyStack<INF>::MyStack() {  
 *this*->top = *nullptr*;  
}  
  
*template*<*class* INF>  
MyStack<INF>::~MyStack() {  
 Node tempNode;  
 *while* (top != *nullptr*) {  
 tempNode = (top->next);  
 *delete* top;  
 top = &tempNode;  
 }  
}  
  
*template*<*class* INF>  
*void* MyStack<INF>::push(INF n) {  
 Node \*tempNode = *new* Node;  
 tempNode->d = n;  
 tempNode->next = top;  
 top = tempNode;  
}  
  
*template*<*class* INF>  
*bool* MyStack<INF>::pop() {  
 *if* (top == *nullptr*) *return false*;  
 Node \*tempNode = top->next;  
 *delete* top;  
 top = tempNode;  
 *return true*;  
}  
  
*template*<*class* INF>  
INF MyStack<INF>::top\_inf() {  
 *if* (top == *nullptr*) *return* 0;  
 *return* top->d;  
}  
  
*template*<*class* INF>  
*void* MyStack<INF>::reverse() {  
 Node \*previousNode = top;  
 Node \*node = top->next;  
 Node \*nextNode;  
 previousNode->next = *nullptr*;  
 *do* {  
 nextNode = node->next;  
 node->next = previousNode;  
 previousNode = node;  
 node = nextNode;  
 } *while* (node->next != *nullptr*);  
 node->next = previousNode;  
 top = node;  
}  
  
*template*<*class* INF>  
*void* MyStack<INF>::print() {  
 Node \*node = top;  
 *while* (node != *nullptr*) {  
 std::cout << node->d << '\*';  
 node = node->next;  
 }  
 std::cout << "\b \n";  
}

**main.cpp**

*#include* <iostream>  
  
  
*#include* "MyStack.hpp"  
*#include* "MyStack.cpp"  
  
*#define* N 3960  
  
*void* Multipliers(*int* i, *int* i1, MyStack<*int*> \*pStack);  
  
*int* main() {  
 MyStack<*int*> \*stack = *new* MyStack<*int*>;  
 *bool* q = stack->empty();  
 std::cout << q << '\n';  
 stack->push(1);  
 stack->push(2);  
 stack->push(3);  
 stack->push(4);  
  
 q = stack->empty();  
 std::cout << q << '\n';  
 std::cout << stack->top\_inf() << '\n';  
 stack->pop();  
 std::cout << stack->top\_inf() << '\n';  
 stack->pop();  
 stack->pop();  
 stack->pop();  
 q = stack->empty();  
 std::cout << q << "\n\n\n";  
  
 Multipliers(N, 2, stack);  
 std::cout << N << '=';  
 stack->print();  
 stack->reverse();  
 std::cout << N << '=';  
 stack->print();  
  
 *return* 0;  
}  
  
*void* Multipliers(*int* n, *int* div, MyStack<*int*> \*stack) {  
 *if* (n == 1) *return*;  
 *if* (n % div == 0) {  
 stack->push(div);  
 Multipliers(n / div, div, stack);  
 } *else if* (div == 2)  
 Multipliers(n, div + 1, stack);  
 *else* Multipliers(n, div + 2, stack);  
}

**Анализ результатов**

****

