**Московский государственный технический**

**университет им. Н.Э. Баумана**

Факультет «Информатика и управление»

Кафедра ИУ5. Курс «Программирование на основе классов и шаблонов»

Отчет по лабораторной работе №4

# «Использование шаблонного класса для хранения простых множителей целых чисел»

|  |  |  |
| --- | --- | --- |
| Выполнил: |  | Проверил: |
| студент группы ИУ5-21Б |  | преподаватель каф. ИУ5 |
| Анцифров Никита |  |  |
|  |  |  |
| Подпись и дата: |  | Подпись и дата: |

Москва, 2020 г.

**Постановка задачи**

Дано описание класса MyStack (в Приложении 1 из описания лабораторной работы, файл MyStack.h), который реализует на основе односвязного списка динамическую структуру данных типа стек.

Необходимо:

1. Разработать реализацию интерфейса класса в виде файла MyStack.cpp.
2. Разработать функцию (глобальную), которая выполняет разложение на простые множители целого числа N. Для хранения множителей функция должна использовать класс MyStack.
3. В функции main() распечатать множители, которые функция Multipliers( ) записывает в стек, сначала по убыванию, а потом по возрастанию.

Нельзя изменять описание класса, приведенное в файле MyStack.h.

**Описание входных, выходных и вспомогательных данных**

**MyStack<int>** numbers – шаблонный класс для множителей числа

**int** n – число для выполнения операций

**int** i, k – счетчики цикла

**int\*** arr – указатель на массив для хранения полученных из стека множителей числа (для вывода на экран)

**MyStack<INF>&** s – указатель на стек в функции Multipliers

**Описание прототипов функций**

**void** Multipliers (int, MyStack<INF>&); – функция для записи множителей числа в стек

**Разработка интерфейса класса**

Класс:

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

**class** ListNode – шаблонный класс для узлов списка

Поля класса:

**INF** d – информационная часть узла

**ListNode\*** next – указатель на следующий узел списка

Дружественный класс:

friend **FRIEND**

Класс:

template <class **INF**>

**class** MyStack – шаблонный класс для стека

Поля класса:

**Node\*** top – указатель на вершину в стеке

Методы класса:

**bool** empty(void) – функция для проверки, пустой ли стек

**bool** push(INF) – функция для добавления узла в вершину стека (принимает узел)

**bool** pop(void) – функция для удаления узла из вершины стека

**INF\*** top\_inf(void) – функция для считывания информации из вершины стека (возвращает адрес на информацию)

**Алгоритм**

**main**



**Multipliers**



**Текст программы**

**main.cpp**

#include "MyStack.h"

#include "MyStack.cpp"

#include "proto.h"

#include <iostream>

using namespace std;

template <class INF>

void Multipliers(int n, MyStack<INF>& s)

{

for (int i=2; i<=n; i++)

{

while (n%i==0)

{

n/=i;

s.push(i);

}

}

}

void main()

{

setlocale(LC\_ALL, "Russian");

MyStack<int> numbers;

int n;

cout << "Введите число: ";

cin >> n;

cout << n << " = ";

Multipliers(n, numbers);

int k=0;

do

{

numbers.pop();

k++;

}

while (numbers.empty()==0);

int\* arr=new int[k];

int i=0;

Multipliers(n, numbers);

do

{

arr[i]=\*numbers.top\_inf();

numbers.pop();

i++;

}

while (numbers.empty()==0);

for (int i=0; i<k; i++)

{

cout << arr[i];

if (i!=k-1)

{

cout << " \* ";

}

}

cout << endl;

cout << n << " = ";

for (int i=k-1; i>=0; i--)

{

cout << arr[i];

if (i!=0)

{

cout << " \* ";

}

}

cout << endl;

system("pause");

}

**proto.h**

template <class INF>

void Multipliers(int, MyStack<INF>&);

**MyStack.h**

#ifndef MyStack\_h // защита от повторной компиляции

#define MyStack\_h // модуль подключен

#include <iostream> // чтобы был NULL

//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) // конструктор

{

next=NULL;

}

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(void); // освободить динамическую память

bool empty(void); // стек пустой?

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

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

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

};

#endif

**MyStack.cpp**

#include "MyStack.h"

#include <iostream>

template <class INF>

MyStack<INF>::MyStack(void)

{

top=NULL;

}

template <class INF>

MyStack<INF>::~MyStack(void)

{

delete[] top;

}

template <class INF>

bool MyStack<INF>::empty(void)

{

if (top==NULL)

{

return true;

}

else

{

return false;

}

}

template <class INF>

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

{

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

if (top==NULL)

{

this->top=new Node;

top->d=n;

top->next=NULL;

}

else

{

Node\* p\_unit=new Node;

p\_unit->d=n;

p\_unit->next=top;

top=p\_unit;

}

return true;

}

template <class INF>

bool MyStack<INF>::pop(void)

{

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

Node\* p\_unit=top;

top=top->next;

delete p\_unit;

return true;

}

template <class INF>

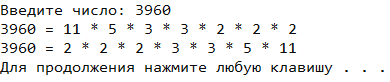
INF\* MyStack<INF>::top\_inf(void)

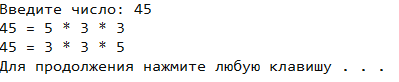
{

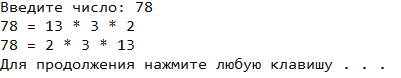
return &top->d;

}

**Анализ результатов**

****

****

****