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 №3

«Шаблоны классов.
Использование шаблонного класса MyStack для хранения простых множителей целых чисел»

Выполнил:		Проверил:
студент группы ИУ5-25Б		преподаватель каф. ИУ5
Петренко Сергей		
Подпись и дата:		Подпись и дата:

г. Москва, 2019 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

Разработка интерфейса класса:

Stack.h

```
template<typename INF>
class Stack
{
public:
  Stack();//конструктор
  ~Stack();//деструктор
  bool isEmpty();// метод проверяющий пустой ли стек
  INF top_inf(); // метод получения данных из вершины
  void pop();//удаление из вершины стека
  void push(INF data);//добавление в вершину стека
  void push_front(INF data); // добавление в начало стека
  int Size;//размер стека
  Node<INF> *head; // узел стека
};
```

Node.h

```
template<typename INF>
class Node {
public:
  Node * pNext; //указатель на следующий узел стека
  INF data; // информация в стеке
public:
  Node(INF data = INF(), Node *pNext = nullptr)
  {
    this->data = data;
    this->pNext = pNext;
  }
};
```

Текст программы:

Stack.cpp

```
template<typename INF>
Stack<INF>::Stack()
{
  Size = 0;
  this->head = nullptr;
}
template<typename INF>
Stack<INF>::~Stack()
{
  while (Size)
  {
     pop();
  }
}
template<typename INF>
void Stack<INF>::pop()
{
  Node<INF> *temp = head;
  head = head->pNext;
  delete temp;
  Size--;
}

template<typename INF>
void Stack<INF>::push(INF data)
{
  Node<INF> *current = new Node<INF>();
  current->data=data;
  current->pNext= this->head;
  this->head=current;
  Size++;
}

template<typename INF>
INF Stack<INF>::top_inf(){
  return this->head->data;
}
template<typename INF>
bool Stack<INF>::isEmpty(){
  return Size==0;
}

template<typename INF>
void Stack<INF>::push_front(INF data)
{
  head = new Node<INF>(data, head);
  Size++;
}
```

Main.cpp

```
#include <iostream>
#include "stack.h"
#include "Node.h"
using namespace std;
template <typename INF>
void multipliers(int n ,Stack<INF> &stack){
      cout << " = ";
      Stack<INF> res;
      for (int i = 2; i <= n; i++) {
         while (n % i == 0) {
            n /= i;
            stack.push(i);
         }
      }
      while (!stack.isEmpty()) {
         cout  << stack.top_inf()<< "*";
         res.push_front(stack.top_inf());
         stack.pop();
      }
      cout << "1\n"<< " = 1" ;
      while (!res.isEmpty()) {
         cout << "*" << res.top_inf();
         res.pop();
      }
      cout << "";
}
int main() {
   Stack<int> stack;
int n;
do{
   cout<<"Введите число \n"<<endl;
   cin >> n;
}while(n<=0);
   multipliers(n, stack);
   return 0;
}
```

Анализ данных: