Министерство науки и высшего образования РФ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Курский государственный университет»

Кафедра программного

обеспечения и администрирования

информационных систем

Направление подготовки

математическое обеспечение и

администрирование

информационных систем

Форма обучения очная

**Отчет**

**по лабораторной работе №6**

«Стеки»

Выполнил:

студент группы 213 Тихонов Е.Е.

Проверил:

профессор кафедры ПОиАИС Кудинов В.А.

Курск, 2022

**Цель работы**: Научиться составлять процедуры (функции), реализующие операции для работы со стеками.

**Задание**

1) Напишите процедуру формирования стека, причем учтите, что стек может содержать не более K звеньев (переполнение стека) и не может быть пустой (опустошение стека) (1 балл);

2) Дан файл, элементами которого являются целые числа, упорядоченные по возрастанию (убыванию). При помощи структуры данных стек провести "обратную" сортировку файла по убыванию (возрастанию). (3 балла);

3) Дан текстовый файл T. Напечатать его содержимое, выписывая литеры каждой его строки в обратном порядке (6 баллов).

4) Пусть построен стек, содержащий целые числа. Вычислить количество простых чисел, содержащихся в стеке. (2 балл).

**Код программы**

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

template <typename DataType>

class Node;

template <typename T>

class Stack

{

public:

friend class Node<T>;

Node<T>\* First;

Node<T>\* Last;

int Count = 0;

int MaxStackLength = 0;

Stack(T FirstElement, int MaxQueueLength)

{

this->First = new Node<T>(FirstElement);

this->Last = First;

this->MaxStackLength = MaxQueueLength;

Count++;

}

void PushBack(T Element)

{

if (Count < MaxStackLength)

{

Node<T>\* temp = new Node<T>(Element);

Last->Next = temp;

temp->Prev = Last;

Last = temp;

Count++;

}

else

{

cout << "переполнение стека" << endl;

}

}

T PopBack()

{

Node<T>\* temp = Last;

T value = temp->Data;

if (Count > 1)

{

Last = temp->Prev;

Last->Next = nullptr;

delete temp;

Count--;

}

else

{

cout << "опустошение стека" << endl;

}

return value;

}

string ToString()

{

string str1 = "";

Node<T>\* Elem = this->First;

while (Elem != nullptr)

{

str1 += to\_string(Elem->Data);

str1 += " ";

Elem = Elem->Next;

}

return str1;

}

};

template <typename DataType>

class Node

{

friend class Stack<DataType>;

DataType Data;

Node\* Next;

Node\* Prev;

Node(DataType Data = 0)

{

this->Data = Data;

this->Next = nullptr;

this->Prev = nullptr;

}

};

int main()

{

setlocale(LC\_ALL, "Russian");

//2

string Path1 = "C:/Users/Evgeniy/Desktop/Code/Algoritms/task6/numbers1.txt";

string Path2 = "C:\\Users\\Evgeniy\\Desktop\\Code\\Algoritms\\task6\\numbers2.txt";

ifstream Reader(Path1);

Stack<string>\* NumsStack = new Stack<string>("", 100);

string Num;

while (Reader >> Num)

{

NumsStack->PushBack(Num);

}

ofstream Writer(Path2);

int count = NumsStack->Count;

for (size\_t i = 0; i < count; i++)

{

string num = NumsStack->PopBack();

Writer << num << " ";

}

//4

Path1 = "C:/Users/Evgeniy/Desktop/Code/Algoritms/task6/text1.txt";

Path2 = "C:/Users/Evgeniy/Desktop/Code/Algoritms/task6/text2.txt";

ifstream Reader2(Path1);

Stack<string>\* TextStack = new Stack<string>("", 100);

string Word;

while (Reader2 >> Word)

{

TextStack->PushBack(Word);

}

ofstream Writer2(Path2);

count = TextStack->Count;

for (size\_t i = 0; i < count; i++)

{

Writer2 << TextStack->PopBack() + "\n";

}

//2 подсчет

Stack<int>\* stack1 = new Stack<int>(1, 100);

stack1->PushBack(3);

stack1->PushBack(5);

stack1->PushBack(7);

stack1->PushBack(9);

stack1->PushBack(11);

stack1->PushBack(13);

stack1->PushBack(15);

stack1->PushBack(17);

stack1->PushBack(19);

stack1->PushBack(21);

stack1->PushBack(23);

stack1->PushBack(25);

cout << "q1: " << stack1->ToString() << endl;

int SimpleNumsCount = 0;

count = stack1->Count;

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

{

int elem = stack1->PopBack();

bool IsSimple = true;

for (int j = 2; j < elem/2; j++)

{

if (elem % j == 0)

{

IsSimple = false;

}

}

if (IsSimple && elem != 1)

{

SimpleNumsCount++;

}

}

cout << "количество простых чисел: " << SimpleNumsCount << endl;

return 0;

}

**Тест программы**

Тест программы представлен на рисунке 1. Содержимое файлов, с которыми работает программа, представлено на рисунках 2-3.

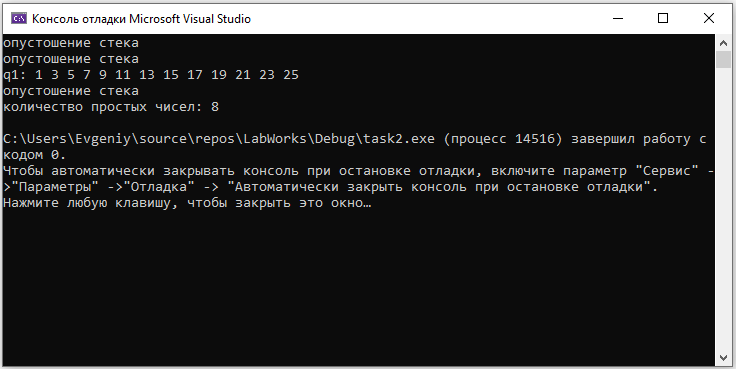


Рисунок 1 - Тест программы 1

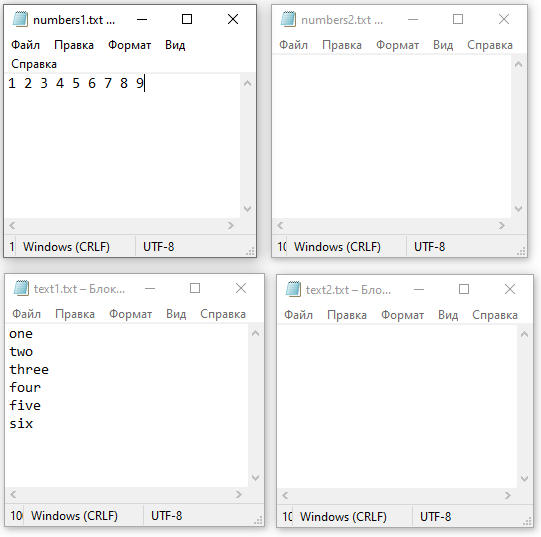


Рисунок 2 – Содержимое файлов до запуска программы

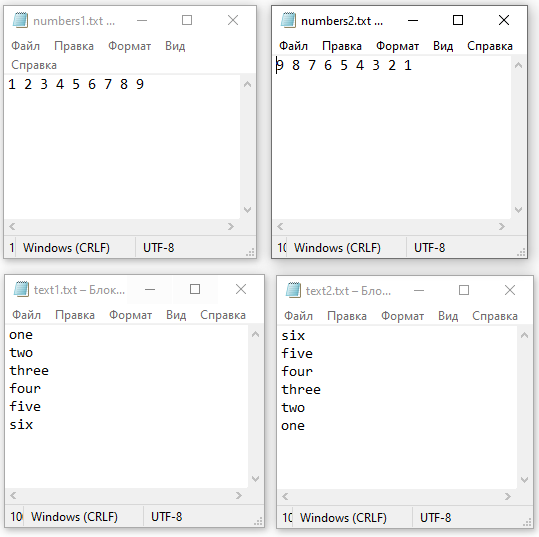


Рисунок 3 - Содержимое файлов после запуска программы