**ФГБОУ ВО**

**Национальный исследовательский университет**

**«МЭИ»**

**Лабораторная работа №2**

**Динамические структуры данных**

**Вариант 8**

**Задание №1:**

**Преобразование структур**

**Выполнил:**

студент группы Аэ-21-23

Нестеров А.С.

**Москва, 2024**

1. **Постановка задачи**

Задание:

1. Формирование (стек / очередь)
2. Добавление (в стек/ очередь)
3. Удаление (из стека/ очереди)
4. Очистка стека
5. Вывод (элементов стека/ очереди)
6. Выход

|  |  |  |
| --- | --- | --- |
| 2 | Real | Стек — > очередь |

Входные данные:

* n(int) – количество элементов списка [1;10]
* Элементы списка (double) – ограничение (8 байт)

1. **Разработка приложения** 
   1. Разработка структуры приложения

Задание реализовано в виде консольного приложения на языке С++ с использованием процедур и функций.

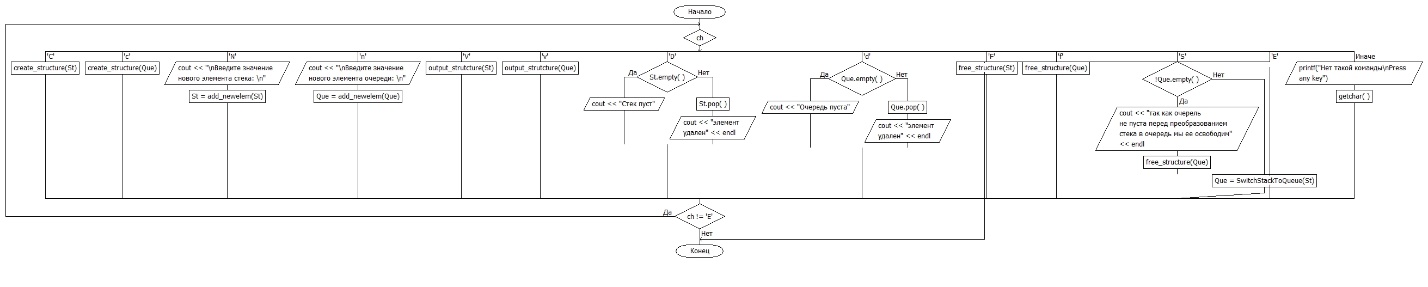


Рисунок 1 – Структура приложения

Таблица 1 - Спецификация функций

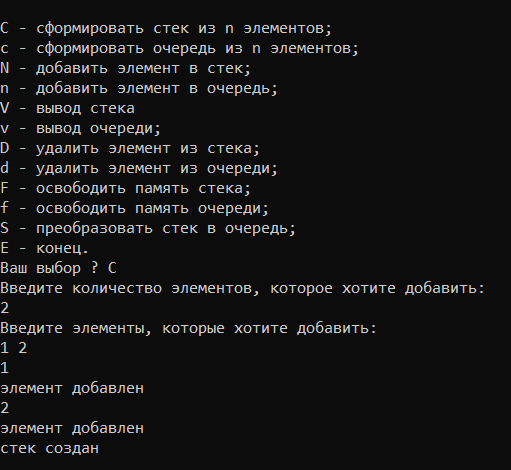
|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название | Назначение | Вх. данные | Вых. данные |
| 1 | create\_structure | создание списка из n элементов | Список  (количество элементов и сами элементы функция запрашивает сама) | список |
| 2 | add\_newelem(St) | Добавление (в стек/ очередь) | Стек/очередь | Стек/очередь |
| 3 | output\_strutcture(St); |  | Стек/очередь | Стек/очередь |
| 4 | St.pop(); | Удаление (из стека/ очереди) | Стек/очередь | Стек/очередь |
| 5 | free\_structure(St); | Очистка стека/очереди | Стек/очередь | Стек/очередь |
| 6 | Que = SwitchStackToQueue(St); | Стек — > очередь | Стек | Очередь |

2.3 Разработка пользовательского интерфейса

Категория пользователей: Студенты

Интерфейс представляет собой консольное приложение, в котором взаимодействие пользователя с программой осуществляется с помощью диалогового окна с выбором опций.

Пример диалогов с пользователем:



1. **Реализация и тестирование приложения**

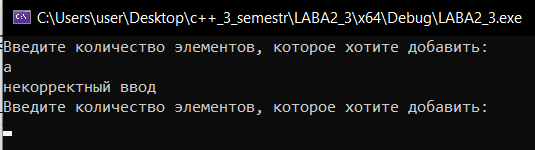
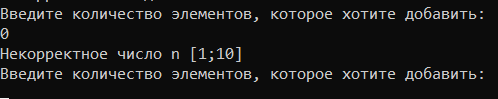
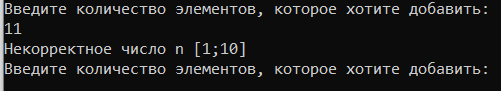
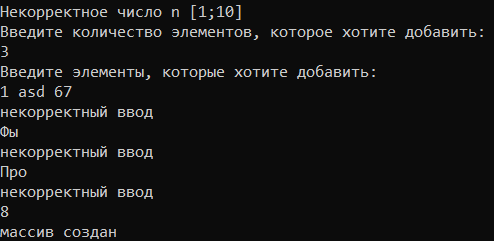
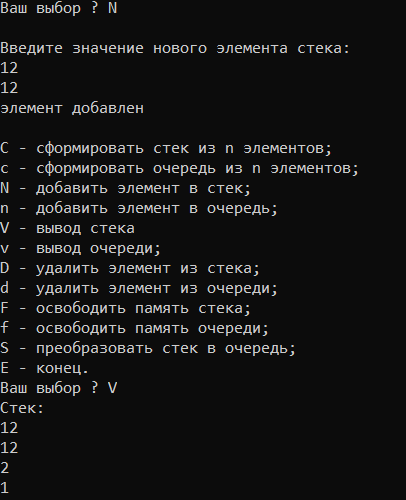
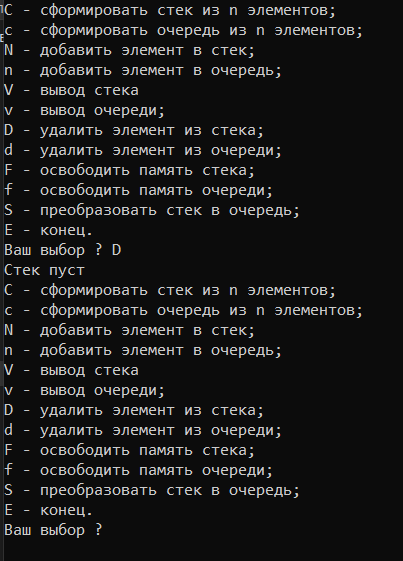
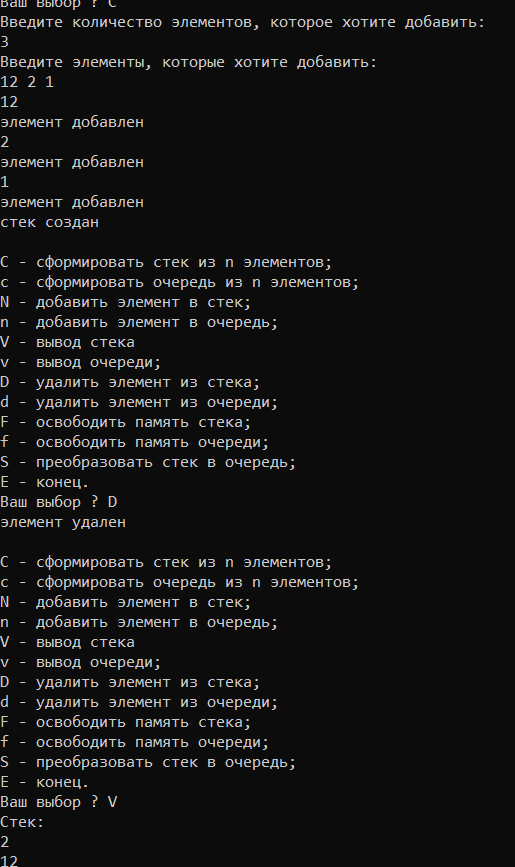
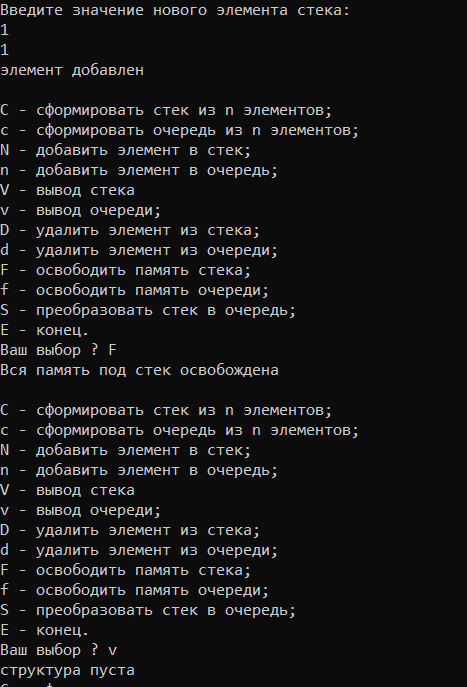
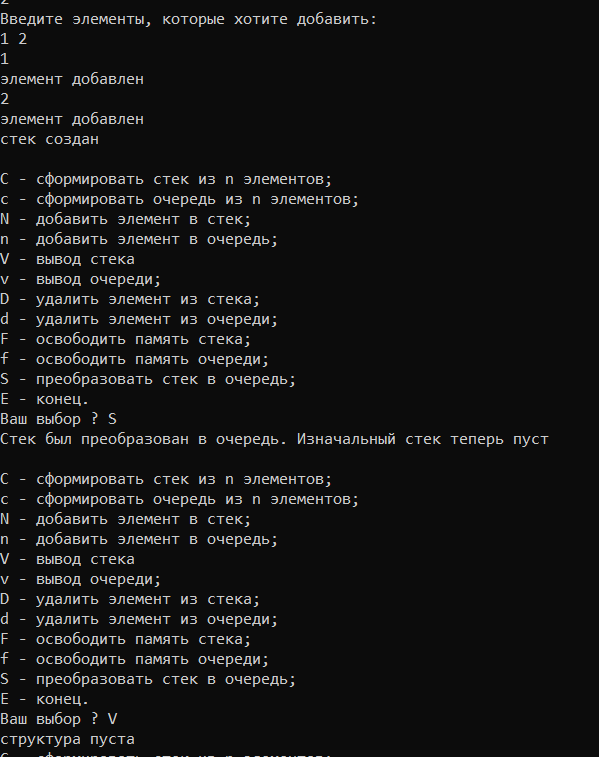
3.1 Описание разработанной программы

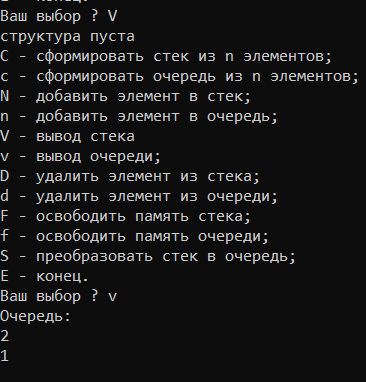
Данная программа позволяет пользователю создать собственную очередь, содержащую элементы типа char, добавить элементы в стек/очередь, очистить стек, просмотреть стек/очередь, преобразовать стек в очередь и завершить работу программы в режиме диалога.

3.2 Тестирование программы

Таблица 2 – Тесты программы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Тестируемая функция | Входные данные | Ожидаемый результат | Цель теста |
| 1 | create\_structure | N = a | некорректный ввод  Введите количество элементов, которое хотите добавить: | Некорректное число n |
| 2 | create\_structure | N=0 | Некорректное число n [1;10]  Введите количество элементов, которое хотите добавить: | Некорректное число n |
| 3 | create\_structure | N = 11 | Некорректное число n [1;10]  Введите количество элементов, которое хотите добавить: | Некорректное число n |
| 4 | create\_structure | N = 3  1 asd 67  Фы  Про  8 | Введите элементы, которые хотите добавить:  1 asd 67  некорректный ввод (запрос ввести заново)  фы  некорректный ввод  (запрос ввести заново)  про  некорректный ввод  (запрос ввести заново)  8  массив создан  Ваш массив:  1 67 8 | Некорректный элемент |
| 5 | add\_newelem(St) | 12 2 1  Добавить 12 | 12 12 2 1 | работа |
| 6 | St.pop(); | Пустой | Пустой | Невозможно |
| 7 | St.pop() | 12 2 1 | 2 1 | работа |
| 8 | free\_structure(St); | 1 2 3 | Пустой | Пустой |
| 9 | Que = SwitchStackToQueue(St); |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

1. 
2. 
3. 
4. 
5. 
6. 
7. 
8. 
9. 



1. **Код**

#include <iostream>

#include "Modul1.h"

using namespace std;

stack <double> add\_newelem(stack <double>& St)

{

string newelem\_string;

cin.ignore(0, '\n') >> newelem\_string;

//cout << newelem<<endl;

try {

double newelem = stod(newelem\_string);

cout << newelem << endl;

try

{

St.push(newelem);

cout << "элемент добавлен" << endl;

}

catch (...)

{

cout << "не удалось добавить элемент в структуру";

}

}

catch (...)

{

cout << "некорректный ввод" << endl;

St = add\_newelem(St);

}

return St;

}

queue <double> add\_newelem(queue <double>& Que)

{

string newelem\_string;

cin.ignore(0, '\n') >> newelem\_string;

//cout << newelem<<endl;

try {

double newelem = stod(newelem\_string);

cout << newelem << endl;

try

{

Que.push(newelem);

cout << "элемент добавлен в очередь" << endl;

}

catch (...)

{

cout << "не удалось добавить элемент в очередь";

}

}

catch (...)

{

cout << "некорректный ввод" << endl;

Que = add\_newelem(Que);

}

return Que;

}

void create\_structure(stack <double>& St)

{

if (!St.empty()) cout << "изначальный стек не пустой, но мы добавим n элементов в него"<<endl;

cout << "Введите количество элементов, которое хотите добавить: " << endl;

string n\_string;

cin.ignore(0, '\n') >> n\_string;

try {

int n = stoi(n\_string);

if (n < 1 or n >10)

{

cout << "Некорректное число n [1;10]" << endl;

cin.clear();

create\_structure(St);

return;

}

cout << "Введите элементы, которые хотите добавить: " << endl;

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

{

St = add\_newelem(St);

}

cout << "стек создан" << endl;

}

catch (...)

{

cout << "некорректный ввод";

create\_structure(St);

return;

}

return;

/\*int n; cin.ignore(0, '\n') >> n;

if (n < 1 or n >10)

{

cout << "Некорректное число n [1;10]" << endl;

create\_structure(St);

return;

}

cout << "Введите элементы, которые хотите добавить: " << endl;

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

{

St = add\_newelem(St);

}

cout << "стек создан" << endl;\*/

}

void create\_structure(queue <double>& Que)

{

if (!Que.empty()) cout << "изначальная очередь не пустая, но мы добавим n элементов в неё" << endl;

cout << "Введите количество элементов, которое хотите добавить: " << endl;

string n\_string;

cin.ignore(0, '\n') >> n\_string;

try {

int n = stoi(n\_string);

if (n < 1 or n >10)

{

cout << "Некорректное число n [1;10]" << endl;

cin.clear();

create\_structure(Que);

return;

}

cout << "Введите элементы, которые хотите добавить: " << endl;

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

{

Que = add\_newelem(Que);

}

cout << "очередь создана" << endl;

}

catch (...)

{

cout << "некорректный ввод";

create\_structure(Que);

return;

}

return;

/\*int n; cin.ignore(0, '\n') >> n;

if (n < 1 or n >10)

{

cout << "Некорректное число n [1;10]" << endl;

create\_structure(Que);

return;

}

cout << "Введите элементы, которые хотите добавить: " << endl;

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

{

Que = add\_newelem(Que);

}

cout << "стек создан" << endl;\*/

}

void output\_strutcture(stack <double> St) {

if (St.empty()) { cout << "структура пуста"; return; }

else

{

cout << "Стек: " << endl;

stack<char> TempSt;

while (!St.empty()) {

cout << St.top() << endl;

TempSt.push(St.top());

St.pop();

}

while (!TempSt.empty()) {

St.push(TempSt.top());

TempSt.pop();

}

}

return;

}

void output\_strutcture(queue <double> Que) {

if (Que.empty()) { cout << "структура пуста"; return; }

else

{

cout << "Очередь: " << endl;

queue <char> TempQue;

while (!Que.empty()) {

cout << Que.front() << endl;

TempQue.push(Que.front());

Que.pop();

}

while (!TempQue.empty()) {

Que.push(TempQue.front());

TempQue.pop();

}

cout << endl;

}

return;

}

void free\_structure(stack <double> St)

{

if (St.empty()) {

cout << "Стек пуст"; return;

}

else

{

while (!St.empty()) {

St.pop();

}

cout << "Вся память под стек освобождена" << endl;

}

return;

}

void free\_structure(queue <double> Que)

{

if (Que.empty()) {

cout << "Очередь пуста"; return;

}

else

{

while (!Que.empty()) {

Que.pop();

}

cout << "Вся память под очередь освобождена" << endl;

}

return;

}

//queue<char> stack\_to\_queue(stack<char> st, queue<char> que) {

// if (!st.empty()) {

// while (!st.empty()) {

// que.push(st.top());

// st.pop();

// }

// cout << "Стек преобразован в очередь" << endl;

// }

// else {

// cout << "Стек пуст" << endl;

// }

// return que;

//}

queue <double> SwitchStackToQueue(stack <double>& St)

{

queue <double> Que;

if (St.empty())

{

cout << "Стек пуст" << endl;

}

else

{

while (!St.empty())

{

Que.push(St.top());

St.pop();

}

cout << "Стек был преобразован в очередь. Изначальный стек теперь пуст" << endl;

}

return Que;

}

void main()

{

char ch;

string newelem\_string;

stack <double> St;

queue <double> Que;

//stack <double> St;

SetConsoleOutputCP(1251);

SetConsoleCP(1251);

do {

printf("\nC - сформировать стек из n элементов;\nс - сформировать очередь из n элементов;\nN - добавить элемент в стек;\nn - добавить элемент в очередь;\nV - вывод стека\nv - вывод очереди;\nD - удалить элемент из стека;\nd - удалить элемент из очереди;\nF - освободить память стека;\nf - освободить память очереди;\nS - преобразовать стек в очередь;\nE - конец.\nВаш выбор ? ");

ch = getchar(); fflush(stdin);

//ch = toupper(ch);

switch (ch) {

case 'C':

create\_structure(St);

break;

case 'c':

create\_structure(Que);

break;

case 'N':

cout << "\nВведите значение нового элемента стека: \n";

St = add\_newelem(St);

break;

case 'n':

cout << "\nВведите значение нового элемента очереди: \n";

Que = add\_newelem(Que);

break;

case 'V':

//printf("Содержимое структуры :\n"); OutputStructure(StackTop);

//cout << "содержимое стека: "<<endl;

output\_strutcture(St);

break;

case 'v':

//cout << "содержимое очереди: " << endl;

output\_strutcture(Que);

break;

case 'D':

if (St.empty()) cout << "Стек пуст";

else

{

St.pop();

cout << "элемент удален" << endl;

}

break;

case 'd':

if (Que.empty()) cout << "Очередь пуста";

else

{

Que.pop();

cout << "элемент удален" << endl;

}

break;

case 'F':

free\_structure(St);

break;

case 'f':

free\_structure(Que);

break;

case 'S':

if (!Que.empty())

{

cout << "так как очерель не пуста перед преобразованием стека в очередь мы ее освободим" << endl;

free\_structure(Que);

}

Que = SwitchStackToQueue(St);

//printf("\nСтек преобразован в очередь\n");

break;

case 'E': return;

default:

fflush(stdin);

printf("Нет такой команды\nPress any key"); getchar();

}

fflush(stdin);

ch = getchar();

} while (ch != 'E');

return;

}