Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

**«Пермский национальный исследовательский**

**политехнический университет»**

Кафедра «Информационные технологии и автоматизированные системы»

**О Т Ч Ё Т**

**по лабораторной работе №11.3**

Дисциплина: основы алгоритмизации и программирования

Тема: “Списки, очереди, стеки"

**Вар.21**

Выполнил работу

студент группы ИВТ-20-2Б

Галинов О.Ю.

Проверила:

Доцент кафедры ИТАС

Полякова О.А.

**Пермь, 2021**

**Постановка задачи**

1. Сформировать однонаправленный и двунаправленный списки или стек и очередь. Тип информационного поля указан в варианте. (Тип информационного поля char\*.Добавить в стек элемент после элемента с заданным информационным полем.).

2. Распечатать полученную структуру.

3. Выполнить обработку структуры в соответствии с заданием.

4. Распечатать полученный результат.

**Анализ задачи**

1. *Используемые типы данных.*

В программе используются: структура Stack, char, stack <char>.

**1.1.** Структура Stack используется для хранения двух элементов типа int.

Stack\* last, \* p;

last = NULL;

p = new Stack;

**1.2.** Тип int используется для хранения данных, как параметр в цикле, для временного хранения размера динамической структуры.

Int size;

**1.3.** Тип char используется для хранения данных внутри структуры.

char elem;

cout << "Введите 1 элемент:\t";

cin >> elem;

* 1. Тип stack<char>используется для хранения стека через STL.

1. *Действия над используемыми данными.*

**2.1.** С данными типа int производятся следующие действия: математические операции (инкремент в циклах с параметром).

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

**2.2.** С данными типа Stack производятся следующие действия: создание стека, добавление элементов в стек, вывод на экран, удаление последнего элемента стека.

**2.3.** Данные типа char используются только для хранения, и над ними не производятся какие-либо действия.

**2.4.** С данными типа stack<char>производятся те же действия (создание стека, добавление элементов в стек, вывод на экран, удаление последнего элемента стека), что и с типом Stack, но они реализованы с использованием STL.

1. *Вид данных.*

**3.1.** Стек реализован в программе с помощью структуры и STL.

1. *Структура.*

**4.1.** В программе используется структура Stack, которая реализует стек. Поле data имеет тип cahr и предназначено для хранения данных. Поля next и prev имеют тип Stack \* и предназначено для хранения указателя на следующий элемент списка.

struct Stack

{

char data;

Stack\* prev, \*next;

};

1. *Ввод и вывод.*

**5.1.** Ввод и вывод всех данных осуществляется в консоль через операторы cin и cout соответственно.

cout << "Введите " << i << " элемент:\t";

cin >> p->data;

**5.2.** Для ввода и вывода структур реализованы функции Stack\* createStack(int size), void PrintStack(Stack\*& last), stack<char> CreateStackSTL(int size), void PrintSTL(stack<char>st). Функции с символами STL в названии предназначены для работы со стеком, реализованным через STL. Остальные предназначены для работы со стеком, реализованным через структуры.

Stack\* createStack(int size)

{

if (size == 0) return 0;

Stack\* last, \* p;

last = NULL;

p = new Stack;

cout << "Введите 1 элемент:\t";

cin >> elem;

p->data = elem;

p->next = NULL;

last = p;

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

{

Stack\* h = new Stack;

p->next = h;

p = p->next;

cout << "Введите " << i << " элемент:\t";

cin >> p->data;

p->next = NULL;

}

return last;

}

void PrintStack(Stack\*& last)

{

if (last == NULL)

cout << "Стек пуст" << endl;

else {

Stack\* p = last;

while (p != NULL)

{

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

}

stack<char> CreateStackSTL(int size)

{

char element;

stack<char> res;

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

{

cout << "Введите " << i + 1 << " элемент: ";

cin >> element;

res.push(element);

}

return res;

}

void PrintSTL(stack<char>st)

{

stack<char> t;

while (!st.empty())

{

t.push(st.top());

st.pop();

}

while (!t.empty())

{

cout << t.top() << " ";

t.pop();

}

cout << endl;

}

1. *Действия для решения задачи.*

**6.1.** Добавление элементов в стек через STL.

Передвижение по стеку происходит через итератор, который увеличивается на каждой итерации цикла.

void AddElementSTL(stack<char>st)

{

stack<char>copy;

char key; int number;

cout << "Введите номер, после которого хотите добавить элемент: ";

cin >> number;

while (number <1 || number > st.size())

{

cout << "Введите номер корректно: ";

cin >> number;

}

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

{

copy.push(st.top());

st.pop();

}

cout << endl;

cout << "Введите новый элемент: ";

for (int i = 0; i < 1; i++)

{

cin >> key;

st.push(key);

}

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

{

st.push(copy.top());

copy.pop();

}

PrintSTL(st);

}

**6.2.** Добавление элементов в стек, реализованный через структуру.

Описывается в отдельной функции void Addelement , которая использует дву другие описанные ранее функции для добавления элемента в начало очереди (PushST) и удаления из начала (PopST).

int PopST(Stack\*& last)

{

Stack\* p = last;

int k = 0;

while (p != NULL)

{

k++;

p = p->next;

}

p = last;

if (k == 1)

{

int tmp = last->data;

delete p;

last = NULL;

return tmp;

}

else

{

Stack\* t = p->next;

int u = p->data;

last = t;

delete p;

return u;

}

}

Stack\* PushST(Stack\*& last, int el)

{

Stack\* p = new Stack;

p->data = el;

p->next = last;

last = p;

return last;

}

void Addelement(int size, Stack\*& last)

{

Stack\* mystack2 = createStack(0);

char key;

int number;

do {

cout << "Введите номер, после которого хотите добавить элемент: ";

cin >> number;

} while (number > size || number < 1);

for (int i = 0; i < size - number; i++)

{

char t = PopST(last);

PushST(mystack2, t);

}

for (int i = 0; i < 1; i++)

{

cout << "Введите новый элемент: ";

cin >> key;

PushST(last, key);

}

for (int i = 0; i < size - number; i++)

{

char t = PopST(mystack2);

PushST(last, t);

}

}

**Код программы**

#include <iostream>

#include<stack>

#include<vector>

using namespace std;

char elem;

//without STL-functions

struct Stack

{

char data;

Stack\* prev, \*next;

};

Stack\* createStack(int size)

{

if (size == 0) return 0;

Stack\* last, \* p;

last = NULL;

p = new Stack;

cout << "Введите 1 элемент:\t";

cin >> elem;

p->data = elem;

p->next = NULL;

last = p;

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

{

Stack\* h = new Stack;

p->next = h;

p = p->next;

cout << "Введите " << i << " элемент:\t";

cin >> p->data;

p->next = NULL;

}

return last;

}

void PrintStack(Stack\*& last)

{

if (last == NULL)

cout << "Стек пуст" << endl;

else {

Stack\* p = last;

while (p != NULL)

{

cout << p->data << " ";

p = p->next;

}

cout << endl;

}

}

int PopST(Stack\*& last)

{

Stack\* p = last;

int k = 0;

while (p != NULL)

{

k++;

p = p->next;

}

p = last;

if (k == 1)

{

int tmp = last->data;

delete p;

last = NULL;

return tmp;

}

else

{

Stack\* t = p->next;

int u = p->data;

last = t;

delete p;

return u;

}

}

Stack\* PushST(Stack\*& last, int el)

{

Stack\* p = new Stack;

p->data = el;

p->next = last;

last = p;

return last;

}

void Addelement(int size, Stack\*& last)

{

Stack\* mystack2 = createStack(0);

char key;

int number;

do {

cout << "Введите номер, после которого хотите добавить элемент: ";

cin >> number;

} while (number > size || number < 1);

for (int i = 0; i < size - number; i++)

{

char t = PopST(last);

PushST(mystack2, t);

}

for (int i = 0; i < 1; i++)

{

cout << "Введите новый элемент: ";

cin >> key;

PushST(last, key);

}

for (int i = 0; i < size - number; i++)

{

char t = PopST(mystack2);

PushST(last, t);

}

}

//STL-functions

stack<char> CreateStackSTL(int size)

{

char element;

stack<char> res;

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

{

cout << "Введите " << i + 1 << " элемент: ";

cin >> element;

res.push(element);

}

return res;

}

void PrintSTL(stack<char>st)

{

stack<char> t;

while (!st.empty())

{

t.push(st.top());

st.pop();

}

while (!t.empty())

{

cout << t.top() << " ";

t.pop();

}

cout << endl;

}

void AddElementSTL(stack<char>st)

{

stack<char>copy;

char key; int number;

cout << "Введите номер, после которого хотите добавить элемент: ";

cin >> number;

while (number <1 || number > st.size())

{

cout << "Введите номер корректно: ";

cin >> number;

}

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

{

copy.push(st.top());

st.pop();

}

cout << endl;

cout << "Введите новый элемент: ";

for (int i = 0; i < 1; i++)

{

cin >> key;

st.push(key);

}

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

{

st.push(copy.top());

copy.pop();

}

PrintSTL(st);

}

int GetSize()

{

int n;

cin >> n;

if (n < 1)

{

cout << "Введите корректное число: ";

cin >> n;

}

return n;

}

int main()

{

setlocale(LC\_ALL, "rus");

//without stl

int size;

cout << "Введите размер стека: ";

size = GetSize();

Stack\* mystack = createStack(size);

cout << "Ваш стек: ";

PrintStack(mystack);

Addelement(size, mystack);

cout << "Ваш стек с добавленным элементом:" << endl;

PrintStack(mystack);

//stl

int stlsize;

cout << "Введите размер стека: ";

stlsize = GetSize();

stack<char>stackstl = CreateStackSTL(stlsize);

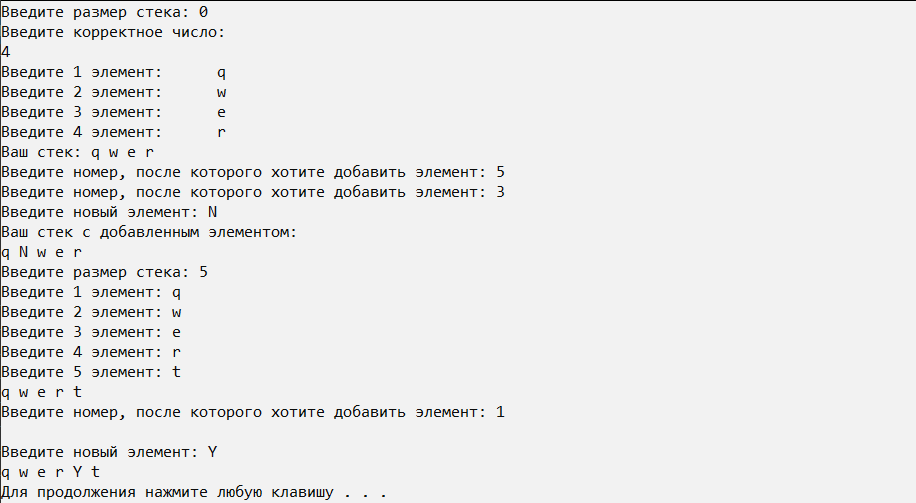
PrintSTL(stackstl);

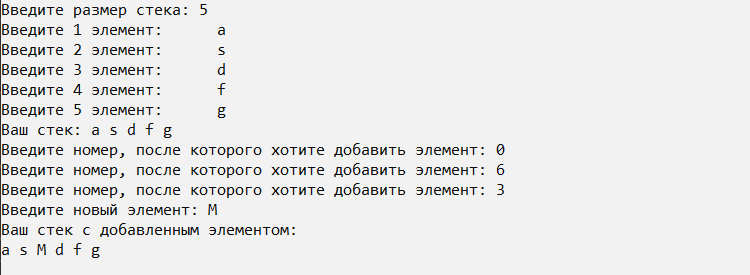
AddElementSTL(stackstl);

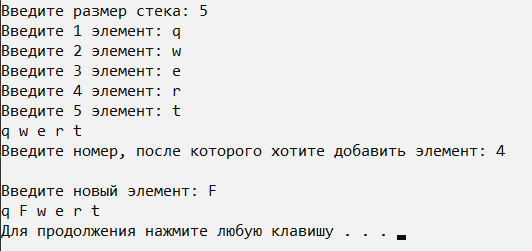
system("pause");

}

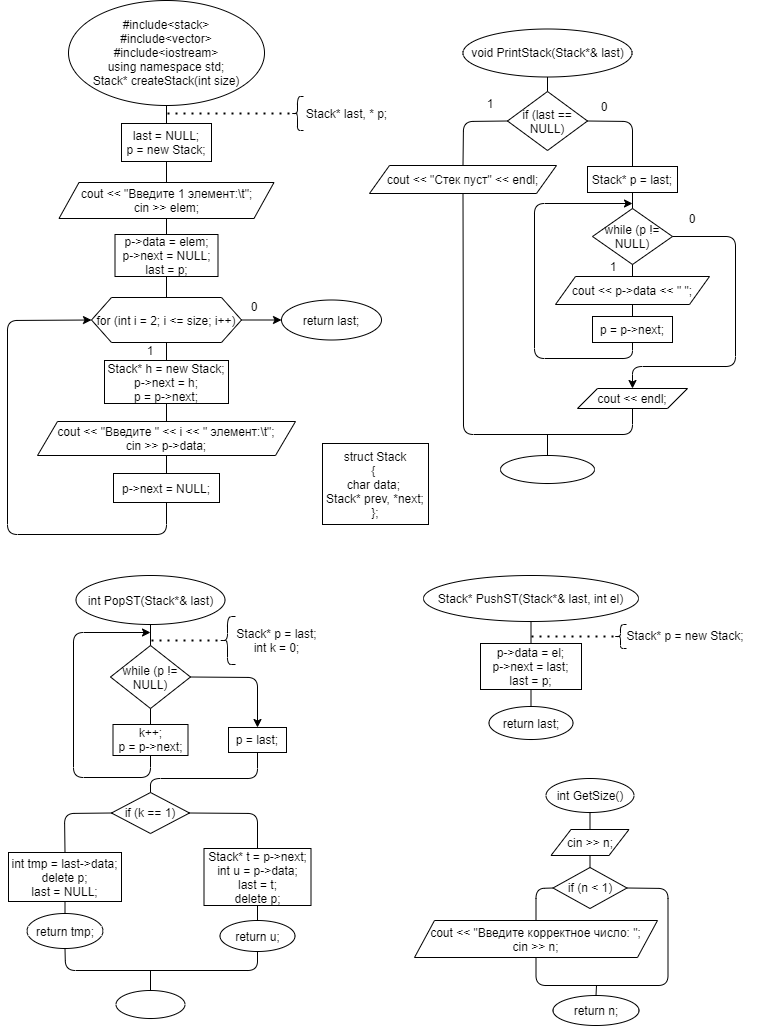
**Скриншоты**

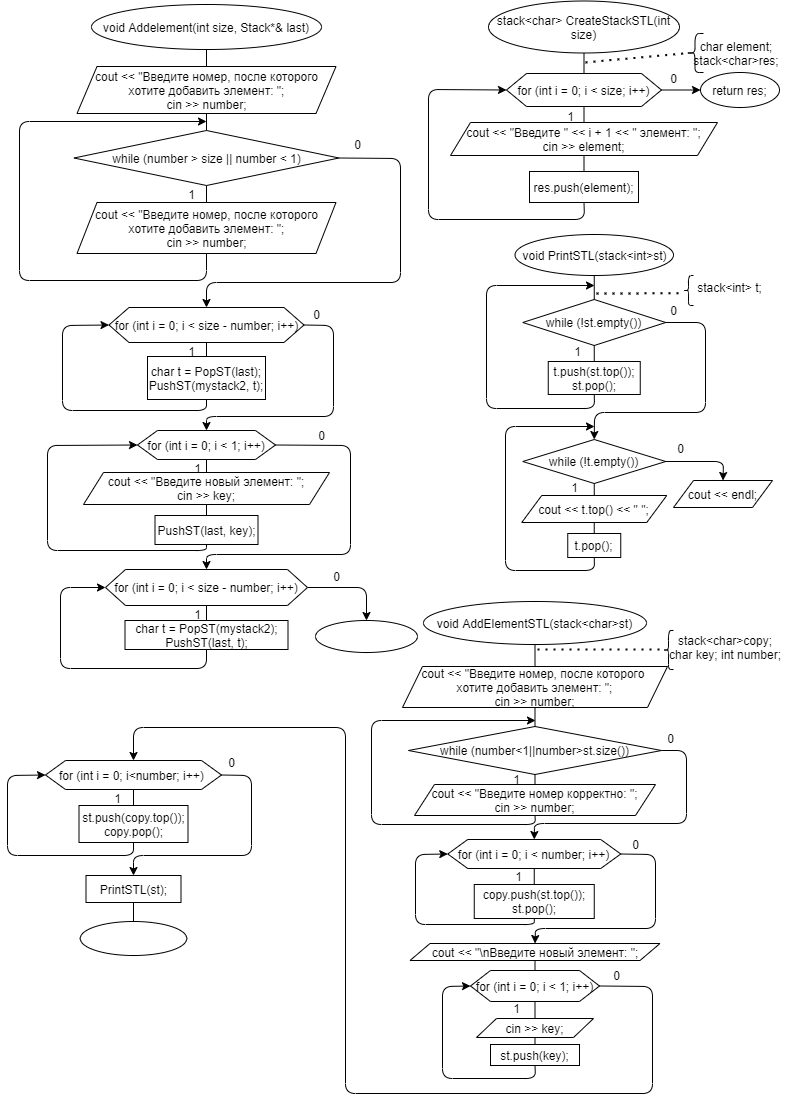
****

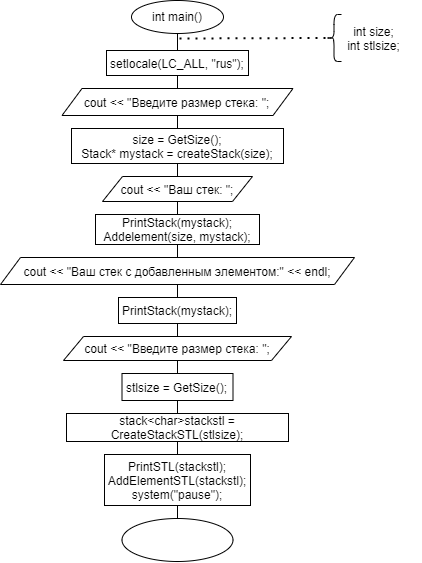
****

****

**Блок-схема**

****

****

****