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**

Дисциплина: основы алгоритмизации и программирования

Тема: “ Динамические структуры данных.

Стеки”

Вариант 16

Выполнил:

студент группы ИВТ-20-2Б Тедеев А.З.

Проверила: доцент кафедры ИТАС

Полякова О.А.

Пермь, 2021

**Цель задачи**

Получить практические навыки работы со стеками.

**Постановка задачи**

Тип информационного поля char\*. Добавить в стек элементы с номерами 1, 3, 5 и т. д.

2

**Анализ задачи**

**1.** Для решения задачи необходимо:

**1.1.** Организовать структуру Stack, поля которой будут использоваться для ввода данных типа char, для хранения адреса следующего элемента стека.

**1.2.** Организовать функцию MakeStack() для формирования стека.

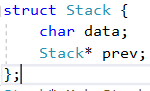
**1.3.** Организовать функцию Addition () для добавления элемента.

**1.4.** Организовать функцию GetStack() для вывода стека на консоль.

**1.5.** Организовать функцию GetStack2() для вывода стека на консоль.

**2.** В ходе работы были использованы следующие типы данных:

**2.1.** Структура Stack для организации стека.



**2.2.** Функция MakeStack(), в качестве параметра передаётся размер стека size. В переменную x записать первый элемент стека и записать его в стек через метод data.

**2.3.** Функция Additon(), в качестве параметра передаётся указатель на ссылку на первый элемент стека – top. Также в переменную x вводится элемент, который надо добавить в стек.

Stack\* Addition(Stack\*& top)

{

char x;

int i = 0;

cout << "Введите элемент для вставки: ";

cin >> x;

Stack\* p = top;

...

}

**2.4.** Функцию GetStack() для вывода введённого стека в консоль для вывода стека, которая принимает в качестве параметра последний введённый элемент - top.

3

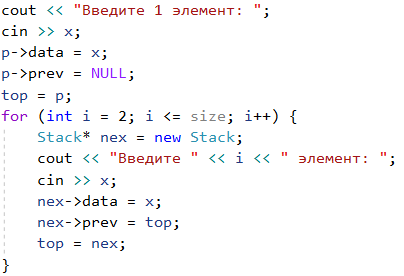
**2.5.** Функцию GetStack2() для выовода изменённого стека на консоль, в качестве параметров передаём размер стека – size, последний элемент стека – top.

**3.** Для решения задачи данные были представлены в следующем виде:

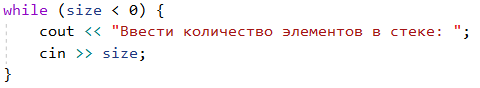
**3.1.** Данные вводятся через консоль в виде очереди через структуру Stack с полем типа char и указателем на предыдущий элемент стека из-за особенности стека.

**4.** Для операций ввода и вывода использовались следующие операторы и функции:

**4.1.** Ввод данных на консоль реализован функцией cin для первого элемента стека и в цикле for (также через cin) для следующих.



**4.2.** В функции main() вводится количество элементов в стеке в переменной size через цикл while, чтобы пользователь не ввёл отрицательное число.

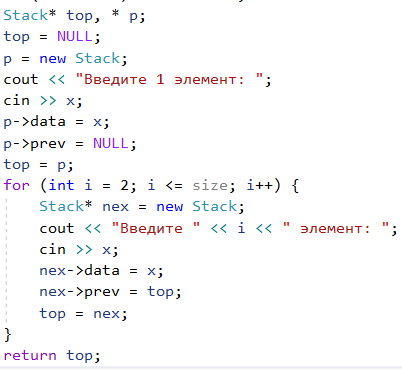


**5.** Поставленные задачи будут решены следующими действиями:

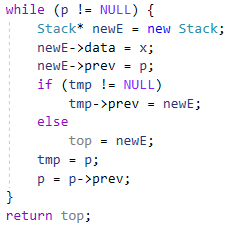
**5.1.** В функции MakeStack() осуществляется ввод всех элементов в стеке. Первый вводится вне цикла for и используется в качестве указателя на стек. В цикле for заполняются элементы от 2 до size: резервируется память в переменной nex, в которой через метод data вводим следующий элемент.

4

Далее через метод prev для переменной nex записываем предыдущий элемент – top, затем присваиваем в top введённый элемент.



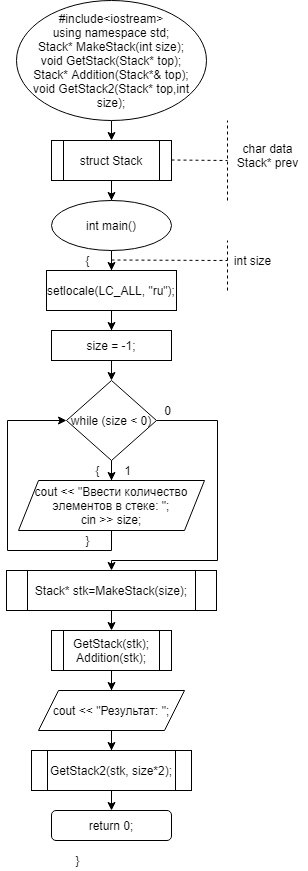
**5.2.** В функции Addition() организовать цикл, который итерируется по стеку, в котором переменная p указывает на последний элемент в стеке (первый в обработке). Резервируется память под новый элемент x и записывается через метод prev в newE, затем в ветвлении проверяется наличие предыдущего элемента: если такового нет (при рассмотрении последнего элемента), то в переменную top записывается новый элемент, иначе через метод prev в переменную tmp запоминается введённый элемент. После ветвления в tmp сохраняется выбранный вначале элемент и в переменную p присваивается следующий элемент.



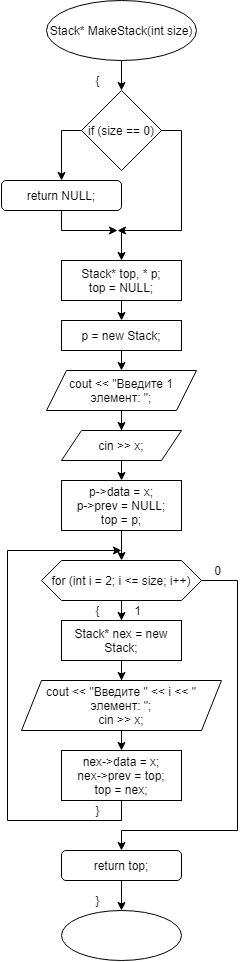
5

**Блок-схема**

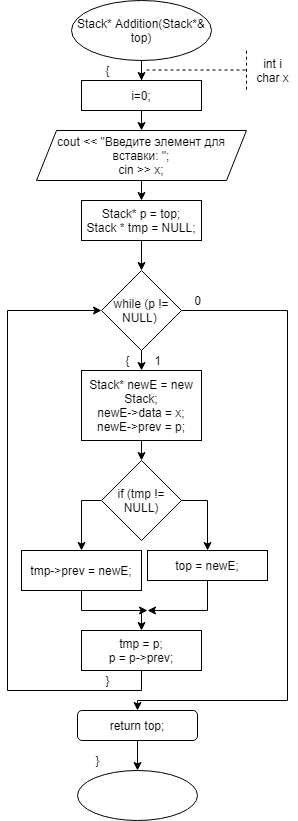
**Реализация через структуру**



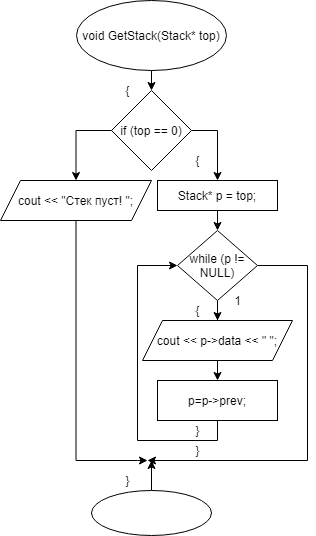
6



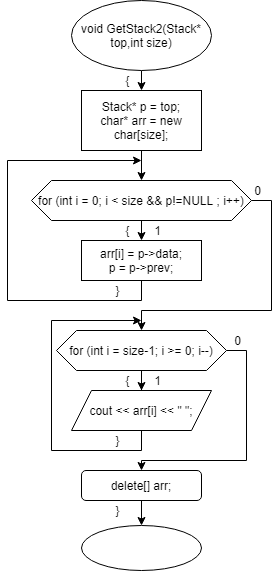
7



8



9



10

**Код**

**Реализация через STL библиотеки**

#include<iostream>

#include<stack>

#include<vector>

using namespace std;

void GetStack(stack<char>stk);

void Addition(stack<char>&stk, int size);

int main()

{

setlocale(LC\_ALL, "ru");

stack<char>stk;

char x;

int size=-1;

while (size < 0) {

cout << "Ввести количество элементов в стеке: ";

cin >> size;

}

for (int i = 1; i <= size; i++) {

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

cin >> x;

stk.push(x);

}

cout << "Стек: ";

GetStack(stk);

Addition(stk,size);

GetStack(stk);

}

void GetStack(stack<char>stk)

{

int size = stk.size();

if (size == 0) {

cout << "Стек пуст!" << endl;

}

else {

for (int i = 1; i <= size; i++) {

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

stk.pop();

}

}

cout << endl;

}

void Addition(stack<char>& stk, int size)

{

vector<char>tmp(size);

char x;

cout << "Введите элемент для вставки: ";

cin >> x;

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

tmp[i] = stk.top();

stk.pop();

}

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

stk.push(x);

stk.push(tmp[i]);

}

}

11

**Реализация через структуру**

#include<iostream>

using namespace std;

struct Stack {

char data;

Stack\* prev;

};

Stack\* MakeStack(int size);

void GetStack(Stack\* top);

Stack\* Addition(Stack\*& top);

void GetStack2(Stack\* top,int size);

int main()

{

setlocale(LC\_ALL, "ru");

int size = -1;

while (size < 0) {

cout << "Ввести количество элементов в стеке: ";

cin >> size;

}

Stack\* stk=MakeStack(size);

GetStack(stk);

Addition(stk);

cout << "Результат: ";

//GetStack(stk);

GetStack2(stk, size\*2);

}

Stack\* MakeStack(int size)

{

char x;

if (size == 0) return NULL;

Stack\* top, \* p;

top = NULL;

p = new Stack;

cout << "Введите 1 элемент: ";

cin >> x;

p->data = x;

p->prev = NULL;

top = p;

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

Stack\* nex = new Stack;

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

cin >> x;

nex->data = x;

nex->prev = top;

top = nex;

}

return top;

}

void GetStack(Stack\* top)

{

if (top == 0) cout << "Стек пуст! ";

else {

Stack\* p = top;

while (p != NULL) {

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

p=p->prev;

}

cout << endl;

}

}

12

Stack\* Addition(Stack\*& top)

{

char x;

int i=0;

cout << "Введите элемент для вставки: ";

cin >> x;

Stack\* p = top;

Stack \* tmp = NULL;

while (p != NULL) {

Stack\* newE = new Stack;

newE->data = x;

newE->prev = p;

if (tmp != NULL)

tmp->prev = newE;

else

top = newE;

tmp = p;

p = p->prev;

}

return top;

}

void GetStack2(Stack\* top,int size)

{

Stack\* p = top;

char\* arr = new char[size];

for (int i = 0; i < size && p!=NULL ; i++) {

arr[i] = p->data;

p = p->prev;

}

for (int i = size-1; i >= 0; i--) {

cout << arr[i] << " ";

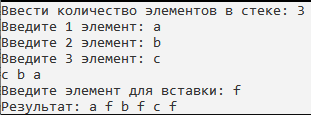
}

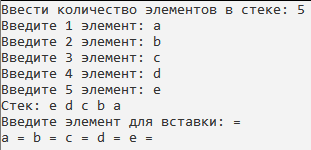
delete[] arr;

}

13

**Скриншоты**





14