**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МО ЭВМ**

**ОТЧЕТ**

**по лабораторной работе № 3**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема:**

**Линейные структуры данных: стек, очередь, дек.**

|  |  |  |
| --- | --- | --- |
| Студент гр. 6383 |  | Никитин К.В. |
| Преподаватель |  | Шолохова О.М. |

Санкт-Петербург

2017

**Цель работы.**

Познакомиться с часто используемыми на практике линейными структурами данных, обеспечивающими доступ к элементам последовательности только через её начало и конец, и способами реализации этих структур, освоить на практике использование стека, очереди и дека для решения задач.

**Постановка задачи.**

Задание предполагает самостоятельную разработку студентом одного или нескольких модулей на языке С++, реализующих согласованный с преподавателем набор операций над иерархическими списками, а также главной программы, непосредственно решающей поставленную задачу. Предполагается выполнение задания в двух вариантах: с использованием базовых функций рекурсивной обработки иерархических списков и без использования рекурсии.

Для представления иерархических списков рекомендуется использовать сокращенную скобочную запись.

Вариант 9 а), б).

В заданном текстовом файле F записан текст, сбалансированный по круглым скобкам:

< текст > ::= < пусто > | < элемент > < текст >

< элемент > ::= < символ > | ( < текст > )

где < символ > – любой символ, кроме (, ). Для каждой пары соответствующих открывающей и закрывающей скобок вывести номера их позиций в тексте, упорядочив пары в порядке возрастания номеров позиций:

а) закрывающих скобок; б) открывающих скобок.

**Основные теоретические положения.**

Ссылочная реализация стека и очереди в динамической памяти в основном аналогична ссылочной реализации линейных списков. Упрощение связано с отсутствием необходимости работать с текущим элементом списка. Идеи такой реализации ясны из рисунка 3.3.

Для ссылочной реализации дека естественно использовать двунаправленный список.

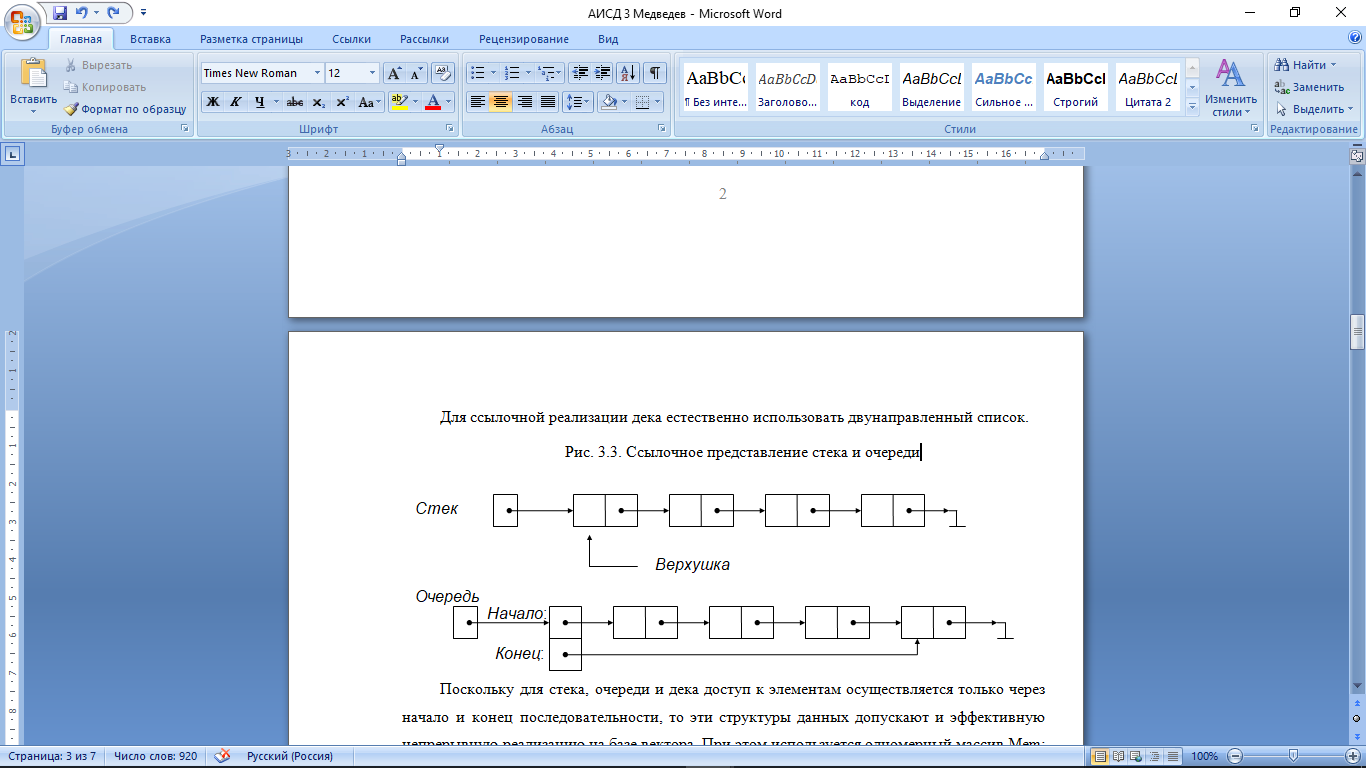


Рис. 3.3. Ссылочное представление стека и очереди

Поскольку для стека, очереди и дека доступ к элементам осуществляется только через начало и конец последовательности, то эти структуры данных допускают и эффективную непрерывную реализацию на базе вектора. При этом используется одномерный массив Mem: array [0..n] of α и переменная Верх :  -1..n.

**Спецификация программы.**

Ограничения на входные данные: Выражение, принимающееся на входе должно быть записано синтаксически правильно. Количества открывающихся и закрывающихся круглых скобок должны быть попарно равны.

Место и форма представления входных данных: входные данные располагаются в файле input.txt или вводятся с клавиатуры (при выполнении программы пользователю предоставляется выбор).

Выходные данные: Программа выводит на экран промежуточные результаты работы функции (занесение и извлечение i-го элемента). Пары чисел, отображающие позиции всех пар скобок.

***Реализация.***

Функции:

1. Функция, выводящая на экран и в файл сообщение об ошибке входных данных и производящая выход из программы.

void ErrorMessege(){

cout << "Error! Uncorrect input. The programm will be interructed.\n";

system("pause");

exit(1);

}

1. Функция, проверяющая корректность введенных данных при работе в меню и выводящая на экран запрос о повторном введении данных, если они введены некорректно.

void fail(int& q){

while (cin.fail() || ((q != 2) && (q !=1))){

cout << "Error. Enter another value.\n";

cin.clear ();

cin.sync();

cin >> q;

}

}

1. Функция, добавляющая элемент в стек.

void Stack::push(int& i){

l++;

st[l] = i;

topOfStack=st[l];

}

1. Функция, извлекающая элемент из стека.

void Stack::pop(){

l--;

topOfStack = st[l];

}

***Тестирование.***

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| (baba(ded)privet) | А) 6 10; 1 17 б) 1 17; 6 10 |
| (()) | А) 2 3; 1 4 Б) 1 4; 2 3 |
| (asdsad( | Error! Wrong input. Program will end. |
| Kirill | There is no brackets |

**Выводы****.**

В результате выполнения лабораторной работы были получены основные сведения о стеке и его реализации для вычисления арифметического выражения, была написана функция, позволяющая вычислить выражение, заданное в постфиксной форме.

**Приложение А. Исходный код.**

**Source.cpp**

#include <cstdlib>

#include "Header.h"

#include <windows.h>

#include <iostream>

#include <fstream>

using namespace std;

using namespace vect;

Stack s;

int main()

{

int w;

setlocale(LC\_ALL, "RUS");

Node vector[MAX];

ifstream f("input.txt");

ofstream f1("output.txt");

list l;

l.head = 0;

l.cur = 0;

char x;

bool g = false;

for (int m = 0; m <= MAX; m++) vector[m].cl = 0;

cout << "Choose input:\n1. from console\n2. from file\n";

cin >> w;

cout << endl;

fail\_menu(w);

bool k = true;

if (w == 1) cin >> x;

if (w == 2)

{

f >> x;

if (f.fail()) ErrorMessege();

}

int j = 0, i = 0;

while (k)

{

j++;

if ((x == ')') && (s.topOfStack == 0)) ErrorMessege();

if (x == '(')

{

g = true;

i++;

s.push(i);

vector[s.topOfStack].op = j;

}

if (x == ')')

{

if (!s.topOfStack) ErrorMessege();

vector[s.topOfStack].cl = j;

if (!l.head)

{

l.head = s.topOfStack;

l.cur = l.head;

}

else

{

vector[l.cur].next = s.topOfStack;

l.cur = s.topOfStack;

}

s.pop();

}

if (w == 1)

{

cin.get(x);

if ((x == '\n') || (x == ' ')) k = false;

}

if (w == 2)

{

f >> x;

if (f.fail()) k = false;

}

}

vector[l.cur].next = 0;

if (s.topOfStack) ErrorMessege();

cout << "Choose task:\n1. a)\n2. б)\n";

cin >> w;

cout << endl;

fail\_menu(w);

cout << endl;

if (w == 1) j = l.head;

if (w == 2) j = 1;

while (vector[j].cl != 0)

{

cout << vector[j].op << " " << vector[j].cl << endl;

f1 << vector[j].op << " " << vector[j].cl << "; ";

if (w == 1) j = vector[j].next;

if (w == 2) j++;

}

if (!g)

{

cout << "There is no brackets.\n";

f1 << "There is no brackets.\n";

}

system("pause");

}

**fucn.cpp**

#include "Header.h"

#include <cstdlib>

#include <iostream>

using namespace std;

namespace vect

{

// Функция, проверяющая корректность введенных данных при работе в меню

//и выводящая на экран запрос о повторном введении данных, если они введены некорректно.

void fail\_menu(int& q)

{

//введенное с клавиатуры значение

while ((cin.fail()) || ((q != 1) && (q != 2)))

{

cout << "Error. Enter another value\n";

cin.clear();

cin.sync();

cin >> q;

}

}

// Функция, выводящая на экран и в файл сообщение об ошибке входных данных и производящая выход из программы.

void ErrorMessege()

{

cout << "Error! Wrong input. Programm will end\n";

system("pause");

exit(1);

}

// Функция, добавляющая элемент в стек.

void Stack::push(int& i)

{

//значение текущего элемента

l++;

st[l] = i;

topOfStack = st[l];

}

// Функция, извлекающая элемент из стека.

void Stack::pop()

{

l--;

topOfStack = st[l];

}

}

**Header.h**

const int MAX = 30;

namespace vect

{

class Stack

{

//объявления класса Stack

public:

int topOfStack;

// переменная, хранящая значение номера элемента, имеющего данные об открывающеся скобке, но не имеющая о закрывающейся

int st[MAX]; // массив, хранящий данные стека

int l;

Stack()

{

l = 0;

st[l] = 0;

topOfStack = 0;

};

void push(int& i); // функция, добавляющая элемент в стек

void pop(); // функция, извлекающая элемент из стека

};

struct list

{

// объявление структупы для хранения позиций

int head;

int cur;

};

struct Node

{

// Объявление структуры для данных

int next;//Значение следующего элемента вектора

int op;//Значение номера позиции открывающейся скобки

int cl;// Значение номера позиции закрывающейся скобки

};

void ErrorMessege(); // функция, выводящая сообщение об ошибке и прерывающая работу программы

void fail\_menu(int& q);//функция проверки на синтаксическую ошибку

}