**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 |  | Медведев Г.О. |
| 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 11 а).

Рассматривается выражение следующего вида:

< выражение >::= < терм > | < терм > + < выражение > |

< терм > - < выражение >

< терм > ::= < множитель > | < множитель > \* < терм >

< множитель > ::= < число > | < переменная > | ( < выражение > ) |

< множитель > ^ < число >

< число > ::= < цифра >

< переменная > ::= < буква >

Такая форма записи выражения называется инфиксной.

Постфиксной (префиксной) формой записи выражения aDb называется запись, в которой знак операции размещен за (перед) операндами: abD (Dab).

Постфиксная и префиксная формы записи выражений не содержат скобок.

Требуется вычислить выражение в постфиксной форме (задан текстовый файл postfix);

**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.

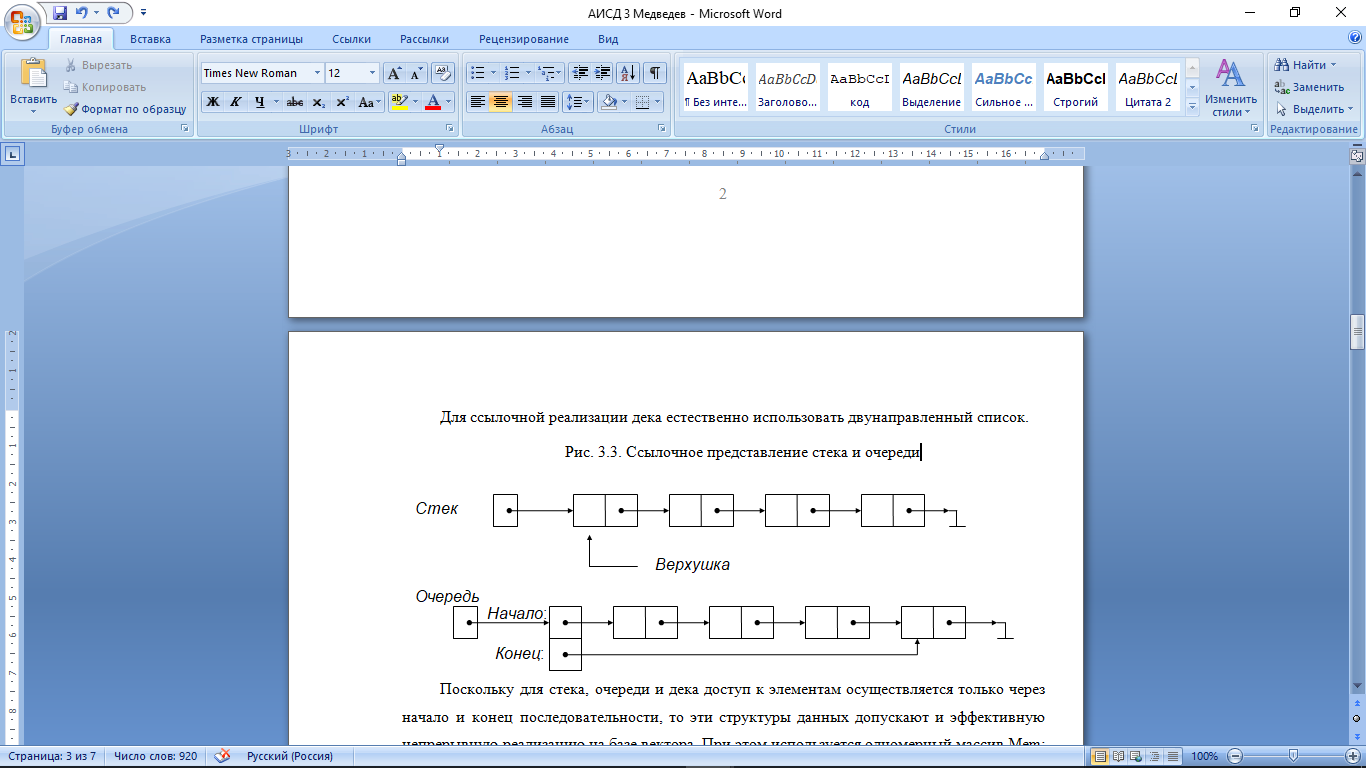


Рис. 3.3. Ссылочное представление стека и очереди

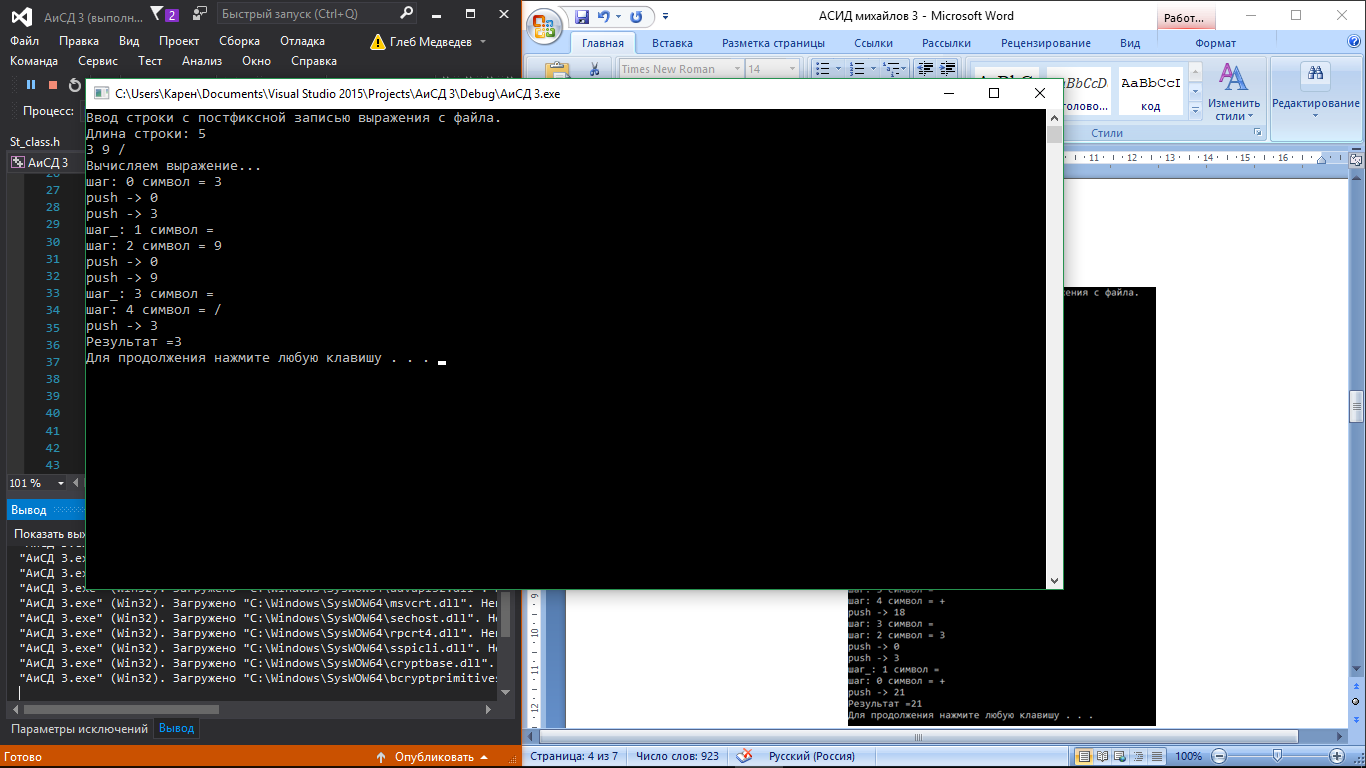
Поскольку для стека, очереди и дека доступ к элементам осуществляется только через начало и конец последовательности, то эти структуры данных допускают и эффективную непрерывную реализацию на базе вектора. При этом используется одномерный массив Mem: array [0..n] of α и переменная Верх :  -1..n.

**Спецификация программы.**

Программа предназначена для вычисления арифметического выражения. Входными данными для программы являются:

-арифметическое выражение, заданное в постфиксной форме.

***Пример диалога с пользователем.***

****

***Реализация.***

Функции:

void push(const Elem &x) – функция, которая помещает элемент в стек.

void pop(void) – функция , которая выталкивает элемент.

Elem top(void) – функция, возвращает указатель на верхушку стека.

Elem pop2(void) – функция, которая совмещает результат действия функций Top и Pop.

***Тестирование.***

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| 2 3 + | 5 |
| 9 9 \* | 81 |
| 100 50 - 23 21 - \* | 100 |
| 3 9 / | 3 |

**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.**

**St\_class.h**

#ifndef ST\_CLASS\_H\_INCLUDED

#define ST\_CLASS\_H\_INCLUDED

//-------------------------------------

template <class Elem>

class Stack

{

private:

struct node

{ //

Elem \*hd;

node \*tl;

// constructor

node()

{

hd = NULL; tl = NULL;

};

};// end node

node \*topOfStack;

public:

Stack()

{

topOfStack = NULL;

}//;

//-------------------------------------

Elem Stack::top(void) //

{// PreCondition: not null

if (topOfStack == NULL) { cerr << "Error: top(null) \n"; system("pause"); exit(1); }

else return \*topOfStack->hd;

}

//-------------------------------------

void Stack::pop(void)//

{// PreCondition: not null

if (topOfStack == NULL) { cerr << "Error: pop(null) \n"; system("pause"); exit(1); }

else

{

node \*oldTop = topOfStack;

topOfStack = topOfStack->tl;

delete oldTop->hd;

delete oldTop;

}

}

//-------------------------------------

Elem Stack::pop2(void)//

{// PreCondition: not null

if (topOfStack == NULL) { cerr << "Error: pop(null) \n"; system("pause"); exit(1); }

else

{

node \*oldTop = topOfStack;

Elem r = \*topOfStack->hd;

topOfStack = topOfStack->tl;

delete oldTop->hd;

delete oldTop;

return r;

}

}

//-------------------------------------

void Stack::push(const Elem &x)//

{

node \*p;

p = topOfStack;

topOfStack = new node;

if (topOfStack != NULL) {

topOfStack->hd = new Elem;

\*topOfStack->hd = x;

cout << "push -> " << x << endl; // Demo

topOfStack->tl = p;

}

else { cerr << "Memory not enough\n"; system("pause"); exit(1); }

}

//-------------------------------------

bool Stack::isNull(void)//

{

return (topOfStack == NULL);

}

//-------------------------------------

Stack::~Stack(void)//

{

while (topOfStack != NULL) {

pop();

}

}

};

#endif // ST\_CLASS\_H\_INCLUDED

**main.cpp**

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <windows.h>

#include "St\_class.h"

using namespace std;

int main() {

char a[100];

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

cout << "Ввод строки с постфиксной записью выражения с файла." << endl;

ifstream fin("postfix.txt");

fin >> noskipws;

if (!fin) { cout << "File not open for reading!\n"; return 1; }

int n1 = 100;

int n = 0;

while (n<n1 && fin >> a[n]) n++;

cout << "Длина строки: " << n << endl;

for (int i = 0; i<n; i++) cout << a[i];

cout << endl;

cout << "Вычисляем выражение..." << endl;

Stack<int> s;

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

{

cout << "шаг: " << i << " символ = " << a[i] << endl;

if (a[i] == '+')

s.push(s.pop2() + s.pop2());

if (a[i] == '-')

s.push(s.pop2() - s.pop2());

if (a[i] == '\*')

s.push(s.pop2() \* s.pop2());

if (a[i] == '/')

s.push(s.pop2() / s.pop2());

if ((a[i] >= '0') && (a[i] <= '9'))

s.push(0);

while ((a[i] >= '0') && (a[i] <= '9'))

{

s.push(10 \* s.pop2() + (a[i++] - '0'));

cout << "шаг\_: " << i << " символ = " << a[i] << endl;

}

}

cout << "Результат =" << s.pop2() << endl;

system("pause");

return (0);

}