**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 |  | Михайлов Ю.А. |
| Преподаватель |  | Шолохова О.М. |

Санкт-Петербург

2017

1. **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 .

1. **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 б).

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 префиксной форме (задан текстовый файл prefix);

1. **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.

*Стек*:

*Верхушка стека*

*Начало*:

*Конец*:

*Очередь*:

Рис. 3.3. Ссылочное представление стека и очереди

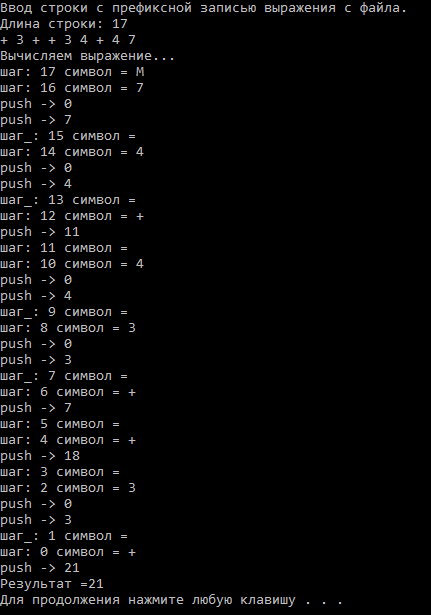
Поскольку для стека, очереди и дека доступ к элементам осуществляется только через начало и конец последовательности, то эти структуры данных допускают и эффективную непрерывную реализацию на базе вектора. При этом используется одномерный массив Mem: array [0..n] of α и переменная Верх :  -1..n.

1. **Спецификация программы.**

Программа предназначена для вычисления арифметического выражения. Программа написана на языке C++. Входными данными для программы являются:

-арифметическое выражение, заданное в префиксной форме.

1. **Пример диалога с пользователем.**

****

1. **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.

void result(ifstream& fin,Stack<int> s) – функция, которая вычисляет значение заданного выражения и печатает его.

**Тестирование.**

|  |  |
| --- | --- |
| Входные данные | Выходные данные |
| +23 | 5 |
| \*99 | 81 |
| -89 | -1 |
| /93 | 3 |
| \*3+34 | 21 |

**Выводы****.**

В результате выполнения лабораторной работы мною были получены основные сведения о стеке и его реализации для вычисления арифметического выражения. Ознакомился с базовыми функциями обработки стека. Была написана функция, позволяющая вычислить выражение , заданное в префиксной форме.

**Приложение А. Исходный код.**

**St\_class.h**

template <class Elem>

class Stack

{

private:

struct node

{ //

Elem \*hd;

node \*tl;

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();

}

}

};

**main.cpp**

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <windows.h>

#include "St\_class.h"

using namespace std;

void result(ifstream& fin, Stack<int> s) {

char a[100];

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;

for (int i = n - 1; i >= 0; i--)

{

if (a[i] == ' ')

continue;

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 (isdigit(a[i])) {

s.push(a[i] - '0');

while (isdigit(a[--i]))

{

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

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

}

}

}

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

}

int main() {

SetConsoleCP(1251);

SetConsoleOutputCP(1251);

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

ifstream fin("prefix.txt");

fin >> noskipws;

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

Stack<int> s;

result(fin, s);

s.~Stack();

system("pause");

return (0);

}