**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.

|  |  |  |
| --- | --- | --- |
| Студентка гр. 6383 |  | Гомонова А. А. |
| Преподаватель |  | Шолохова О. М. |

Санкт-Петербург

2017

**Цель работы**

Научиться использовать стек на языке С++.

Используя стек, написать программу, реализующую поставленную задачу.

**Задание**

**Вариант 6в.**

Проверить, является ли содержимое заданного текстового файла *F* правильной записью формулы следующего вида:

< формула > ::= < терм > | < терм > + < формула > | < терм > - < формула >  
< терм > ::= < имя > | ( < формула > )

< имя > ::= *x* | *y* | *z*

Если не является, то указать место ошибочной позиции.

**Содержательная постановка задачи**

На входе программа получает последовательность символов из файла “queue.txt”. Для решения данной задачи нужно и требуется использовать стек. Стек — динамическая структура данных, представляющая из себя упорядоченный набор элементов, в которой добавление новых элементов и удаление существующих производится с одного конца, называемого вершиной стека.

По определению, элементы извлекаются из стека в порядке, обратном их добавлению в эту структуру, т.е. действует принцип "последний пришёл — первый ушёл".

Исходные данные представляют собой строку, содержащую операнды и операции такие, как + , - .

**Структура данных и алгоритм работы программы**

class Stack

{

public:

char arr[];

int work[2];

void init();

void push(char a);

bool can\_pop();

char pop();

Stack();

};

*Структуры данных*: текст, содержащую операнды и операции такие, как +, -.

Стек — динамическая структура данных, представляющая из себя упорядоченный набор элементов, в которой добавление новых элементов и удаление существующих производится с одного конца, называемого вершиной стека.

По определению, элементы извлекаются из стека в порядке, обратном их добавлению в эту структуру, т.е. действует принцип "последний пришёл — первый ушёл".

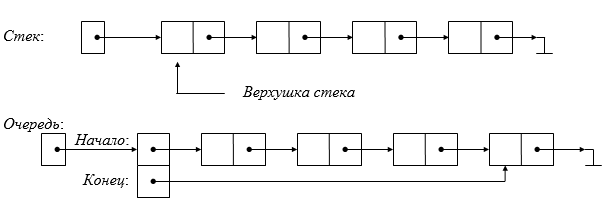


Рисунок 1-Ссылочное представление стека и очереди

О́чередь — [абстрактный тип данных](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D1%8B%D0%B9_%D1%82%D0%B8%D0%BF_%D0%B4%D0%B0%D0%BD%D0%BD%D1%8B%D1%85) с дисциплиной доступа к элементам «первый пришёл — первый вышел» ([FIFO](https://ru.wikipedia.org/wiki/FIFO_(%D0%B8%D0%BD%D1%84%D0%BE%D1%80%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), First In — First Out). Добавление элемента (принято обозначать словом enqueue — поставить в очередь) возможно лишь в конец очереди, выборка — только из начала очереди (что принято называть словом dequeue — убрать из очереди), при этом выбранный элемент из очереди удаляется.

Ввод осуществляется с консоли или из файла.

Если входные данные нас удовлетворяют, тогда заносим формулу в стек. После мы вытаскиваем из стека элемент и проверяем, удовлетворяет ли он нашим условиям. Удовлетворяющие нас элементы заносим в стек для вывода на экран. Далее проводим аналогичные преобразования. На выходе мы должны получить является ли написанная нами формула ФОРМУЛОЙ, если при написании мы допустили ошибку, программа нам выведет место ошибки и подскажет нам как можно было бы исправить что бы формула удовлетворяла нашему условию.

**Описание функций**

* void call\_error(Queue & Q3, char symb, int numb) - функция показывающая место ошибки (Q3 – элемент класса очередь, symb – проверяемый символ, numb – номер ошибки)
* bool bracket\_c\_f(char symb) - проверка закрывающей скобки (symb – проверяемый символ, функция возвращает true если скобка закрывающая, false – если нет)
* bool bracket\_o\_f(char symb) ­- проверка открывающей скобки (symb – проверяемый символ, функция возвращает true если скобка открывающая, false – если нет)
* bool sign\_f(char symb) - проверка является элемент знаком (symb – проверяемый символ, функция возвращает true если проверяемый символ – арифметический знак, false – если нет)
* bool name\_f(char symb) - проверка является ли элемент именем (symb – проверяемый символ, функция возвращает true если проверяемый символ –переменная x|y|z , false – если нет)
* bool is\_it\_f(Stack Q1, Queue & Q3) – проверка формула или нет (в функции из стека извлекается элемент и проверяется его является ли он переменной или арифметическим знаком, если все условия выполнены, то возвращается true, если какой-то из элементов не удовлетворяет условиям, то - false)
* void mainf(ifstream & fin, Stack & Q2) - является ли строка формулой, в функции осуществляется вывод ошибок или вывод формулы или части строки, являющейся формулой.

**Тестирование**

|  |  |  |  |
| --- | --- | --- | --- |
| **Номер тестирования** | **Входные данные** | **Результирующие данные** | **Оценка** |
| 1 | (x-z-o+x+y-2) | Это не формула | Ожидалось имя, типа [x],[y],[z], или открывающая скобка - [(] |
| 2 | (x+y\*(x-z)) | Это не формула | Ожидался знак, типа [+],[-], или закрывающая скобка - [)] |
| 3 | (x+y-z) | Это формула |  |
| 4 | () | Это не формула | Ожидалось имя, типа [x],[y],[z], или открывающая скобка - [(] |
| 5 | x+y+(y-z)+(x-z) | Это формула |  |
| 6 | x+y+x-(xz)(xy) | Это не формула | Из стека извлечен элемент [+], который не удовлетворяет условиям |
| 7 | x | Это формула |  |
| 8 | 100 | Это не формула | Ожидалось имя, типа [x],[y],[z], или открывающая скобка - [(] |
| 9 | (x)+(y) | Это формула |  |
| 10 | z-x | Это формула |  |

**Вывод**

В ходе лабораторной работы была написана, отлажена и протестирована программа, с использованием стека.

**Код программы**

**Main.cpp**

#include <iostream>

#include <fstream>

#include <cstdlib>

#include <locale.h>

#include "Stack.h"

#include "Queue.h"

using namespace std;

void call\_error(Queue & Q3, char symb, int numb) // функция показывающая место ошибки

{

cout << ",который не удовлетворяет условиям формулы." << endl;

cout << "Это не формула!" << endl;

switch (numb)

{

case 1:

{

cout << "Ожидалось имя, типа [x],[y],[z], или открывающая скобка - [(]." << endl;

break;

}

case 2:

{

cout << "Ожидался знак, типа [+],[-], или закрывающая скобка - [)]." << endl;

break;

}

default: break;

}

cout << "Часть строки, которая является формулой и может быть завершена." << endl;

while (Q3.can\_pop())

cout << Q3.pop() << " ";

cout << "\nПрограмма будет завершена." << endl;

system("pause");

exit(1);

}

bool bracket\_c\_f(char symb) // проверка закрывающей скобки

{

if (symb == ')')

return true;

else

return false;

}

bool bracket\_o\_f(char symb) // проверка открывающей скобки

{

if (symb == '(')

return true;

else

return false;

}

bool sign\_f(char symb) // проерка является эллемент знаком

{

if (symb == '+' || symb == '-')

return true;

else

return false;

}

bool name\_f(char symb) // проверка является ли элеммент именем

{

if (symb == 'x' || symb == 'y' || symb == 'z')

return true;

else

return false;

}

bool is\_it\_f(Stack Q1, Queue & Q3) // является ли строка формулой

{

char symb;

bool name = true;

bool sign = false;

bool bracket\_c = false;

bool bracket\_o = true;

bool is\_el = false;

int open = 0;

int close = 0;

while (Q1.can\_pop())

{

symb = Q1.pop();

cout << "Из стека извлечен элемент [" << symb << "] ";

is\_el = false;

if (!is\_el && name && name\_f(symb))

{

cout << ",который является именем. " << endl;

is\_el = true;

name = false;

sign = true;

bracket\_c = true;

bracket\_o = false;

Q3.push(symb);

}

if (!is\_el && sign && sign\_f(symb))

{

cout << ",который является знаком." << endl;

is\_el = true;

name = true;

sign = false;

bracket\_c = false;

bracket\_o = true;

Q3.push(symb);

}

if (!is\_el && bracket\_o && bracket\_o\_f(symb))

{

cout << ",который является открывающей скобкой. " << endl;

is\_el = true;

bracket\_o = true;

bracket\_c = false;

name = true;

sign = false;

open++;

Q3.push(symb);

}

if (!is\_el && bracket\_c && bracket\_c\_f(symb))

{

close++;

cout << ",который является закрывающей скобкой." << endl;

if (close <= open)

{

close--;

open--;

is\_el = true;

bracket\_o = false;

bracket\_c = true;

name = false;

sign = true;

Q3.push(symb);

}

else

call\_error(Q3, symb, 2);

}

if (!is\_el && (name && bracket\_o))

call\_error(Q3, symb, 1);

else

if (!is\_el && (sign && bracket\_c))

call\_error(Q3, symb, 2);

}

if (open != close)

call\_error(Q3, symb, 2);

return true;

}

void print\_f(Queue & Q3) // вывести формулу

{

while (Q3.can\_pop())

{

cout << Q3.pop() << " ";

}

}

void mainf(ifstream & fin, Stack & Q2) // функция в которой проверятся условие является ли формулой

{

Queue Q3;

int n = 1;

char symb;

fin >> symb;

while (!(fin.eof()))

{

Q2.push(symb);

Q3.push(symb);

fin >> symb;

}

Stack Q1;

cout << "Заносим элементы строки в стек:" << endl;

while (Q2.can\_pop())

{

char a = Q2.pop();

cout << "В стек помещен элемент [" << a << "]." << endl;

Q1.push(a);

}

cout << "Считанная из файла строка.";

print\_f(Q3);

cout << endl;

bool f = is\_it\_f(Q1, Q3);

cout << "Это формула." << endl;

cout << "Ее вид: " << endl;

print\_f(Q3);

}

int main()

{

setlocale(LC\_ALL, "rus");

int answer;

cout << "Выберите нужное действие." << endl;

cout << "============================================" << endl;

cout << "1.Считать формулу из файла." << endl;

cout << "2.Завершить программу." << endl;

cout << "============================================" << endl;

cout << "Ваш ответ: ";

cin >> answer;

cout << "============================================" << endl;

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

{

system("cls");

cout << "Вы ввели неверное значение. Попытайтесь снова." << endl;

cin.clear();

cin.sync();

cout << "============================================" << endl;

cout << "1.Считать формулу из файла." << endl;

cout << "2.Завершить программу." << endl;

cout << "============================================" << endl;

cout << "Ваш ответ: ";

cin >> answer;

cout << "============================================" << endl;

}

if (answer == 1)

{

ifstream fin("queue.txt");

Stack Q2;

mainf(fin, Q2);

fin.close();

}

else

cout << "Работа будет завершена." << endl;

cout << endl;

system("pause");

}

**Stack.h**

#include <iostream>

using namespace std;

const size\_t size\_1 = 100;

class Stack

{

public:

char arr[size\_1];

int start, top;

void init()

{

for (int i = 0; i < size\_1; i++)

arr[i] = 0;

top = -1;

start = -1;

}

void push(char a)

{

if (start != size\_1 - 1)

{

if (top == -1)

{

arr[0] = a;

start = 0;

top = 0;

}

else

{

arr[top + 1] = a;

top++;

}

}

else

cout << "Стек переполнен." << endl;

}

bool can\_pop()

{

if (top != -1)

return 1;

else

return 0;

}

char pop()

{

if (top != -1)

{

if (top != start)

{

char tmp = arr[top];

arr[top] = 0;

top--;

return tmp;

}

else

{

char tmp = arr[top];

arr[top] = 0;

top = -1;

start = -1;

return tmp;

}

}

else

{

cout << "Не удалось извлечь элемент, стек пуст." << endl;

return 0;

}

}

Stack()

{

init();

}

};

**Queue.h**

#include <iostream>

using namespace std;

class Queue

{

public:

char arr[size\_1];

int work[2];

void init()

{

for (int i = 0; i < size\_1; i++)

arr[i] = 0;

work[0] = -1;

work[1] = -1;

}

void push(char a)

{

if (work[1] != size\_1 - 1)

{

if (work[0] == -1)

{

arr[0] = a;

work[1] = 0;

work[0] = 0;

}

else

{

arr[work[1] + 1] = a;

work[1]++;

}

}

else

cout << "Очередь переполнена." << endl;

}

bool can\_pop()

{

if (work[0] != -1)

return 1;

else

return 0;

}

char pop()

{

if (work[0] != -1)

{

if (work[0] != work[1])

{

char tmp = arr[work[0]];

arr[work[0]] = 0;

work[0]++;

return tmp;

}

else

{

char tmp = arr[work[0]];

arr[work[0]] = 0;

work[0] = -1;

work[1] = -1;

return tmp;

}

}

else

{

cout << "Не удалось извлечь элемент, очередь пуста." << endl;

return 0;

}

}

Queue()

{

init();

}

};