МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра МОЭВМ

ОТЧЕТ

по лабораторной работе №3
по дисциплине «Алгоритмы и структуры данных»
Тема: Стеки и очереди

Студент гр. 8304 _____ Перелыгин Д.С.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Формирование практических навыков работы с иерархическими списками и их рекурсивной обработки. Изучение и практическое применение постфиксной записи арифметических выражений.

Задание.

Пусть выражение (логическое, арифметическое, алгебраическое*) представлено иерархическим списком. В выражение входят константы и переменные, которые являются атомами списка. Операции представляются в префиксной форме ((<операция> <аргументы>)), либо в постфиксной форме (<аргументы> <операция>)). Аргументов может быть 1, 2 и более. Например (в префиксной форме): (+ a (* b (- c))) или (OR a (AND b (NOT c))).

В задании даётся один из следующих вариантов требуемого действия с выражением: проверка синтаксической корректности, упрощение (преобразование), вычисление.Пример упрощения: (+ 0 (* 1 (+ a b))) преобразуется в (+ a b).В задаче вычисления на входе дополнительно задаётся список значений переменных((x1 c1) (x2 c2) … (xk ck)), где xi – переменная, а ci – её значение (константа).В индивидуальном задании указывается: тип выражения (возможно дополнительно - состав операций), вариант действия и форма записи. Всего 9 заданий.

17) логическое, упрощение, префиксная форма

Основные теоретические положения.

2.2. Реализация стека и очереди

Ссылочная реализация стека и очереди в динамической памяти в основном аналогична ссылочной реализации линейных списков, подробно рассмотренной в 1.3. Упрощение здесь связано с отсутствием необходимости работать с текущим («внутренним») элементом списка. Идеи такой реализации ясны из рис. 2.1.

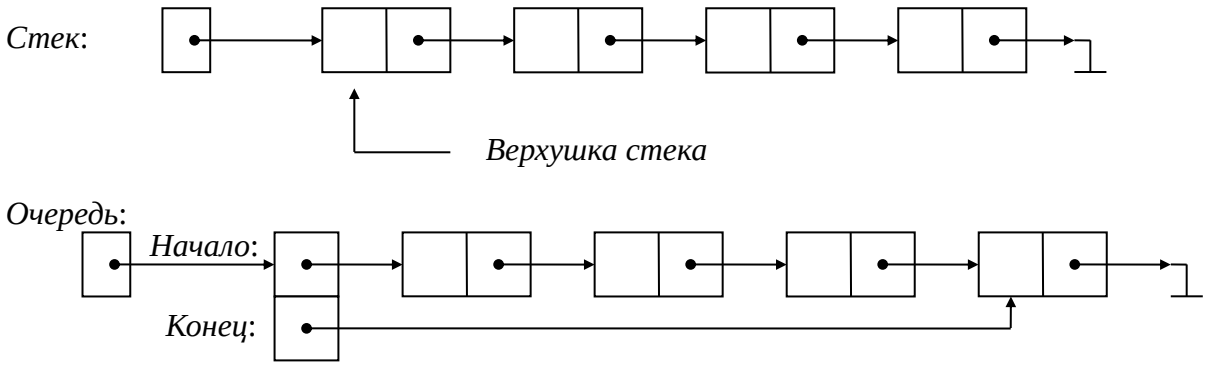

Рис. 2.1. Ссылочное представление стека и очереди

Для ссылочной реализации дека естественно использовать Л2-список (см. 1.5).

Поскольку для стека, очереди и дека доступ к элементам осуществляется только через начало и конец последовательности, то эти структуры данных допускают эффективную *непрерывную реализацию на базе вектора* (в отличие от Л1- и Л2-списков, см. 1.1, с. 4).

При *непрерывной реализации ограниченного стека* на базе вектора для представления стека используется одномерный массив (вектор) *Мет*: **array** [1..*n*] **of** α и переменная *Верх*: 1..*n* (рис. 2.2).

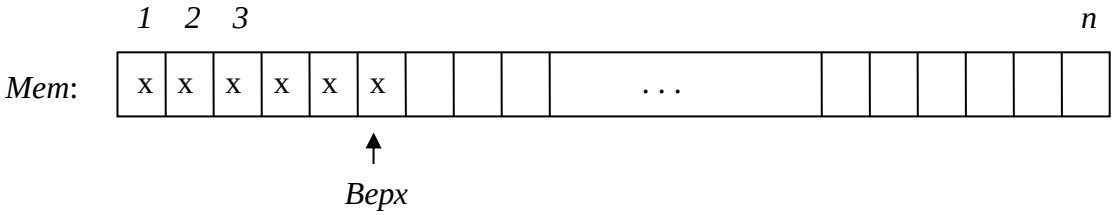

Рис. 2.2. Непрерывное представление стека в векторе

Для пустого стека *Верх* = 0, для целиком заполненного стека *Верх* = *n*. Вершина стека доступна как *Мет* [*Верх*], операция *Pop* реализуется как *Верх*:= *Верх* – 1, а операция *Push* (*p*, *s*) как **begin** *Верх*:= *Верх* + 1; *Мет* [*Верх*]: = *p* **end** при 0 ≤ *Верх* < *n*.

На базе одного вектора можно реализовать два стека, ограниченных в совокупности. Такую реализацию иллюстрирует рис. 2.3.

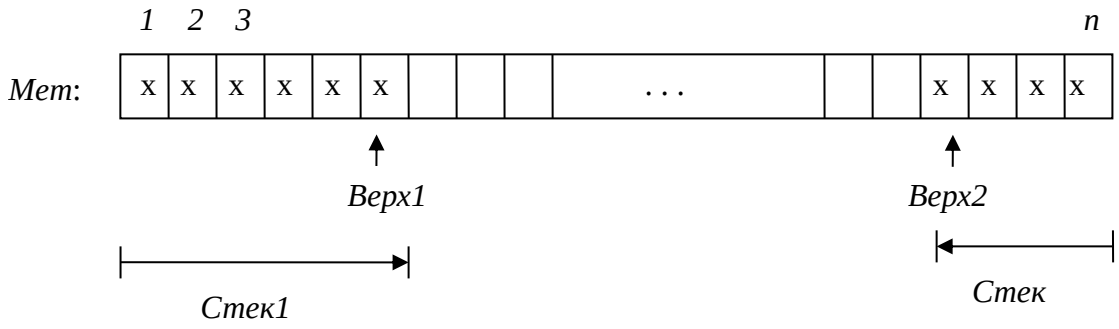

Рис. 2.3. Непрерывное представление двух стеков,
ограниченных в совокупности

Рассмотрим основные идеи *непрерывной реализации ограниченной очереди* на базе вектора. На рис. 2.4 изображен вектор *Мет* [1..*n*] и переменные *Начало*, *Конец*: 1..*n*, идентифицирующие начало и конец очереди при ее непрерывном размещении в векторе *Мет*.

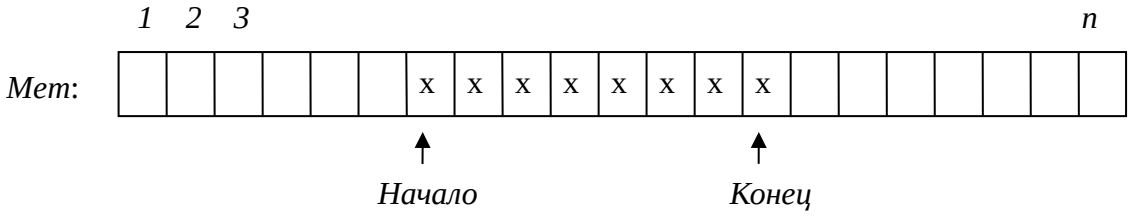

Рис. 2.4. Непрерывное представление очереди в векторе

Особенностью такого представления является наличие ситуации, когда последовательность элементов очереди по мере их добавления может выходить за границу вектора, продолжаясь с его начала (вектор имитирует здесь так называемый *кольцевой буфер*). Эта ситуация изображена на рис. 2.5.

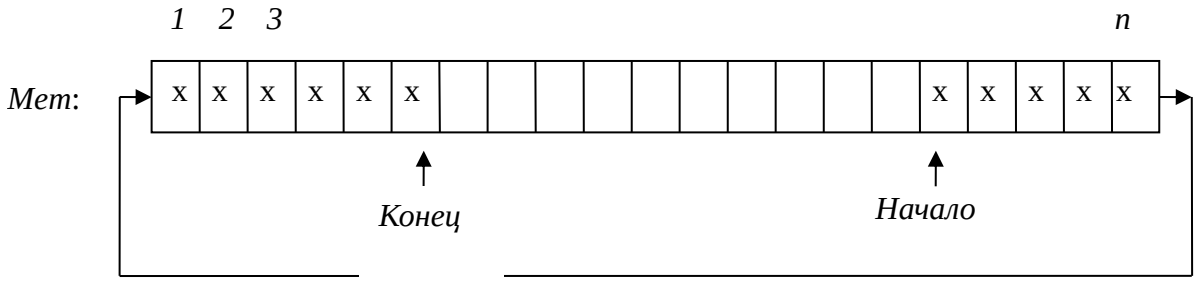

Рис. 2.5. Непрерывное представление очереди в кольцевом буфере

Двух переменных *Начало* и *Конец* недостаточно, чтобы различить в данном представлении, например, два следующих состояния очереди: 1) *Начало* = *Конец* + 1 и очередь пуста (рис. 2.6, *а*); 2) *Начало* = *Конец* + 1 и очередь полна (рис. 2.6, *б*). Простым решением этой проблемы является введение еще одной переменной, идентифицирующей состояние очереди, а именно переменной *Длина*, значение которой задает текущее количество

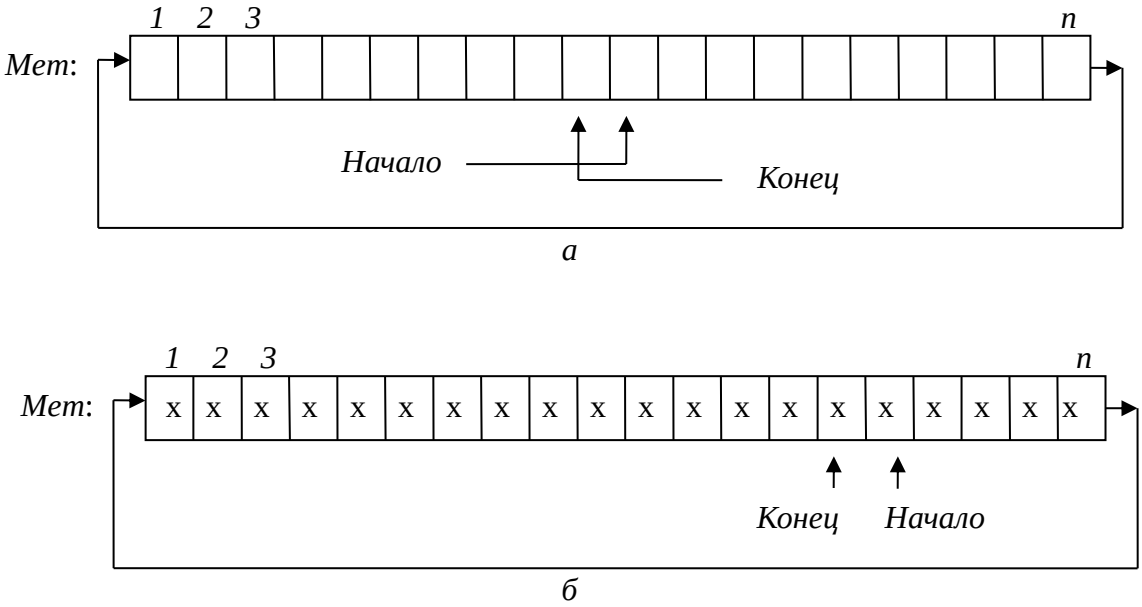

Рис.2.6. Два состояния очереди, при которых *Начало*=*Конец*+1:
а – очередь пуста, *б* – очередь полна

элементов в очереди (для пустой очереди *Длина* = 0, для полной очереди *Длина* = *n*).

Аналогичным образом может быть реализован *дек*.

Ход работы.

Программа реализована с помощью QT creator.

1. Был создан класс стека на основе массива, перегружены методы для работы с ним. Также был сделан простейший графический интерфейс.

2. Производится считывание открытие и считывание строки из файла, после этого при нажатии соответствующей кнопки происходит обработка строки с одновременным отображением текущего состояния стека и происходящих проверок. В случае нарушения порядка, программа говорит об этом и завершает выполнение функции.

3. Аналогичным образом работает пошаговое выполнение этого же алгоритма, запуск и шаг которого реализованы на две разные кнопки.

Тестирование.

Входные данные	Ответ программы	Ожидаемый ответ
AAAAA	**NOPE**	**NOPE**
ABBBCBBBA DABACAAAD BBACABB	**NOPE**	**NOPE**
ABBBCBBBA DAAACAAAD BBACABB	**YES**	**YES**

Выводы.

В ходе выполнения работы были отработаны навыки работы с стеком, их представлением, реализацией и обработкой. Создана программа для проверки выражения на корректность с простейшим графическим интерфейсом.

Приложение А.

Файл mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include <QApplication>
#include <QtGui>
#include <QFileDialog>
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#include <clocale>
#include <windows.h>
#include "mainwindow.h"
#include <QMessageBox>
#include <QTextBrowser>
#include <QTextEdit>
#include <vector>
#include <conio.h>
using namespace std;

string Name;
bool started = true;


template <class T>
class Stack
{
private:
    T* arr;
    int size, len;
    void new_size(int s)
    {
        if (s <= 0) return;
```

```
        T* new_arr = new T[s];
        int min_s = s < len ? s : len;
        for (int i = 0; i < min_s; i++) new_arr[i] = arr[i];
        delete[] arr;
        size = s;
        arr = new_arr;
    }

public:
    Stack()
    {
        len = 0;
        arr = new T[size = 5];
    }

    Stack(Stack& st)
    {
        arr = new T[size = st.size];
        len = st.len;
        for (int i = 0; i < len; i++) arr[i] = st.arr[i];
    }

    ~Stack()
    {
        delete[] arr;
    }

    int Size()
    {
        return len;
    }

    void Push(T x)
    {
        if (len == size) new_size(size << 1);
        arr[len++] = x;
    }
    T Pop()
    {
        if (size > 10 && len < size >> 2) new_size(size >> 1);
        if (len == 0) return arr[len];
        return arr[--len];
    }
    T Top()
    {
        return arr[--len];
    }
    void Clear()
    {
        delete arr;
        len = 0;
        arr = new T[size = 5];
    }

    bool IsEmpty()
    {
        return len == 0;
    }

};

int check()
{}
```

```
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}


void MainWindow::on_pushButton_3_clicked()
{
    /*QString fileName = QFileDialog::getOpenFileName(this,
                                QString::fromUtf8("Открыть файл"),
                                QDir::currentPath(),
                                "Images (*.png *.xpm *.jpg);;All files
(*.*)");*/
    QString FileName = QFileDialog::getOpenFileName(this, "OpenDialog",
    QDir::homePath(), "*.txt;; *.*");
    Name = FileName.toStdString();

}
//void QTextEdit::append ( const QString & text )
/* for( ; ; )
                {
                QApplication::processEvents();
                if(!started)break;


                }
                started = true;*/

void MainWindow::on_pushButton_clicked()
{
    using namespace std;
        ui->textBrowser->clear();
        bool correct = true;
        int i = 0, j = 0;

        string out;
        string text;
         vector<char> show;

        ifstream fin;
        //cout << Name;
        fin.open(Name);
        getline(fin, text);
        fin.close();

        cout << text << '\n';
        Stack<char> S;
        ui->textBrowser->insertPlainText("Text:\n");
        ui->textBrowser->insertPlainText(QString::fromStdString(text));
        ui->textBrowser->insertPlainText("\n\n\n");
        if(text.length()==0)
        {
            ui->textBrowser->insertPlainText("NOPE ");
            correct=false;
        }
```

```
while (i< text.length())
{
    ui->textBrowser->insertPlainText("Input:\n\n\n");
    while (text[i] != 'C')
    {
        if (text[i]== 'D' || i == text.length())
        {

            ui->textBrowser->insertPlainText("NOPE ");
            correct=false;
            break;
        }

        S.Push(text[i]);
        show.push_back(text[i]);
         ui->textBrowser->insertPlainText("Stack:\n");
         for (j=show.size()-1;j>=0;j--)
         {
             ui->textBrowser->insertPlainText(QChar(show[j]));
             ui->textBrowser->insertPlainText("\n");
         }

        i++;
        //cout << i << '\n';

    }
    if (correct==false)
    break;
    i++;
    ui->textBrowser->insertPlainText("Check symbols after C:\n\n\n");
    while (i<text.length() && text[i] != 'D')
    {
        char temp;
        temp = S.Pop();
        ui->textBrowser->insertPlainText("Stack:\n");
        for (j=show.size()-1;j>=0;j--)
        {
            ui->textBrowser->insertPlainText(QChar(show[j]));
            ui->textBrowser->insertPlainText("\n");
        }
        ui->textBrowser->insertPlainText("Try to check:\n");
        ui->textBrowser->insertPlainText(QChar(show[show.size()-1]));
        ui->textBrowser->insertPlainText(" and ");
        ui->textBrowser->insertPlainText(QChar(text[i]));
        ui->textBrowser->insertPlainText("\n");
        if (temp != text[i])
        {
            ui->textBrowser->insertPlainText("NOPE ");
            cout << temp << "   " << text[i] << '\n';
            cout << i << '\n';
            correct = false;
            break;
        }
        else
        {
            show.pop_back();
            i++;
            ui->textBrowser->insertPlainText("Succeed check!:\n");
        }

    }
    if (correct==false)
    break;
```

```
            i++;
        }
        if (correct)
        {
        ui->textBrowser->insertPlainText("YES");
        }

}

void MainWindow::on_pushButton_2_clicked()
{
    QWidget::close();
}

void MainWindow::on_pushButton_4_clicked()
{
    started=false;
}

void MainWindow::on_pushButton_5_clicked()
{
    using namespace std;
    ui->textBrowser->clear();
        bool correct = true;
        int i = 0, j = 0;

        string out;
        string text;
         vector<char> show;

        ifstream fin;
        //cout << Name;
        fin.open(Name);
        getline(fin, text);
        fin.close();

        cout << text << '\n';
        Stack<char> S;


        ui->textBrowser->insertPlainText("Text:\n");
        ui->textBrowser->insertPlainText(QString::fromStdString(text));
        ui->textBrowser->insertPlainText("\n\n\n");
        if(text.length()==0)
        {
            ui->textBrowser->insertPlainText("NOPE ");
            correct=false;
        }
        while (i< text.length())
        {
            ui->textBrowser->insertPlainText("Input:\n\n\n");
                        for( ; ; )
                        {
                        QApplication::processEvents();
                        if(!started)break;


                        }
                        started = true;
            while (text[i] != 'C')
            {
                if (text[i]== 'D' || i == text.length())
                {
```

```
            ui->textBrowser->insertPlainText("NOPE ");
            correct=false;
            break;
        }

        S.Push(text[i]);
        show.push_back(text[i]);
         ui->textBrowser->insertPlainText("Stack:\n");
         for (j=show.size()-1;j>=0;j--)
         {
             ui->textBrowser->insertPlainText(QChar(show[j]));
             ui->textBrowser->insertPlainText("\n");
         }

        i++;
                        for( ; ; )
                        {
                        QApplication::processEvents();
                        if(!started)break;


                        }
                        started = true;

}
if (correct==false)
break;
i++;

ui->textBrowser->insertPlainText("Check symbols after C:\n\n\n");
while (i<text.length() && text[i] != 'D')
{

        char temp;
        temp = S.Pop();
                        for( ; ; )
                        {
                        QApplication::processEvents();
                        if(!started)break;


                        }
                        started = true;
        ui->textBrowser->insertPlainText("Stack:\n");
        for (j=show.size()-1;j>=0;j--)
        {
            ui->textBrowser->insertPlainText(QChar(show[j]));
            ui->textBrowser->insertPlainText("\n");
        }
        ui->textBrowser->insertPlainText("Try to check:\n");
        ui->textBrowser->insertPlainText(QChar(show[show.size()-1]));
        ui->textBrowser->insertPlainText(" and ");
        ui->textBrowser->insertPlainText(QChar(text[i]));
        ui->textBrowser->insertPlainText("\n");
        if (temp != text[i])
        {
            ui->textBrowser->insertPlainText("NOPE ");
            cout << temp << "    " << text[i] << '\n';
            cout << i << '\n';
            correct = false;
            break;
        }
```

```
                else
                {
                    show.pop_back();
                    i++;
                    ui->textBrowser->insertPlainText("Succeed check!:\n");
                }

            }
            if (correct==false)
            break;
            i++;
        }
        if (correct)
        {
        ui->textBrowser->insertPlainText("YES");
        }

}
```

Файл mainwindow.cpp

```
#include "mainwindow.h"
#include <QtGui>
#include <QApplication>
#include <iostream>
#include <fstream>
#include <string>
#include <cstring>
#include <clocale>
#include <windows.h>
#include "mainwindow.h"
using namespace std;




int main(int argc, char *argv[])
{
    QApplication a(argc, argv);
    MainWindow w;
    w.show();
    return a.exec();
}
```