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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: Рекурсия

Студентка гр. 8381 _____ Лисок М.А.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург

2019

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++. Разработать программу, использующую рекурсию, и сопоставить рекурсивное решение с итеративным решением задачи.

Задание.

Вариант №12. Построить синтаксический анализатор для понятия *скобки*.

скобки::=квадратные | *круглые* | *фигурные*

квадратные::=[*круглые фигурные*] | +

круглые::=(*фигурные квадратные*) | **–**

фигурные::={*квадратные круглые*} | 0

Основные теоретические положения.

Функция называется рекурсивной, если во время ее обработки возникает ее повторный вызов, либо непосредственно, либо косвенно, путем цепочки вызовов других функций.

Прямой (непосредственной) рекурсией является вызов функции внутри тела этой функции.

Косвенной рекурсией является рекурсия, осуществляющая рекурсивный вызов функции посредством цепочки вызова других функций. Все функции, входящие в цепочку, тоже считаются рекурсивными.

Одним из примеров программы, основанной на рекурсии является программа, выполняющая синтаксический анализ текста. Задача синтаксического анализатора – проверить правильность записи выражения.

Пусть требуется построить *синтаксический анализатор* понятия
скобки: скобки::=квадратные | *круглые*
квадратные::=[*круглые круглые*] | +
круглые::=(*квадратные квадратные*) **–**

В этом рекурсивном определении последовательности символов, называемой *скобки*, присутствуют две взаимно-рекурсивные части: *квадратные* определяются через *круглые*, и наоборот, *круглые* – через *квадратные*. В простейшем случае *квадратные* есть символ «+», а *круглые* есть символ «–». Другие примеры последовательностей, порождаемых этим

рекурсивным определением:

‘[– –]’, ‘(++)’, ‘[(++)([–(++)][– –])]’, ‘(+[(++)([–(++)][(+[– –])–])])’.

Синтаксическим анализатором назовём программу, которая определяет, является ли заданная (входная) последовательность символов *скобками* или нет. В случае ответа «нет» сообщается место и причина ошибки.

Выполнение работы.

Написание работы производилось на базе операционной системы MacOs в среде разработки QtCreator с использованием фреймворка Qt. Сборка, отладка и тестирование также производились в QtCreator.

Для реализации программы был разработан графический интерфейс с помощью встроенного в QtCreator UI-редактора. Он представляет из себя поле ввода, кнопку считывания и переноса в поле ввода информации из файла, запускающую синтаксический анализатор для текущей строки, поле вывода.

Были созданы слоты, обрабатывающие сигналы clicked() кнопок. При нажатии на кнопку считывания из файла, открывается файловый диалог с помощью функции класса QFileDialog – getOpenFileName(). После выбора файла для загрузки, создается объект класса ifstream для файла, и информация из него считывается в виджет LineEdit с именем inputLine.

После того, как были получены входные данные, начинается основная часть программы – синтаксический анализ. Т.к. нам известны символы, с которых могут начинаться скобки (‘[’, ‘+’, ‘(’, ‘-’, ‘{’, ‘0’) сначала проходит сравнение первого элемента текста с возможным началом скобок. Если символы совпали, то начинается проход по рекурсивным функциям, которые отвечают за определение того, скобки нам даны или нет. Рассмотрим эти функции подробнее.

1) `int square(string & arr, char& a, string & result, int & pos, int & depth)`

Эта функция проверяет, являются ли входные данные квадратными скобками. Данная функция принимает на вход ссылку на проверяемую строку arr, переменную char a, которая служит хранилищем для символа, с которым в данный момент проводится работа (анализ, вывод и т.д.), позиция текущего проверяемого символа в строке int & pos, которая передается по ссылке, так её

значение изменяется в течении программы, а так же глубина рекурсии, которая служит счётчиком для ширины отступа при выводе данных на консоль и в файл, которая соответствует глубине рекурсии.

Возвращаемое значение равно единице в том случае, если скобки соответствуют символам “+”, “0” или “-”, т.е. для их распознания не требуется вызова других функций. Ноль получается, если функция определения скобок дошла до конца (при этом из этой функции вызывались другие, анализирующие скобки) или если был обнаружен символ, из-за которого входные данные не могут являться скобками.

Работа функции основана на следующем принципе: вначале она анализирует текущий символ. Если он является “+”, то работу функции можно завершить, т.к. это квадратные скобки. Если этот символ – “[”, то функция должна обратиться к функции round (см. в п.2), и если та вернула единицу, то затем – к функции figure. Если возвращаемое значение вновь является единицей, то функция должна проверить следующий символ: если тот символ – “]”, то проанализированный текст является квадратными скобками и функция завершает работу и возвращает единицу. Если же при вызове функций round или figure был получен ноль, функция square завершает свою работу со значением 0, что означает, что входные данные не являются скобками.

2) `int round(string & arr, char& a, string & result, int & pos, int & depth)` – метод для определения круглых скобок;

3) `int figure(string & arr, char& a, string & result, int & pos, int & depth)` – метод для определения фигурных скобок;

Функции round и figure очень похожи на первую описанную функцию. Входные данные, работа с ними и возвращаемое значение – те же. Работа отличается лишь тем, что символы, которым должен соответствовать текущий, другие. Т.е. аналогом “+” для круглых и фигурных скобок являются “-” и “0” соответственно, аналогом “[” для круглых и фигурных скобок являются “(” и “{” соответственно и аналогом “]” для круглых и фигурных скобок являются “)” и “}” соответственно.

Для удобства в программе была введена ещё одна функция - `void symbol(string & arr, char& a, string & result, int & pos)`, которая отвечает за то, чтобы считывать по символу извходных данных по мере необходимости и выводить их в файл и на экран. Это позволяет сэкономить место и не прописывать много раз одни и те же строки.

После завершения работы функции в поле вывода переносится следующая информация (учитывается отмеченный флаг для дополнительной информации):

- Глубина рекурсии
- Проанализированная часть строки
- Если введенная строка не является скобками, сообщение с характером ошибки
- Сообщение о том, является ли введенная строка скобками

Вывод автоматически сохраняется в файле.

Тестирование программы.

Для начала рассмотрим примеры (см.табл.1), в которых входные данные являются скобками.

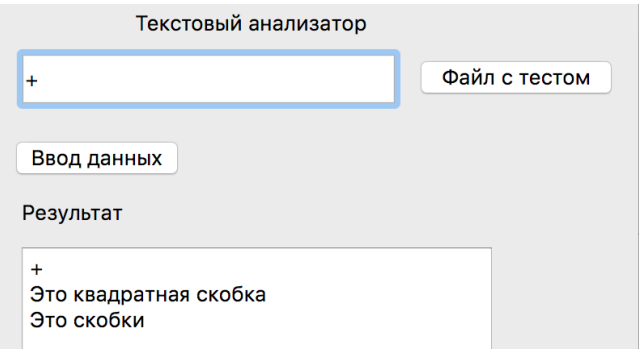
Таблица 1 – Корректные данные

Ввод данных	Вывод данных
+	+ Это квадратные скобки. Это скобки.

{[-0](0+)}	{ Это фигурная скобка. Выполняем проверку дальше... [Это квадратная скобка. Выполняем проверку дальше... - Это круглые скобки. 0 Это фигурные скобки.] Это квадратная скобка. (Это круглая скобка. Выполняем проверку дальше... 0 Это фигурные скобки. + Это квадратные скобки.) Это круглая скобка. } Это фигурная скобка. Это скобки.
{+-}	{ Это фигурная скобка. Выполняем проверку дальше... + Это квадратные скобки. - Это круглые скобки. } Это фигурная скобка. Это скобки.

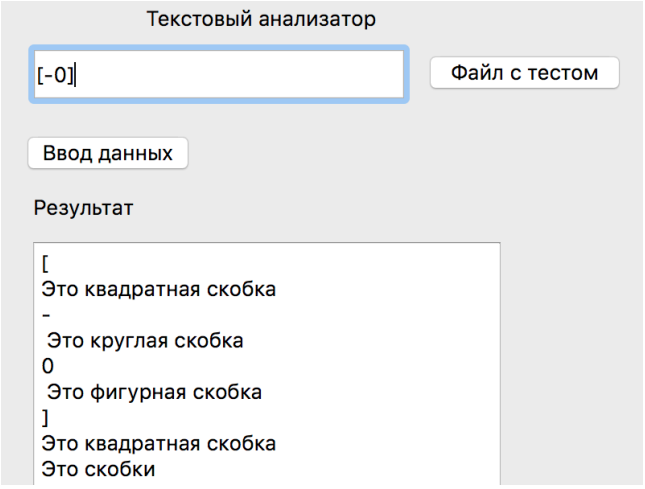
Ввод данных: ‘+’.

Вывод:



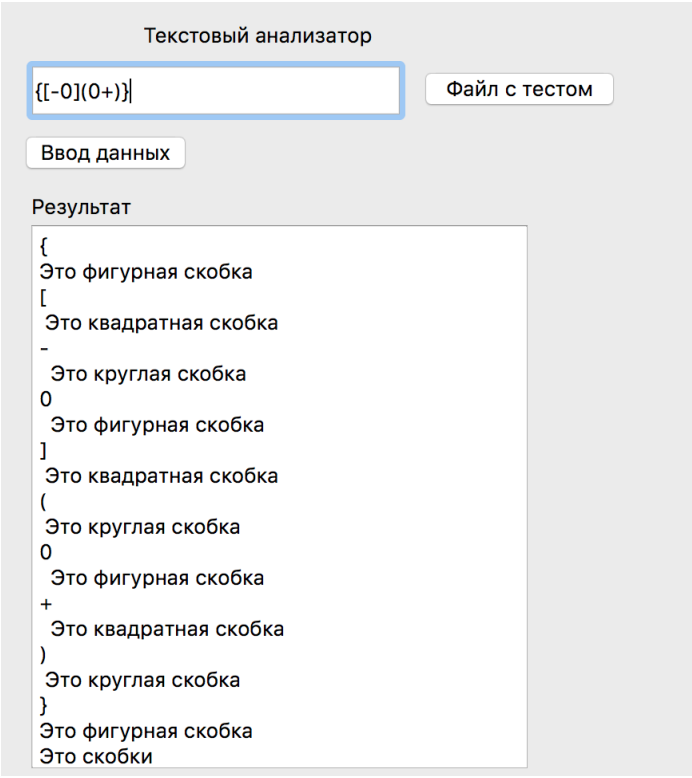
Ввод данных: ‘[-0]’.

Вывод:



Ввод данных: ‘{[-0](0+)}’.

Вывод:



Выводы.

В ходе выполнения лабораторной работы был изучен такой вид программ, как синтаксические анализаторы. Была реализована программа, которая анализирует строку рекурсивным методом, определяя соответствие определению.

Было выявлено, что для рекурсивных определений сложно найти оптимальное итеративное решение, при этом рекурсивный метод дает высокую эффективность.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: basicComponents.h

```
#ifndef BASICCOMPONENTS_H
#define BASICCOMPONENTS_H
#include <iostream>
#include <cstring>
#include <fstream>
using namespace std;
#endif // BASICCOMPONENTS_H
```

Название файла: components.h

```
#ifndef COMPONENTS_H
#define COMPONENTS_H
#include <QFileDialog>
#include <QMessageBox>
#include <QLabel>
#endif // COMPONENTS_H
```

Название файла: mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>

QT_BEGIN_NAMESPACE
namespace Ui { class MainWindow; }
QT_END_NAMESPACE

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    void checkInput();
    MainWindow(QWidget *parent = nullptr);
    ~MainWindow();

private slots:
    void on_fileInput_clicked();

    void on_consoleInput_clicked();

private:
    Ui::MainWindow *ui;
};
#endif // MAINWINDOW_H
```

Название файла: textAnalyze.h

```
#ifndef TEXTANALYZE_H
#define TEXTANALYZE_H
#include "basicComponents.h"
void brasket(string arr, string & result);
#endif // TEXTANALYZE_H
```

Название файла: main.cpp

```
#include "mainwindow.h"

#include <QApplication>

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

Название файла: mainwindow.cpp

```
#include "mainwindow.h"
#include "ui_mainwindow.h"
#include "components.h"
#include "textAnalyze.h"

#include <thread>
#include <qapplication>
MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
}

MainWindow::~MainWindow()
{
    delete ui;
}
void MainWindow::on_fileInput_clicked()
{
    QString fileName = QFileDialog::getOpenFileName(this,
            tr("Open TXT File"),  QDir::homePath(),
            tr("TXT text (*.txt);;All Files (*)"));
    ifstream sourceFile(fileName.toUtf8().constData());
    string input;
    sourceFile >> input;
    ui->inputLine->setText(QString::fromStdString(input));
    checkInput();
}

void MainWindow::on_consoleInput_clicked()
{
    checkInput();
}
void MainWindow::checkInput(){
    QString checkInput = ui->inputLine->text();
    string result;
    brasket(checkInput.toUtf8().constData(), result);
    ui->windowResult->setText(QString::fromStdString(result));
    ofstream resultFile("/Users/marialisok/прога/xcode/xcode/recursia/
recursia/result.txt");
    resultFile << result;
    resultFile.close();
    this->resize(445, 537);
    this->resize(444, 536);
}
```

Название файла: textAnalyze.cpp

```
#include "basicComponents.h"
void symbol(string&, char&, string&, int&);
void printDepth(string&, int&);
void printDepthReverse(string&, int&);
int square(string&, char&, string&, int&, int&);
int round(string&, char&, string&, int&, int&);
int figure(string&, char&, string&, int&, int&);
void symbol(string & arr, char& a, string & result, int & pos){
    a = arr[pos];
    result+=a;
    result+='\n';
    pos++;
}
void printDepth(string & result, int & depth){
    for(int i=0; i < depth; i++){
        result.append(" ");
    }
    depth++;
}
void printDepthReverse(string & result, int & depth){
    depth--;
    for(int i=0; i < depth; i++){
        result.append(" ");
    }
}
int square(string & arr, char& a, string & result, int & pos, int & depth){
    printDepth(result, depth);
    if(a == '+'){
        depth--;
        result.append("Это квадратная скобка\n");
        return 1;
    }
    if(a == '['){
        result.append("Это квадратная скобка\n");
        symbol(arr, a, result, pos);
        if(round(arr, a, result, pos, depth)){
            symbol(arr, a, result, pos);
            if(figure(arr, a, result, pos, depth)){
                symbol(arr, a, result, pos);
                if(a == ']'){
                    printDepthReverse(result, depth);
                    result.append("Это квадратная скобка\n");
                    return 1;
                }
            }
        }
    }
    else{
        result.append("Отсутствует + или [\n");
    }
    return 0;
}
int round(string & arr, char& a, string & result, int & pos, int & depth){
    printDepth(result, depth);
    if(a == '-'){
        depth--;
        result.append("Это круглая скобка\n");
        return 1;
    }
```

```
    if(a == '('){
        result.append("Это круглая скобка\n");
        symbol(arr, a, result, pos);
        if(figure(arr, a, result, pos, depth)){
            symbol(arr, a, result, pos);
            if(square(arr, a, result, pos, depth)){
                symbol(arr, a, result, pos);
                if(a == ')'){
                    printDepthReverse(result, depth);
                    result.append("Это круглая скобка\n");
                    return 1;
                }
            }
        }
    }
    else{
        result.append("Отсутствует - или (\n");
    }
    return 0;
}
int figure(string & arr, char& a, string & result, int & pos, int & depth){
    printDepth(result, depth);
    if(a == '0'){
        depth--;
        result.append("Это фигурная скобка\n");
        return 1;
    }
    if(a == '{'){
        result.append("Это фигурная скобка\n");
        symbol(arr, a, result, pos);
        if(square(arr, a, result, pos, depth)){
            symbol(arr, a, result, pos);
            if(round(arr, a, result, pos, depth)){
                symbol(arr, a, result, pos);
                if(a == '}'){
                    printDepthReverse(result, depth);
                    result.append("Это фигурная скобка\n");
                    return 1;
                }
            }
        }
    }
    else{
        result.append("Отсутствует 0 или {\n");
    }
    return 0;
}
void brasket(string arr, string & result){
    char a;
    int b=0;
    int depth=0;
    int pos=0;
    symbol(arr, a, result, pos);
    if(a == '+' || a == '['){
        b = square(arr, a, result, pos, depth);
    }
    else if(a == '-' || a == '('){
        b = round(arr, a, result, pos, depth);
    }
    else if(a == '0' || a == '{'){
        b = figure(arr, a, result, pos, depth);
```

```
    }
    if(b && pos != arr.length()){
        result.append("Присутствуют дополнительные символы. Это не
скобки\n");
    }
    else if(b){
        result.append("Это скобки\n");
    }
    else{
        result.append("Это не скобки\n");
    }
}
```

Название файла: mainwindow.ui

```
<?xml version="1.0" encoding="UTF-8"?>
<ui version="4.0">
 <class>MainWindow</class>
 <widget class="QMainWindow" name="MainWindow">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>444</width>
    <height>536</height>
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralwidget">
   <widget class="QPushButton" name="fileInput">
    <property name="geometry">
     <rect>
      <x>260</x>
      <y>60</y>
      <width>131</width>
      <height>31</height>
     </rect>
    </property>
    <property name="text">
     <string>Файл с тестом</string>
    </property>
   </widget>
   <widget class="QLabel" name="title">
    <property name="geometry">
     <rect>
      <x>90</x>
      <y>20</y>
      <width>161</width>
      <height>41</height>
     </rect>
    </property>
    <property name="text">
     <string>Текстовый анализатор</string>
    </property>
    <property name="textFormat">
     <enum>Qt::AutoText</enum>
    </property>
   </widget>
   <widget class="QLabel" name="result">
```

```
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>140</y>
      <width>91</width>
      <height>16</height>
     </rect>
    </property>
    <property name="text">
     <string>Результат</string>
    </property>
   </widget>
   <widget class="QPushButton" name="consoleInput">
    <property name="geometry">
     <rect>
      <x>10</x>
      <y>100</y>
      <width>113</width>
      <height>32</height>
     </rect>
    </property>
    <property name="text">
     <string>Ввод данных</string>
    </property>
   </widget>
   <widget class="QTextEdit" name="windowResult">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>160</y>
      <width>311</width>
      <height>341</height>
     </rect>
    </property>
   </widget>
   <widget class="QLineEdit" name="inputLine">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>60</y>
      <width>231</width>
      <height>31</height>
     </rect>
    </property>
   </widget>
  </widget>
  <widget class="QStatusBar" name="statusbar"/>
 </widget>
 <resources/>
 <connections/>
</ui>
```