**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**

Студент гр. 8381 Облизанов А.Д.

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.**

Построить синтаксический анализатор для понятия *скобки.*

*скобки*::=A | B | ( *скобки* *скобки* )

**Основные теоретические положения.**

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

Синтаксический анализатор для рекурсивно заданных скобок удобно реализовать с помощью рекурсии. Функция, проверяющая выражение на скобки, может запускать внутри себя функции, проверяющие часть выражения на скобки (или другие определения) в соответствии с выражением для скобок.

**Выполнение работы.**

Написание работы производилось на базе операционной системы Windows 10 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, флаг записи дополнительной информации в вывод, кнопку, запускающую синтаксический анализатор для текущей строки, поле вывода, а также кнопку выгрузки в файл информации, содержащейся в 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, запускается функция getInfo. В ней создается объект класса ifstream для файла, и информация из него считывается в строку, возвращаемую функцией. Далее текст выводится на виджет LineEdit с именем input.

При нажатии на кнопку запуска анализатора, текст из поля ввода, номер символа для анализа (изначально 0) и счетчик для подсчета глубины рекурсии передаются по ссылке в функцию checkString(). Эта функция вызывает функцию checkExp(), которая и производит рекурсивную проверку выражения.

Сначала проверяется, является ли текущий символ буквами ‘A’ или ‘B’. В таком случае счетчик номера символа увеличивается, и функция возвращает значение, соответствующее отсутствию ошибок (0).

Если функция встречает открывающую скобку, фиксируется ее наличие с помощью булевой переменной isOpen, увеличивается счетчик номера символа и два раза рекурсивно вызывается функция checkExp(). Если возвращаемое значение отлично от 0, оно возвращается, иначе функция продолжает работу.

Если встречена закрывающаяся скобка, то проверяется наличие открывающей скобки до (переменная isOpen), и, в случае ее отсутствия, возвращается код ошибки EMPTY (5), иначе возвращается OK (0).

Если закрывающая скобка не встречена, счетчик номера символа равен длине строки и есть наличие открывающей скобки, возвращается код ошибки NOCLOSE (4).

Если встречен любой другой символ, функция возвращает код ошибки SIGNERR (2).

В функции checkString() при получении кода 0 проверяется, была ли проверена вся строка. В случае, если счетчик номера символа не находится в конце строки, возвращается код ошибки SIGNERR (2). В противном случае функция возвращает значение checkExp().

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

При нажатии кнопки сохранения вывода в файл открывается файловый диалог с помощью функции класса QFileDialog – getSaveFileName(). Информация из поля вывода преобразуется в класс string и переносится в файл.

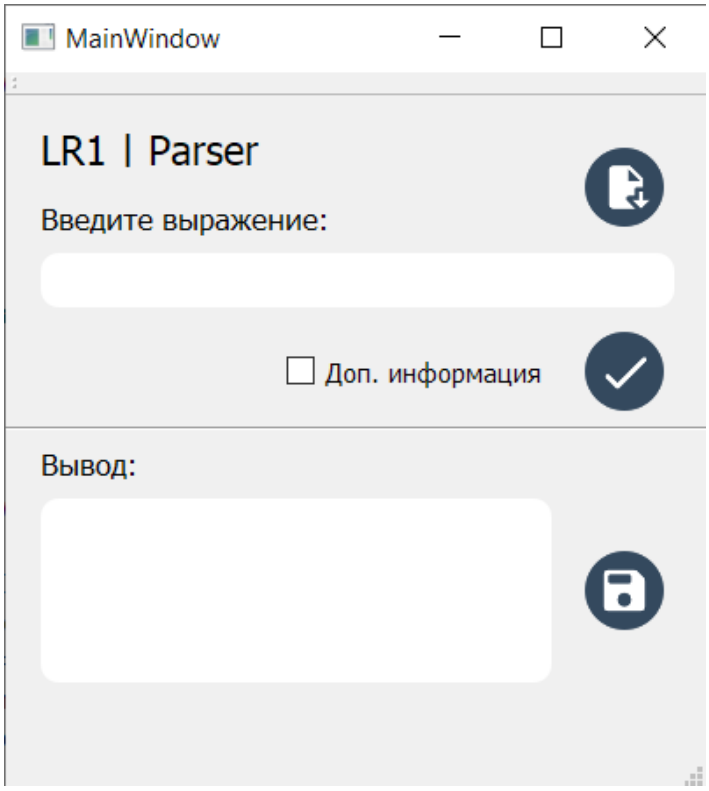
**Оценка эффективности алгоритма.**

Алгоритм, реализованный в программе, имеет линейную зависимость от длины строки, то есть сложность оценивается как O(n). Рассуждения отталкиваются от того, что функция checkExp() имеет сложность O(1) и запускается не более 1 раза для каждого символа строки.

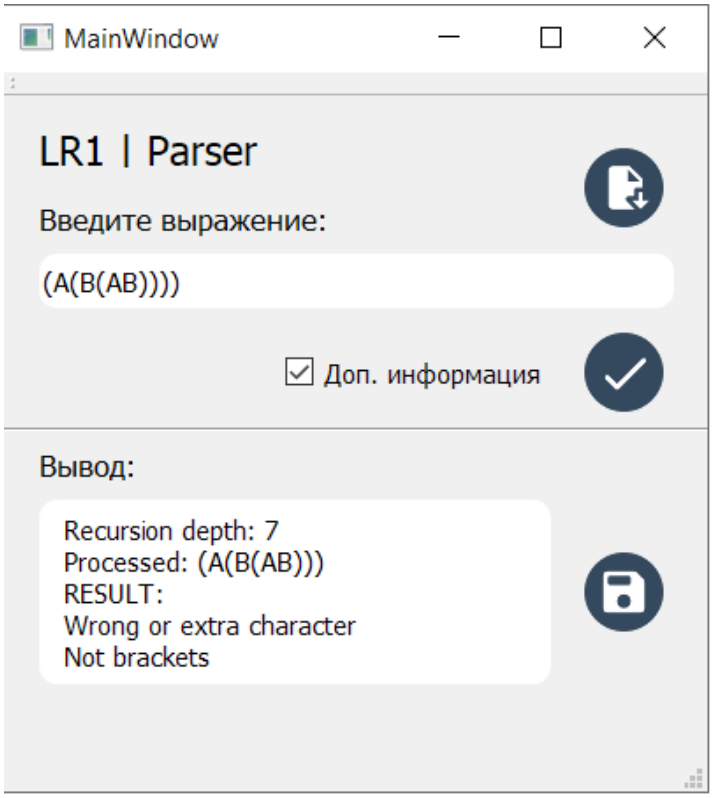
Ввиду рекурсивного алгоритма рост занимаемой памяти также растет линейно из-за создаваемых в функциях временных переменных. Для снижения потребления памяти переменные в функции checkExp() и checkString() передаются по ссылке.

**Тестирование программы.**

Внешний вид окна программы:




Пример выходных данных:



**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: 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"
#define EMPTY 5
#define NOCLOSE 4
#define NOOPEN 3
#define SIGNERR 2
#define ERR 1
#define OK 0

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

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

void MainWindow::on_start_clicked()
{
    string expIn = qPrintable(ui->input->text());
    unsigned long pos = 0;
    bool logs = ui->logs->isChecked();
    int funcNum = 1;
    string output;
    int result = checkString(expIn, pos, funcNum);
    if (logs)
    {
        output.append("Recursion depth: ");
        output.append(to_string(funcNum));
        output.append("\n");
        string expOut = expIn;
        expOut.erase(pos);
        output.append("Processed: ");
```

```
            output.append(expOut);
            output.append("\n");
        }
        output.append("RESULT:\n");
        switch (result)
        {
            case OK:
                output.append("Brackets\n");
            break;
            case SIGNERR:
                output.append("Wrong     or     extra     character\nNot
brackets\n");
            break;
            case NOOPEN:
                output.append("Missing opening bracket\nNot brackets\n");
            break;
            case NOCLOSE:
                output.append("Missing closing bracket\nNot brackets\n");
            break;
            case EMPTY:
                output.append("Missing brackets in  ()\nNot brackets\n");
            break;
        }
        ui->output->setText(QString::fromStdString(output));
    }

    int MainWindow::checkString(string &expIn, unsigned long &pos, int
&funcNum)
    {
        int result = checkExp(expIn, pos, funcNum);
        if (pos != expIn.length() && !result)
            return SIGNERR;
        return result;
    }

    int  MainWindow::checkExp(string  &expIn,  unsigned  long  &pos,  int
&funcNum)
    {
        bool isOpen = false;
        int k;
        if (expIn[pos] == 'A' || expIn[pos] == 'B')
        {
            pos++;
            return OK;
        }
        if (expIn[pos] == '(')
        {
            isOpen = true;
            pos++;
            for (int i=0; i<2; i++)
            {
                funcNum++;
                k = checkExp(expIn, pos, funcNum);
                if (k)
                    return k;
            }
```

```
        }
        if (expIn[pos] == ')')
        {
            if (isOpen)
            {
                pos++;
                return OK;
            }
            else
                return EMPTY;
        }
        if (pos == expIn.length() && isOpen)
            return NOCLOSE;
        return SIGNERR;
    }

    string MainWindow::getInfo(string filePath)
    {

        ifstream fin(filePath);
        string out;
        char buff[1010];
        fin.getline(buff, 1000);
        fin.close();
        out.append(buff);
        return out;
    }

    void MainWindow::on_file_clicked()
    {
        QString fileName = QFileDialog::getOpenFileName(this, tr("Choose
input file (TXT)"), QDir::homePath(), tr("*.txt"));
        if (QString::compare(fileName, QString()) != 0)
        {
            string expIn = getInfo(qPrintable(fileName));
            ui->input->setText(QString::fromStdString(expIn));
        }
    }

    void MainWindow::on_save_clicked()
    {
        QString filePath = QFileDialog::getSaveFileName(this, tr("Save
to TXT file"), QDir::homePath(), tr("*.txt"));
        if (QString::compare(filePath, QString()) != 0)
        {
            ofstream fout(qPrintable(filePath));
            fout << qPrintable(ui->output->text());
            fout.close();
        }
    }
```

Название файла: mainwindow.h

```
#ifndef MAINWINDOW_H
#define MAINWINDOW_H

#include <QMainWindow>
```

```
#include <QGraphicsEffect>
#include <QGraphicsView>
#include <QFileDialog>
#include <QStandardPaths>
#include <QtGui>
#include <QLabel>
#include <QColorDialog>
#include <QInputDialog>
#include <QMainWindow>
#include <QPushButton>
#include <QMessageBox>
#include <iostream>
#include <fstream>
using namespace std;


namespace Ui {
class MainWindow;
}

class MainWindow : public QMainWindow
{
    Q_OBJECT

public:
    explicit MainWindow(QWidget *parent = nullptr);
    ~MainWindow();
    string getInfo(string filePath);
    int checkExp(string &expIn, unsigned long &pos, int &funcNum);
    int checkString(string &expIn, unsigned long &pos, int &funcNum);

private slots:
    void on_start_clicked();

    void on_file_clicked();

    void on_save_clicked();

private:
    Ui::MainWindow *ui;
};

#endif // MAINWINDOW_H
```

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>400</width>
    <height>410</height>
```

```
   </rect>
  </property>
  <property name="windowTitle">
   <string>MainWindow</string>
  </property>
  <widget class="QWidget" name="centralWidget">
   <widget class="QPushButton" name="start">
    <property name="geometry">
     <rect>
      <x>330</x>
      <y>135</y>
      <width>45</width>
      <height>45</height>
     </rect>
    </property>
    <property name="whatsThis">
     <string>Проверить</string>
    </property>
    <property name="styleSheet">
     <string notr="true">border-image: url(:/ok.png);</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QLineEdit" name="input">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>90</y>
      <width>361</width>
      <height>31</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">border-radius: 10px;
font: 9pt &amp;quot;Tahoma&amp;quot;;</string>
    </property>
   </widget>
   <widget class="QLabel" name="name">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>20</y>
      <width>131</width>
      <height>21</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <family>Tahoma</family>
      <pointsize>14</pointsize>
      <weight>50</weight>
      <italic>false</italic>
      <bold>false</bold>
     </font>
```

```
    </property>
    <property name="styleSheet">
     <string notr="true">font: 14pt &amp;quot;Tahoma&amp;quot;;</string>
    </property>
    <property name="text">
     <string>LR1 | Parser</string>
    </property>
   </widget>
   <widget class="QLabel" name="output">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>230</y>
      <width>291</width>
      <height>105</height>
     </rect>
    </property>
    <property name="styleSheet">
     <string notr="true">background-color: white;
border-radius: 10px;
padding: 10px;
font: 9pt &amp;quot;Tahoma&amp;quot;;</string>
    </property>
    <property name="text">
     <string/>
    </property>
   </widget>
   <widget class="QLabel" name="message">
    <property name="geometry">
     <rect>
      <x>20</x>
      <y>60</y>
      <width>241</width>
      <height>21</height>
     </rect>
    </property>
    <property name="font">
     <font>
      <family>Tahoma</family>
      <pointsize>10</pointsize>
      <weight>50</weight>
      <italic>false</italic>
      <bold>false</bold>
     </font>
    </property>
    <property name="styleSheet">
     <string notr="true">font: 10pt &amp;quot;Tahoma&amp;quot;;</string>
    </property>
    <property name="text">
     <string>Введите выражение:</string>
    </property>
   </widget>
   <widget class="Line" name="line">
    <property name="geometry">
     <rect>
      <x>0</x>
```

```
    <y>180</y>
    <width>401</width>
    <height>20</height>
   </rect>
  </property>
  <property name="orientation">
   <enum>Qt::Horizontal</enum>
  </property>
 </widget>
 <widget class="QPushButton" name="file">
  <property name="geometry">
   <rect>
    <x>330</x>
    <y>30</y>
    <width>45</width>
    <height>45</height>
   </rect>
  </property>
  <property name="whatsThis">
   <string>Проверить</string>
  </property>
  <property name="styleSheet">
   <string notr="true">border-image: url(:/open.png);</string>
  </property>
  <property name="text">
   <string/>
  </property>
 </widget>
 <widget class="QPushButton" name="save">
  <property name="geometry">
   <rect>
    <x>330</x>
    <y>260</y>
    <width>45</width>
    <height>45</height>
   </rect>
  </property>
  <property name="whatsThis">
   <string>Проверить</string>
  </property>
  <property name="styleSheet">
   <string notr="true">border-image: url(:/savef.png);</string>
  </property>
  <property name="text">
   <string/>
  </property>
 </widget>
 <widget class="QLabel" name="message_2">
  <property name="geometry">
   <rect>
    <x>20</x>
    <y>200</y>
    <width>241</width>
    <height>21</height>
   </rect>
  </property>
```

```
   <property name="font">
    <font>
     <family>Tahoma</family>
     <pointsize>10</pointsize>
     <weight>50</weight>
     <italic>false</italic>
     <bold>false</bold>
    </font>
   </property>
   <property name="styleSheet">
    <string notr="true">font: 10pt &amp;quot;Tahoma&amp;quot;;</string>
   </property>
   <property name="text">
    <string>Вывод:</string>
   </property>
  </widget>
  <widget class="QCheckBox" name="logs">
   <property name="geometry">
    <rect>
     <x>160</x>
     <y>135</y>
     <width>151</width>
     <height>45</height>
    </rect>
   </property>
   <property name="styleSheet">
    <string notr="true">font: 9pt &amp;quot;Tahoma&amp;quot;;</string>
   </property>
   <property name="text">
    <string>Доп. информация</string>
   </property>
  </widget>
 </widget>
 <widget class="QMenuBar" name="menuBar">
  <property name="geometry">
   <rect>
    <x>0</x>
    <y>0</y>
    <width>400</width>
    <height>26</height>
   </rect>
  </property>
 </widget>
 <widget class="QToolBar" name="mainToolBar">
  <attribute name="toolBarArea">
   <enum>TopToolBarArea</enum>
  </attribute>
  <attribute name="toolBarBreak">
   <bool>false</bool>
  </attribute>
 </widget>
 <widget class="QStatusBar" name="statusBar"/>
</widget>
<layoutdefault spacing="6" margin="11"/>
<resources/>
<connections/>
```

```
</ui>
```