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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: Рекурсия

Студент гр. 7383 _____ Зуев Д.В.

Преподаватель _____ Размочева Н.В.

Санкт-Петербург

2018

ОГЛАВЛЕНИЕ

Цель работы

Цель работы: познакомиться с основными понятиями и приемами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования С++.

Формулировка задачи: Построить синтаксический анализатор для понятия *простое выражение.*

простое_выражение::=простой_идентификатор | (простое_выражение знак_операции простое_выражение)

простой_идентификатор::= буква

*знак_операции:: = – | + | **

Реализация задачи

В функции main выводится приглашение выбрать способ ввода входных данных либо выйти из программы. В случае выбора файла, программа устанавливает поток входных данных из файла test.txt, находящегося в той же директории. В случае ввода информации с консоли, создается файл test1.txt, функция main считывает строку с консоли, записывает эту строку в файл test1.txt, устанавливает поток входных данных из этого файла и удаляет файл.

Если файл пустой, то main выводит ошибку и начинает выполнение программы заново. Иначе вызывает функцию Expression. Если после выполнения поток входных данных не пустой или функция Expression вернула false, то main выводит ошибку.

Переменные, используемые в функции main:

- tmp — переменная, в которой хранится номер команды, отвечающий за выбор способа ввода данных или выхода из программы.
- infile — поток входных данных (так же используется в функциях Expression и Operation).
- str — вспомогательная строка для ввода с консоли.
- c — символ, считываемый из потока входных данных (так же используется в функциях Expression и Operation).
- b — булева переменная, говорящая о правильности входных данных.

Функция Expression проверяет чем является символьный аргумент. Если символ открывающая круглая скобка, то функция вызывает саму себя со следующим в потоке символом, функцию Operation и опять же саму себя, каждый раз проверяя не является ли поток пустым. Потом функция проверяет закрывающую скобку и, если никаких ошибок не встречает то возвращает true. Если символ буква то функция возвращает true. Иначе возвращает false.

Функция Operaion получает на вход символ и проверяет, является ли этот символ арифметической операцией. Если является то функция возвращает true, иначе false.

Функция Error вызывается остальными функциями, когда те встречают ошибку во входных данных, и выводит сообщение об этой ошибке на экран.

Переменные, используемые в функции Error:

- k — номер ошибки.

Тестирование

Программа была собрана в компиляторе G++ в среде разработки Qt creator в OS Linux Ubuntu 16.04 LTS.

В ходе тестирования была найдена ошибка — при вводе чисел неравных единице или двум (числа инициализации считывания из консоли или файла) программа выводила некорректный результат, который являлся оценкой простого выражения, которое не вводилось.

Для исправления программы перед выводом результата была добавлена проверка введенного числа, является ли оно единицей или двойкой.

Некорректный случай представлен в табл. 1.

Таблица 1 — Некорректный случай.

Входные данные	Выходные данные
4	Введено неверное число ЭТО НЕ ПРОСТОЕ ВЫРАЖЕНИЕ.

Корректные тестовые случаи представлены в приложении А.

Вывод

В ходе работы были получены навыки рекурсивного программирования на языке C++. Поскольку структуру логического выражения, описанного в формулировке задачи, можно представить в виде дерева, где все концевые узлы это буквы, а узлы ветвления - это знаки операции, которое является рекурсивной структурой. Поэтому рекурсивный подход является простым и удобным способом проверки входных данных. Глубина рекурсии не превышает глубину вложения скобок.

ПРИЛОЖЕНИЕ А.

ТЕСТОВЫЕ СЛУЧАИ

Таблица 2 — Корректные тестовые случаи

Входные данные	Выходные данные
1 (a*b)	(a*b) ЭТО ПРОСТОЕ ВЫРАЖЕНИЕ.
4	Введите верное число
2	(a*b err#4! - Отсутствует ')'.
3	
1 (((	(((err#5! - Очередное простое выражение — пустое ЭТО НЕ ПРОСТОЕ ВЫРАЖЕНИЕ.
1 (a	(a err#3! - Отсутствует или неверно введена операция ЭТО НЕ ПРОСТОЕ ВЫРАЖЕНИЕ.
1 (a*b)c	(a*b) err#1! - Лишние символы после простого выражения ЭТО НЕ ПРОСТОЕ ВЫРАЖЕНИЕ.
1 c(a*b)	c err#1! - Лишние символы после простого выражения ЭТО НЕ ПРОСТОЕ ВЫРАЖЕНИЕ.
1 ((a+b)(a-b))	((a+b)(err#3! - Отсутствует или неверно введена операция ЭТО НЕ ПРОСТОЕ ВЫРАЖЕНИЕ.
1	err#0! - Пустая входная строка ЭТО НЕ ПРОСТОЕ ВЫРАЖЕНИЕ.

ПРИЛОЖЕНИЕ Б.

ИСХОДНЫЙ КОД ПРОГРАММЫ

```
#include<iostream>
#include<fstream>
#include<cctype>
using namespace std;

bool Expression(ifstream &stream, char c);
bool Identifier(char c);
bool Operation(char c);
void Error(short k);

int main()
{
    char c = '0';
    bool b = false;
    short tmp = 0;
    cout<<"Анализатор понятия простое выражение:"<<endl;
    while(true)
    {
        cout<<"Введите 1, если желаете вводить выражение с
клавиатуры.\n"
              "Введите 2, если желаете брать выражение из файла
test.txt.\n"
              "Введите, 3 если хотите закончить работу."<<endl;
        cin>>tmp;
        switch(tmp)
        {
            case 1:
            {
                cout<<"Введите выражение"<<endl;
                char str[100];
                cin.getline(str,1);
                cin.getline(str,100);
                ofstream fout;
                fout.open("test1.txt");
                fout<<str;
                fout.close();
                ifstream infile("test1.txt");
                do
                    infile>>c;
                while(isspace(c));
                if(!c)
                    Error(0);
                else
                {
                    cout<<c;
                    b = Expression(infile, c);
                    while(isspace(c))
                        infile>>c;
```

```
                    if(infile>>c && b)
                    {
                        Error(1);
                        b = false;
                    }
                }
                remove("test1.txt");
                break;
            }
            case 2:
            {
                ifstream infile("test.txt");
                if(!infile)
                {
                    cout<<"Входной файл не открыт."<<endl;
                    b = false;
                }
                do
                    infile>>c;
                while(isspace(c));
                if(!c)
                    Error(0);
                else
                {
                    cout<<c;
                    b = Expression(infile, c);
                    while(isspace(c))
                        infile>>c;
                    if(infile>>c && b)
                    {
                        Error(1);
                        b = false;
                    }
                }
                break;
            }
            case 3:
                break;
            default:
            {
                cout<<"Введено неверное число"<<endl;
                break;
            }
        }
        if(tmp == 1 || tmp == 2)
            if(b)
                cout<<endl<<"ЭТО ПРОСТОЕ ВЫРАЖЕНИЕ."<<endl;
            else
                cout<<"ЭТО НЕ ПРОСТОЕ ВЫРАЖЕНИЕ."<<endl;
    }

}
```

```
bool Expression(ifstream &infile, char c)
{
        if(c == '(')
        {
        while(isspace(c))
            infile>>c;
        if(infile>>c)
                {
                        cout<<c;
            if(!Expression(infile, c))
                return false;
                }
                else
                {
            Error(5);
                        return false;
                }
        while(isspace(c))
            infile>>c;
        if(infile>>c)
        {
            cout<<c;
            if(!Operation(c))
                return false;
        }
        else
        {
            Error(3);
            return false;
        }
        while(isspace(c))
            infile>>c;
        if(infile>>c)
        {
            cout<<c;
            if(!Expression(infile, c))
                    return false;
        }
        else
        {
            Error(5);
            return false;
        }
        while(isspace(c))
            infile>>c;
        if(infile>>c)
                {
                        cout<<c;
            if (c == ')')
                return true;
            else
            {
```

```
                Error(4);
                return false;
            }
                }
                else
                {
            Error(4);
                        return false;
                }

        }
    else if(isalpha(c))
        return true;
    else
    {
        Error(2);
        return false;
    }
}

bool Operation(char c)
{
        if( c == '+' || c == '-' || c == '*')
                return true;
    else
    {
        Error(3);
        return false;
    }
}

void Error (short k)
{
    cout << endl << "err#" << k;
    switch (k) {
        case 0: cout << "! - Пустая входная строка" << endl;
break;
        case 1: cout << "! - Лишние символы после простого
выражения" << endl; break;
        case 2: cout << "! - Недопустимый начальный символ" <<
endl; break;
        case 3: cout << "! - Отсутствует или неверно введена
операция" << endl; break;
        case 4: cout << "! - Отсутствует ')'." << endl; break;
        case 5: cout << "! - Очередное простое выражение —
пустое" << endl; break;
        default : cout << "! - ...";break;
    };
}
```