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

ОТЧЕТ

по лабораторной работе №1

по дисциплине «Алгоритмы и структуры данных»

Тема: Рекурсия

Студентка гр. 8304 _____ Чечеткина К.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2019

Цель работы.

Изучить основы рекурсии и составления эффективных алгоритмов.

Постановка задачи.

1) Разработать программу, использующую рекурсию;
2) Сопоставить рекурсивное и итеративное решение задачи;
3) Сделать вывод о целесообразности и эффективности рекурсивного подхода для решения данной задачи

Вариант 20.

Построить синтаксический анализатор понятия *список_параметров*.

список_параметров::=параметр | параметр, список_параметров

параметр::=имя=цифра цифра | имя=(список_параметров)

имя::=буква буква буква

Описание алгоритма.

Для решения поставленной задачи была реализована рекурсивная функция Recursion, которая анализирует строку на вхождение в нее параметра, выделяет его из строки и рекурсивно вызывает саму себя. Сначала функция проверяет соответствует ли отрезок в 6 символов простейшему условию *список_параметров::=буква буква буква=цифра цифра.* Далее функция проверяет более сложные условия с вложенными простейшими. При нахождении второго рекурсивно обращается сама

к себе, сдвигая границы отрезка. Функция работает, пока правая и левая границы не совпадут или не найдется отрезок не соответствующий условию.

Спецификация программы.

Программа предназначена для синтаксического анализа выражения методом рекурсии. Программа написана на языке C++. Входными данными является строка. Выходными данными являются обрабатываемые отрезки и конечно подтверждение выполнения задачи.

Описание функций.

1) Функция read().

Функция считывает строку и считает ее длину.

2) Функция test(int ,int).

Вывод обрабатываемые отрезки строки.

3)Функция Recursion(int, int).

Основная функция программы.

Вывод.

Был получен опыт работы с рекурсией и с построением синтаксического анализатора. На мой взгляд, итеративное решение поставленной задачи более эффективно.

ПРИЛОЖЕНИЕ

1. ТЕСТИРОВАНИЕ:

Работа программы для строки are=(qwe=12,hot=(cat=34,bal=77))

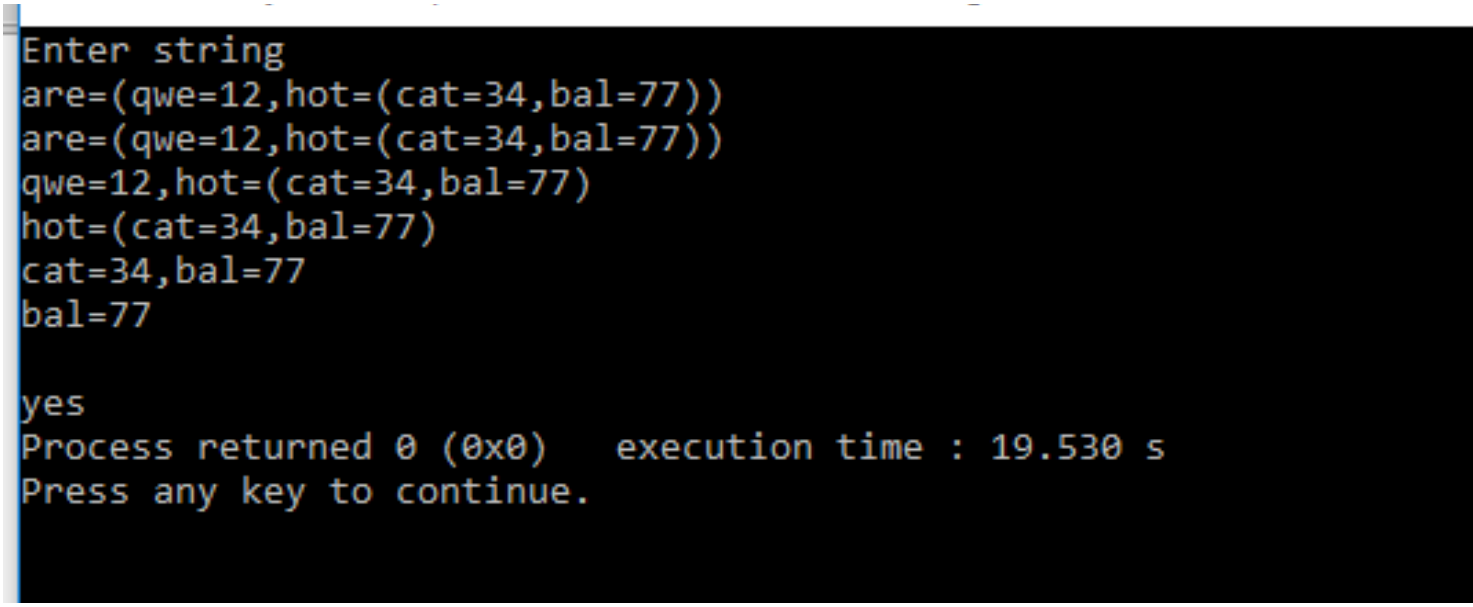

Таблица ввода/вывода тестирования программы

Входная строка	Вывод программы
are=12	yes
are=123	no
art=66,qwe=77	yes
solt=45	no
are=(two=45,one=34)	yes
are=(two=45,one=344)	no
are=(twooo=45,one=34)	no
are=22,rer=(sre=67,ftr=88)	yes
are=22,rer=(sre=67,ftr)	no
are=(are=12,tre=34),res=54	no
wet=78,ret=(day=66,ree=(tur=	yes

33,les=(qwe=09,jet=78)))	

2. ИСХОДНЫЙ КОД:

```
#include <iostream> // std::cin, std::cout, std::ostream
#include <string> // std::string и сопутствующие функции

class MyRecursion{
private:
    std::string str;
public:
    int length;
    int Recursion(int, int); //основная рекурсивная функция
    void read(); //считывание строки
    void test(int,int ); //вывод промежуточных результатов
};
void MyRecursion::read() { //функция считывает введенную
    std::cin >> str;                пользователем строку
    length = str.length()-1;
}

void MyRecursion::test(int leftBorder,int rightBorder){
    for(int i=leftBorder;i<=rightBorder;i++)
        std::cout << str[i];
    std::cout<<"\n";
}
```

```
int MyRecursion::Recursion(int leftBorder, int rightBorder) {
    test(leftBorder,rightBorder); //функция выводит промежуточные
                                  результаты
    if (leftBorder>=rightBorder)
    {
        std::cout<<"yes";
        return 0;
    }

    bool flag = true;
    int symbCount = 0;

    for (symbCount; symbCount < 3; symbCount++) {
        if (!isalpha(str[leftBorder+symbCount])) {
            flag = false;
        }
    }

    if (str[leftBorder+symbCount] != '=') {
        flag = false;
    }
    symbCount++;

    for (symbCount; symbCount < 6; symbCount++) {
```

```
        if (!isalnum(str[leftBorder+symbCount])) {
            flag = false;
        }
    }

    if (flag){
        if ((isalnum(str[leftBorder+6])!=0 || isalpha(str[leftBorder+6]!=0))){
            std::cout<<"no";
            return 0;
        }
        Recursion(leftBorder+((str[leftBorder+6] = ',')? 7:6),rightBorder);
        return 0;
    }

    flag = true;
    symbCount = 0;

    for (symbCount; symbCount < 3; symbCount++) {
        if (!isalpha(str[leftBorder+symbCount])) {
            flag = false;
        }
    }

    if ((!flag) || (str[leftBorder+3] != '=') || (str[leftBorder+4] != '(') ||
(str[rightBorder] != ')')){
```

```
        flag = false;
    }

    if (flag){
        Recursion(leftBorder+5,rightBorder-1);
        return 0;
    } else {
        std::cout<<"no";
        return 0;
    }
}
int main() {
    MyRecursion Text;
    std::cout<<"Enter string"<<"\n";
    Text.read();
    Text.Recursion(0,Text.length);
    return 0;
}
```