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. 7382 Давкаева В. С.
Преподаватель Фирсов М. А.

Санкт-Петербург
2018

Задание.

15. Построить синтаксический анализатор для понятия *скобки.*

*скобки::=*A | A (*ряд_скобок*)

ряд_скобок::= скобки | *скобки* ; *ряд_скобок*

Пояснение задания.

Требуется, чтобы программа считывала выражение с консоли , затем, используя рекурсивную функцию, проверяющую соответствие переданного выражения понятию «скобки» или понятию «ряд_скобок» получить ответ и вывести его на экран. При этом на экран должны выводиться промежуточные результаты обработки и глубина текущего вызова рекурсивной функции.

Описание алгоритма.

1. Происходит получение выражения от пользователя;

2. Строка передается в функцию *SyntaxAnalys*;

3. В функции *SyntaxAnalys* происходит подсчёт символов полученной строки, путём увеличения переменной *size,* и одновременно проверка на корректность введённых символов: если введён символ, отличный от «(»,«)»,«A» или «;», то переменной *brackets* присваивается значение 0;

4. Если в переданной строке нет символов (переменная *size*=0), *brackets* присваивается значение 0, дальнейшие условия в функции *SyntaxAnalys* не выполняются;

5. Если в строке встретились некорректные символы (переменная *brackets*==0), дальнейшие условия в функции функции *SyntaxAnalys* не выполняются;

6. Если введенные символы корректны (переменная *brackets*==1), происходит проверка на то, что первый символ «A» .Если это так и количество символов *size* равно 1, то происходит выход из функции *SyntaxAnalys*;

7. Проходит проверка на второй символ. Если size > 1 а второй символ не «(» то данное выражение не скобки.

8. Если первый символ «(», то происходит вызов рекурсивной функции *AnalysOfBrackets*. В нее передаётся указатель на обрабатываемую строку (*str), размер строки size, номер текущего элемента 1 и глубина рекурсии 0;

9. В рекурсивной функции *AnalysOfBracket* если весь массив не прочитан то считываем следующей элемент current пока 0 < current < size :

9.1. Если текущий символ с номером current равен «(» :

9.1.1. Происходит вызов рекурсивной функции *AnalysOfBracket* (её возвращаемое значение присваивается счетчику current), в которую передаётся указатель на обрабатываемую строку (**str*), размер строки *size*, номер текущего элемента *current* и глубина рекурсии *deep*(переход к п. 10);

9.1.2. Если вся строка прочитана, а глубина рекурсии == 1 (не равна 0) то строка - не скобки, и функция возвращает значении current умноженное на -1.

9.2. Если текущий символ с номером current равен «)» :

9.2.1. Если current==size-1, deep ==1 а на месте current-1 не находятся символы «(» или «;» , то функция возвращает значение current и строка является скобками.

9.2.2. Если current==size-1, или а на месте current-1 не находятся символы «(» или «;» , или глубина рекурсии deep<2, то функция возвращается значение current умноженное на -1.

9.3. Если текущий символ с номером current равен «A» :

9.3.1. Если предыдущий символ является символом «A», рекурсивная функция возвращает значение current умноженное на -1.

9.4. Если текущий символ с номером current равен «;» :

9.4.1. Если предшествующий этому элемент имеет значение «;» или «(», функция возвращает значение current*-1;

10. Если вся строка прочитана, а глубина рекурсии == 1 (не равна 0), то вернуть current умноженное на -1.

Рекурсивная функция AnalysOfBrackets возвращает значение current.

11. В функции *SyntaxAnalys*, в зависимости от возвращаемого значения, выводится ответ на экран.

Описание функций.

int AnalysOfBracket(char *str (указатель на обрабатываемую строку)**, int size** (размер строки), **int deep** (глубина рекурсии)**, int current** (номер текущего элемента)**)** – рекурсивная функция, возвращающая значение типа int : отрицательное в случае, если выражение не относится к понятию «скобки (номер элемента с ошибкой умноженный на -1), и положительное значение current равное размеру строки (номер последнего обработанного элемента), если выражение относится к понятию «скобки».

void SyntaxAnalys(char *str (указатель на обрабатываемую строку)**)** – функция, предназначенная для вывода решения программы на экран.

Тестирование.

№	Исходное выражение:	Результат:
1	\n	Error: No symbols! Not brackets!
2	«A»	A Is brackets!
3	«DED»	Error: Incorrect symbol! Not brackets!
4	«A(A(A))»	It is brackets! A(A(A))
5	«(A(AB))(»	A(AB Error: Incorrect symbol!

		Not brackets!
6	**«A(A(A(A);A))»**	**см. рис. 1**
7.	**«A(A(A)))»**	**см. рис. 2**
8.	**«A(A;(A(;)»**	**см. рис. 3**

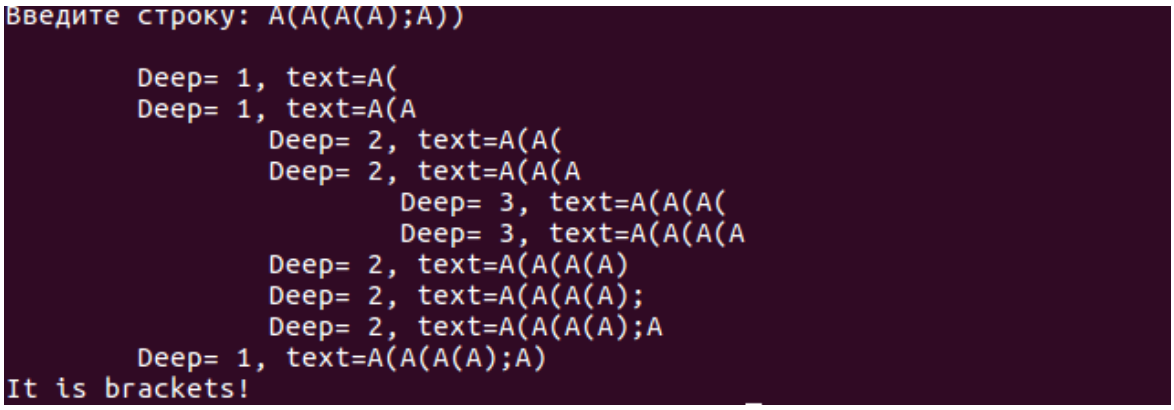

Рисунок 1-Тест 6

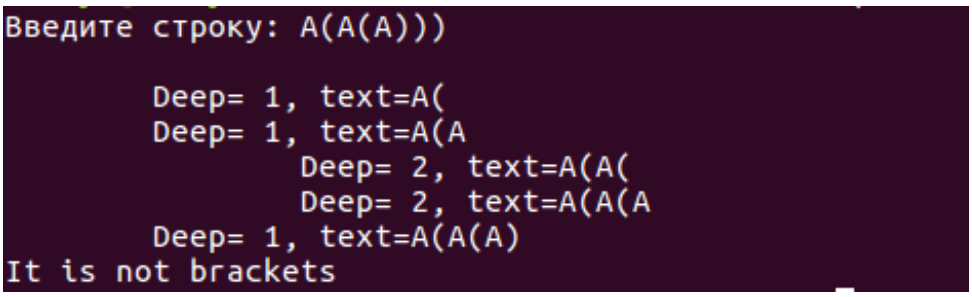

Рисунок 2-Тест 7

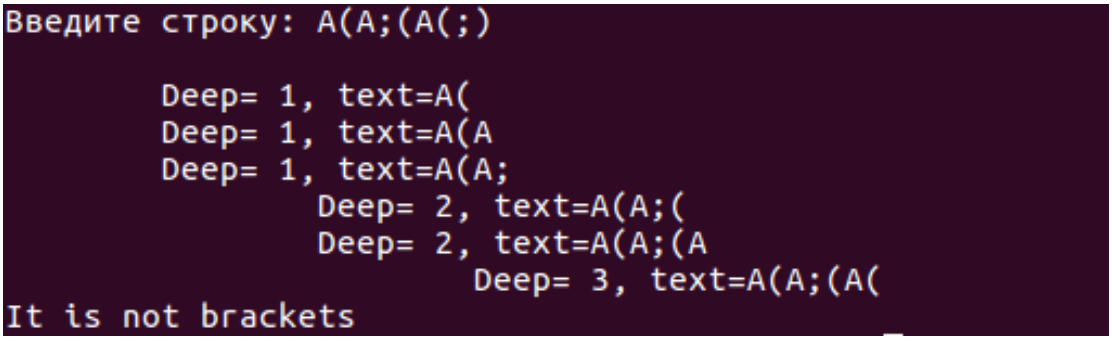

Рисунок 3-Тест 8

Вывод.

В ходе выполнения данной лабораторной работы было проведено ознакомление с основными понятиями и приёмами рекурсивного программирования, получены навыки программирования рекурсивных функция на языке программирования Си. В результате выполненной работы был создан синтаксический анализатор, который определяет: относится ли введённое выражение к понятию «скобки».

Приложение 1. Код программы.

```
#include <stdio.h>

int AnalysOfBracket(char *str, int size, int deep, int current)//возвращает
номер текущего элемента, если отрицательный - то не скобки
{
        deep++;//Уровень глубины рекурсии повышается на 1
                while(current>0 && current<size-1)
                {

                                for(int i=0; i<deep; i++)
                                        printf("\t");
                                printf("Deep= %d, text=", deep);
                                for (int i = 0; i<=current; i++)
                                        printf("%c", str[i]);
                                printf("\n");
                                current++;
                                switch(str[current])
                                {
                                        case('('):
                                                current=AnalysOfBracket(str, size,
deep, current);//Если встречаем скобку - рекурсивно вызываем функцию анализа
скобки
                                                if(current==size-1&&deep==1)//Если
вся строка прочитана, а глубина рекурсии == 1 (не равна 0)
                                                        return -1*current;//то
строка - не скобки
                                        break;
                                        case(')'):
                                                if(current==size-1 && deep==1 &&
str[current-1]!=';' && str[current-1]!='(')
                                                        return current;
                                                else if(current==size-1 ||
str[current-1]==';' || deep<2 || str[current-1]=='(') //символы ';' и '(' не
могут находиться перед ')'
                                                        current*=-1;
                                                return current;
                                        break;
                                        case('A'):
                                                if(str[current-1]=='A')//символу
'A' не может предшествовать символ 'A'
                                                        current*=-1;;
                                        break;
                                        case(';'):
                                                if(str[current-
1]==';'||str[current-1]=='(')//символы ';' и '(' не могут находиться перед ';'
                                                        current*=-1;
                                        break;

                                }

                }
                if(current==size-1&&deep==1)//Если вся строка прочитана, а
глубина рекурсии == 1 (не равна 0)
                         return -1*current;//то строка - не скобки
                return current;
}

void SyntaxAnalys(char *str)//функция для обработки введенной строки
{
    int current=0;//счетчик текущего элемента
```

```
    int size=0;//счетчик размера введенного массива
    int brackets=1;//1-true, 0-false
    int result;//1-true 0-false


    while ((str[current] != '\n') &&(str[current] != '\0') && (brackets ==
1))//проверка на неверный символ
    {
        if ((str[current] != ';') && (str[current] != 'A') && (str[current] !=
'(') && (str[current] != ')'))
            brackets=0;
         size++;
         current++;
    }

    if(size==0)//если нет введенных сиволов
    {
        printf("\nError: No symbols!\n");
        printf("Not brackets!\n");
        brackets=0;
    }

    else if(brackets==0)//если некорректный символ
    {
        for (int i = 0; i < size; i++)
            printf("%c", str[i]);
        printf("\nError: Incorrect symbol!\n");
        printf("Not brackets!\n");
    }

    if(brackets==1)
    {
        if(str[0]!='A')//если первый символ не A
        {
               printf("%c\n",str[0] );
                printf("\nIsn't brackets!\n");
        }
        else if(size==1)
        {
               printf("A\n");
               printf("It is brackets!\n");
        }
        else if(str[1]!='(')//Если второй символ не '('
        {
               printf("%c%c\n",str[0], str[1] );
                printf("\nIsn't brackets!\n");
        }
        else
        {
               current=1;
               result=AnalysOfBracket(str,size,0,current);
               if(result>0)
                      printf("It is brackets!\n");
               else
               {
                      result*=-1;
                      printf("It is not brackets\n");
               }
        }


    }
```

```
}

int main()
{
        char str[100];//массив для запроса

        printf("Введите строку: ");
         fgets(str,100,stdin);
         printf("\n");
         SyntaxAnalys(str);

return 0;
```