

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

ОТЧЕТ
по практической работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Рекурсия

Студент гр. 7382

Находько А.Ю.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2018

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций.

Задание.

Построить синтаксический анализатор для понятия скобки.

скобки::=A | скобка скобки

скобка::= (B скобки)

Пояснение задачи.

Требуется, чтобы программа считывала текст с консоли, используя рекурсивную функцию, которая проверяет соответствие выражения понятию “скобки” или “скобка”, получить ответ и вывести результат анализа на экран.

Выполнение работы.

- 1) Выделим память под строку и попросим пользователя ввести текст.
- 2) Произведём подсчёт символов введённой строки.
- 3) Передадим текст в функцию `Correct_Place_Bracket` с аргументами: длина строки, сам текст.
- 4) Описание функции `Correct_Place_Bracket`. После получения текста в функцию, заводим переменную `bracket_count`, также элемент который будет указывать на первый символ строки. Запускается цикл, который пройдёт до конца текста. Если встретим открывающуюся скобку, то увеличиваем `bracket_count` на 1, если закрывающуюся, то уменьшаем `bracket_count` на 1. Если количество открывающихся и закрывающихся скобок окажется одинаковым, тогда функция вернёт значение 1.
- 5) Проверим первый символ текста --- если он не является ‘A’, не является ‘(’ и для каждой открывающейся скобки не найдётся закрывающейся скобки, то данный текст не является верным.
- 6) Описание функции `Find_Close_Brackets`. Принимает в качестве аргументов

длину текста и сам текст. Заводим переменную `bracket_count=1`, также элемент который будет указывать на второй символ строки. Запускается цикл, который пройдёт до конца текста. Если встретим открывающуюся скобку, то увеличиваем `bracket_count` на 1, если закрывающуюся, то уменьшаем `bracket_count` на 1, если достигнем конца текста, то возвращение функцией 0. Функция возвращает разность позиции текущего элемента и начального.

7) Вызовем функцию `rec_analis` для проверки текста на понятие “скобки”. В качестве аргументов функция принимает длину текста и сам текст.

8) Описание функции `rec_analis`. Получив аргументы, функция производит проверку длины текста. Если в рекурсивную функцию поступают данные, в которых длина текста равна 1 и текущий элемент ‘А’, то возвращает значение 1. Также, если текущий элемент текста ‘(’, а последующий ‘В’, то выполним проверку на наличие закрывающихся скобок для всех открывающихся, если проверка не пройдена, то выходим со значением 0. Создадим 2 переменные `corr_section` и `trans_section`. Данные переменные могут принимать значение 1 либо 0, в зависимости от выполнения рекурсивной функции. В случае когда введённый текст успешно прошёл анализ на понятие скобки, в `main` возвращается 1.

9) Принцип работы рекурсии в анализаторе.

```
int corr_section=rec_analis(Find_Close_Bracket(len_of_text,text)-3,text+2);
```

Используя данный вызов функции, функция обнаруживает участок текста, в котором наблюдается правильная постановка скобок. Затем начинает анализ текста начиная с третьего(по счёту) элемента --- т.к. два предыдущих элемента являлись ‘(’ и ‘В’, диапазон проверки не превышает значения длины правильного участка -3. В процессе проверки найденный изначально участок с правильной постановкой скобок будет уменьшаться. В случае когда одна пара скобок будет проанализирована, функция использует:

```
Int trans_section=rec_analis(len_of_text-Find_Close_Bracket(len_of_text,text),  
text+Find_Close_Bracket(len_of_text,text));
```

для перехода на не анализированную часть. При последующем обращении к функции `rec_analis` произойдёт вызов функции `trans_section` переменной `corr_section`. Функция будет работать пока не будет проанализирован весь текст.

10) Программа выведет результат исследования, произведёт очищение памяти.

Описание функций.

`int Find_Close_Bracket(int len_of_text, char *text)` – функция для поиска позиции закрывающей скобки для первой открывающей скобки в тексте.

`int rec_analis(int len_of_text, char *text)` – рекурсивная функция для анализа строки на понятие “скобки”. Функция возвращает 1, если введенный текст удовлетворяет понятию “скобки” и 0, если нет.

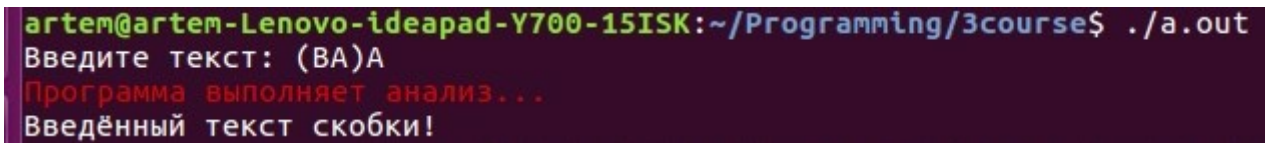
Тестирование.

Таблица тестирования программы.

№ теста	Входные данные:	Результат:
1	(BA)A	Введенный текст скобки!
2	(B(BA)(BA)A)(BA)(BA)A	Введенный текст скобки!
3	(B(B(BA)A)A)A	Введенный текст скобки!
4	(BA(BA)A)A	Введенный текст не является скобками!
5	(AB)A	Введенный текст не является скобками!
6	(B(BA)(BA)(BA)A)	Введенный текст не является скобками!
7	(B(BA)(B((((((Некорректный ввод данных!

Иллюстрация работы тестирования:

1. Ввод данных: (BA)A



```
artem@artem-Lenovo-ideapad-Y700-15ISK:~/Programming/3course$ ./a.out
Введите текст: (BA)A
Программа выполняет анализ...
Введённый текст скобки!
```

Рассмотрим ход работы программы на примере теста №1:

- 1) Строка проходит проверку на равенство открывающихся и закрывающихся скобок.
- 2) Проходит проверку на пустоту строки.
- 3) Начинаем анализ строки, т.к. длина строки передаваемая в функцию не равна 1 и первый элемент строки равен “(”, а следующий за ним “B”, то функцией Find_Close_Bracket выполняем поиск закрывающей первую открывающуюся скобку и возвращаем её позицию+1.
- 4) Переменная corr_section рекурсивно вызывает функцию res_analis. Которая в качестве аргументов примет:
 - длину участка в “правильных” скобках-3. В рассматриваемом случае это значение будет равно 4-3=1;
 - элемент с которого будет продолжаться анализ (начальный элемент + 2). В данном случае это элемент “A”.
- 5) При рекурсивном вызове с такими аргументами, функция вернёт 1, т.к. длина текста равна 1 и текущий элемент “A”.
- 6) Происходит рекурсивный вызов функции res_analis переменной trans_section. В качестве аргументов функция получит:
 - длина текста – значение возвращаемое функцией Find_Close_Bracket. В данном случае 5-4=1;
 - элемент для анализа, в рассматриваемом случае “A”.
- 7) Т.к. длина текста равна 1 и текущий элемент “A”, то значение переменной trans_section будет равно 1.
- 8) Возврат в main 1 в качестве результата выполнения функции res_analis.

9) Анализ выполнен успешно, вывод соответствующего текста и выход.

2.Ввод данных: (B(BA)(BA)A)(BA)(BA)A

```
artem@artem-Lenovo-ideapad-Y700-15ISK:~/Programming/3course$ ./a.out
Введите текст: (B(BA)(BA)A)(BA)(BA)A
Программа выполняет анализ...
Введённый текст скобки!
```

3.Ввод данных: (B(B(BA)A)A)A

```
artem@artem-Lenovo-ideapad-Y700-15ISK:~/Programming/3course$ ./a.out
Введите текст: (B(B(BA)A)A)A
Программа выполняет анализ...
Введённый текст скобки!
```

4.Ввод данных: (BA(BA)A)A

```
artem@artem-Lenovo-ideapad-Y700-15ISK:~/Programming/3course$ ./a.out
Введите текст: (BA(BA)A)A
Программа выполняет анализ...
Введённый текст не является скобками!
```

5.Ввод данных: (AB)A

```
artem@artem-Lenovo-ideapad-Y700-15ISK:~/Programming/3course$ ./a.out
Введите текст: (AB)A
Программа выполняет анализ...
Введённый текст не является скобками!
```

5. Ввод данных: (BA)A(BA)A

```
artem@artem-Lenovo-ideapad-Y700-15ISK:~/Programming/3course$ ./a.out
Введите текст: (BA)A(BA)A
Программа выполняет анализ...
Введённый текст не является скобками!
```

6. Ввод данных: (B(BA)(BA)(BA)A)

```
artem@artem-Lenovo-ideapad-Y700-15ISK:~/Programming/3course$ ./a.out
Введите текст: (B(BA)(BA)(BA)A)
Программа выполняет анализ...
Введённый текст не является скобками!
```

7. (B(BA)(B((((((

```
artem@artem-Lenovo-ideapad-Y700-15ISK:~/Programming/3course$ ./a.out
Введите текст: (B(BA)(B((((((
Программа выполняет анализ...
Некорректный ввод данных!
```

Выводы.

Ознакомился с основными понятиями и приёмами рекурсивного программирования, получил навыки программирования рекурсивных процедур и функций на примере написания синтаксического анализатора для понятия “скобки”.

ПРИЛОЖЕНИЕ А

КОД ПРОГРАММЫ

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#define RED "\033[0;31m"
#define NONE "\033[0m"

//Функция проверки равенства закрывающихся и открывающихся скобок
int Correct_Place_Bracket(int len_of_text, char *text)
{
    int bracket_count=0; //Счётчик кол-ва скобок
    char *element=text; //Указатель на второй эл-нт текста
    while(*element!=0)
    {
        if(*element=='(')
            bracket_count++; //Увеличиваем счётчик, если скобка открыта
        if(*element==')')
            bracket_count--; //Уменьшаем счётчик, если скобка закрыта
        element++;
    }
    if(bracket_count==0) //Если счётчик равен 0, то наблюдается равенство открытых и закрытых скобок
        return 1;
    else
        return 0;
}

//Функция нахождения закрывающихся скобок для открытых
int Find_Close_Bracket(int len_of_text, char *text)
{
    if(len_of_text==1) return 0;
    int bracket_count=1;
    char *element=text+1;
    while(bracket_count!=0) //Цикл, пока для первой открытой скобки не найдём закрытую
    {
        if(*element=='(')
            bracket_count++;
    }
```



```

if(*element=='')
bracket_count--;
if(*element==0)
return 0;
if(element==text+len_of_text)
break;
element++;
}
return element-text; //Возвращаем кол-во элементов в найденных закрытых скобках
}

//Рекурсивная функция проверки текста на понятие "скобки"
int rec_analis(int len_of_text,char *text)
{
if(len_of_text<1)
return 0;
if(*text=='A' && len_of_text==1)
return 1;
if(*text=='(' && *(text+1)=='B') //Если первый эл-нт ( и последующий B
{
if(Find_Close_Bracket(len_of_text,text)==0) return 0;
//Находим "правильный" участок для запуска анализа
int corr_section=rec_analis(Find_Close_Bracket(len_of_text,text)-3,text+2);
//Используем как переход на новый участок
Int trans_section=rec_analis(len_of_textFind_Close_Bracket(len_of_text,text),text+
Find_Close_Bracket(len_of_text,text));
return corr_section * trans_section; //В случае верной постановки скобок, функция вернёт значение 1
}
else
{
return 0;
}
}

int main()
{
char *text=(char*)malloc(500);
printf("Введите текст: ");

```

```

scanf("%s",text);
int len_of_text=strlen(text); //Длина текста
if(len_of_text==0)
{
printf("%sПрограмма выполняет анализ%s\nТекст пуст!\n", RED, NONE);
return 0;
}
//Если функция не прошла проверку на равенство закрытых и раскрытых скобок
if(Correct_Place_Bracket(len_of_text, text)==0)
{
printf("%sПрограмма выполняет анализ...%s\nНекорректный ввод данных!\n", RED, NONE);
return 0;
}
if(*text!='A' && (*text!='(' || Find_Close_Bracket(len_of_text, text)==0))
{
printf("%sПрограмма выполняет анализ...%s\nВведённый текст не является скобками!\n", RED, NONE);
return 0;
}
if(rec_analis(len_of_text,text)==1) //Производим анализ введённого текста
{
printf("%sПрограмма выполняет анализ...%s\nВведённый текст скобки!\n", RED, NONE);
}
else
{
printf("%sПрограмма выполняет анализ...%s\nВведённый текст не является скобками!\n", RED, NONE);
}
free(text);
return 0;
}

```