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
Вариант 15

Студент гр. 8383 _____ Шишкин И.В.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2019

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных алгоритмов и функций на языке программирования С.

Постановка задачи.

15. Построить синтаксический анализатор для понятия *скобки*.
скобки::=A | A (*ряд_скобок*)
ряд_скобок::= *скобки* | *скобки* ; *ряд_скобок*

Описание алгоритма.

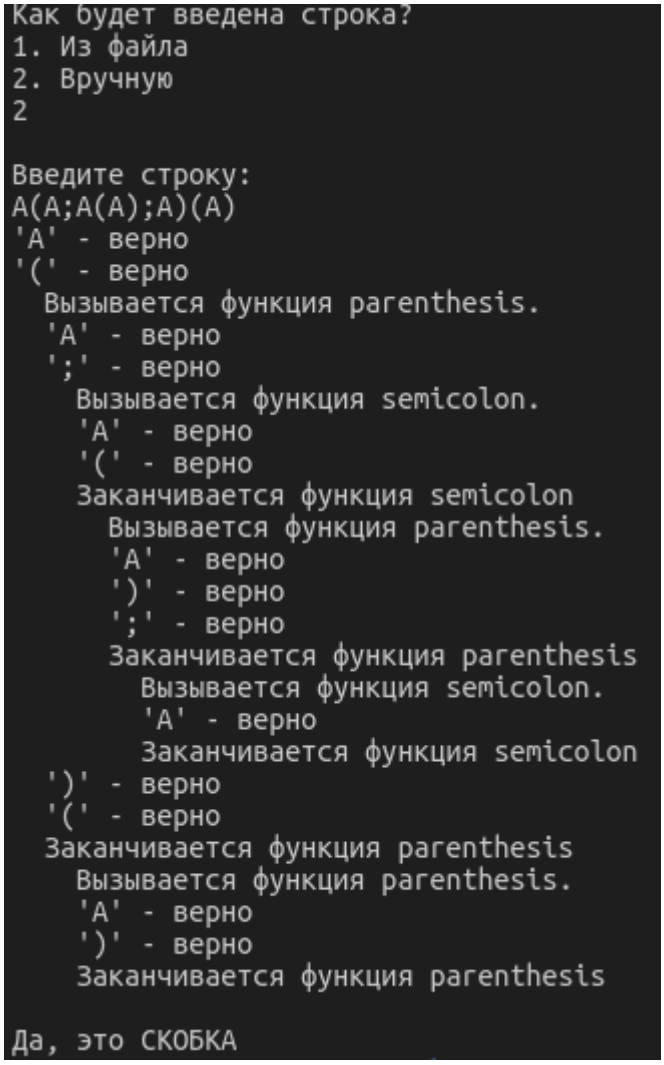
Чтобы определить, является ли введенная строка скобкой, было написано 2 функции. Во-первых, строка всегда должна начинаться с 'A', после которой обязательно должны идти символы '(' или ';'. Если был введен символ '(', то программа вызывает рекурсивную функцию `int parethesis(char symbol, int check, file f)`. Если же был введен символ ';', то программа вызывает рекурсивную функцию `semicolon(char symbol, int check, file f)` (В обеих функциях на вход подаются следующие переменные: symbol - считанный символ, check - переменная, хранящая в себе уровень рекурсии, создана для понятного вывода данных на экран, и f - файл, из которого считываются данные). Далее, после этого обязательно должна быть введена 'A', что и проверяют обе функции. Уже после введенного 'A' могут идти '(', ')', ';', в зависимости от которых будут вызваны соответствующие функции. Обе функции возвращают 0, если введенный символ не подходит под условие, 1 — если верно, и 2, если встречается ')'. Значение 2 было создано для регулировки количества открывающих и закрывающих скобок.

Спецификация программы.

Программа является синтаксическим анализатором для понятия скобки. Программа написана на языке С. Входными данными является строка -

считывается из файла или терминала. Итог работы программы выводится на экран.

Тестирование.



Input	Output
B(A);A	Это не скобка
AA;A(A)	Это не скобка
A;A(A(A;A;A(A;A));A;A(A;A))	Это скобка
A	Это скобка
A(A;A)(A;A);A	Это скобка
A(A;A;)A;A	Это не скобка
A;A;A(A)(A)(A)	Это скобка

Вывод.

В ходе выполнения лабораторной работы был изучен метод рекурсии. Этот метод является удобным для выполнения задачи, так как упрощает выражение алгоритма.

ПРИЛОЖЕНИЕ.

Код программы.

```
#include <stdio.h>
#include <string.h>
#include <dirent.h>
#include <stdlib.h>

int semicolon(char *symbol, int check, FILE *f); //если встречается ;

void printDepth(int indent){       //функция для печати отступа
    for (int i = 0; i < indent * 2; i++)
        printf(" ");
}

char reading(FILE *f){             //функция для считывания символа
    if(f == NULL){
        return getchar();     //если считывание происходит из терминала
    }
    else{
        return fgetc(f);      //если считывание происходит из файла
    }
}

int parenthesis(char *symbol, int check, FILE *f){  //если встречается (

    printDepth(check);
    printf("Вызывается функция parenthesis.\n");
    printDepth(check);
    if( (*symbol = reading(f)) != 'A'){
        printf("'%c' - неверно! Ожидалось 'A'\n", *symbol);
        printDepth(check);
        printf("Заканчивается функция parenthesis\n");
        return 0;
    }

    printf("'%c' - верно\n", *symbol);
    *symbol = reading(f);

    if(*symbol == ')'){
        //check--;
        printDepth(check);
        printf("'%c' - верно\n", *symbol);
        if((*symbol = reading(f)) == '\n'){
            printDepth(check);
            printf("Заканчивается функция parenthesis\n");
            return 1;
        }
```

```
    else if(*symbol == ')'){
        printDepth(check);
        printf("Заканчивается функция parenthesis\n");
        return 2;
    }
    else if(*symbol == ';'){
        printDepth(check);
        printf("'%c' - верно\n", *symbol);
        printDepth(check);
        printf("Заканчивается функция parenthesis\n");
        return semicolon(symbol, check+1, f);
    }
    else if(*symbol == '('){
        printDepth(check);
        printf("'%c' - верно\n", *symbol);
        printDepth(check);
        printf("Заканчивается функция parenthesis\n");
        return parenthesis(symbol, check+1, f);
    }
}

else if (*symbol == ';'){
    printDepth(check);
    printf("'%c' - верно\n", *symbol);
    if (semicolon(symbol, check+1, f) == 2){
        if (*symbol == ')'){
            printDepth(check);
            printf("'%c' - верно\n", *symbol);
            if ((*symbol = reading(f)) == '\n'){
                printDepth(check);
                printf("Заканчивается функция parenthesis\n");
                return 1;
            }
            else if (*symbol == ')'){
                printDepth(check);
                printf("Заканчивается функция parenthesis\n");
                return 2;
            }
            else if (*symbol == ';'){
                printDepth(check);
                printf("'%c' - верно\n", *symbol);
                printDepth(check);
                printf("Заканчивается функция parenthesis\n");
                return semicolon(symbol, check+1, f);
            }
            else if (*symbol == '('){
                printDepth(check);
                printf("'%c' - верно\n", *symbol);
                printDepth(check);
                printf("Заканчивается функция parenthesis\n");
```

```
                    return parenthesis(symbol, check+1, f);
                }
            }
            printDepth(check);
            printf("Заканчивается функция parenthesis\n");
            return 0;
        }
    }

    else if(*symbol == '('){
        printDepth(check);
        printf("'%c' - верно\n", *symbol);
        if(parenthesis(symbol, check+1, f) == 2){
            if(*symbol == ')'){
                printDepth(check);
                printf("'%c' - верно\n", *symbol);
                if((*symbol = reading(f)) == '\n'){
                    printDepth(check);
                    printf("Заканчивается функция parenthesis\n");
                    return 1;
                }
                else if (*symbol == ')'){
                    printDepth(check);
                    printf("Заканчивается функция parenthesis\n");
                    return 2;
                }

            }

        }
    }

    return 0;
}

int semicolon(char *symbol, int check, FILE *f){

    printDepth(check);
    printf("Вызывается функция semicolon.\n");
    printDepth(check);
    if( (*symbol = reading(f)) != 'A'){
        printf("'%c' - неверно! Ожидалось 'A'\n", *symbol);
        printDepth(check);
        printf("Заканчивается функция semicolon\n");
        return 0;
    }

    printf("'%c' - верно\n", *symbol);
    *symbol = reading(f);
```

```
    if(*symbol == ')'){
        printDepth(check);
        printf("Заканчивается функция semicolon\n");
        return 2;
    }
    else if (*symbol == ';'){
        printDepth(check);
        printf("'%c' - верно\n", *symbol);
        printDepth(check);
        printf("Заканчивается функция semicolon\n");
        return semicolon(symbol, check+1, f);
    }
    else if (*symbol =='('){
        printDepth(check);
        printf("'%c' - верно\n", *symbol);
        printDepth(check);
        printf("Заканчивается функция semicolon\n");
        return parenthesis(symbol, check+1, f);
    }
    else if( *symbol == '\n'){
        printDepth(check);
        printf("Заканчивается функция semicolon\n");
        return 1;
    }
    printDepth(check);
    printf("Заканчивается функция semicolon\n");
    return 0;

}


int main(){

    char symbol;
    char *file = NULL;
    char **files = NULL;
    FILE *f;
    int check = 0;
    int textOrType = 0;
    int i = 1;

    printf("Как будет введена строка?\n1. Из файла\n2. Вручную\n");
    scanf("%d", &textOrType);
    printf("\n");
    if (textOrType != 1 && textOrType != 2){
        printf("Нужно ввести 1 или 2!\n");
        return 0;
    }
    else if (textOrType == 2)
        printf("Введите строку:\n");
```

```
    else if (textOrType == 1){
        printf("Выберите файл:\n");
        DIR *dir = opendir("directory");
        struct dirent *ent;
        if (dir){                                                    //Выводит
доступные файлы
            for (i = 1; (ent = readdir (dir)) != NULL; ){
                if (!strchr(ent->d_name, '.')){
                    files = realloc(files, i * sizeof(char*) + 1);
                    files[i-1]     =     malloc(strlen(ent->d_name)     *
sizeof(char) + 1);
                    strcpy(files[i-1], ent->d_name);
                    printf("%d. %s\n", i, files[i-1]);
                    i++;
                }
            }
        }
         textOrType = -1;
        scanf("%d", &textOrType);
        if (textOrType > i-1 || textOrType < 1){
            printf("Неправильно выбран файл!\n");
            return 0;
        }

        file = realloc(file, (strlen(files[textOrType-1]) + 4 + 10) *
sizeof(char) + 1);
        strcpy(file, "directory/");
        strcat(file, files[textOrType-1]);
        file[strlen(file)] = '\0';
        printf("Вы выбрали файл %s\n", files[textOrType-1]);

        for (int j = 0; j < i; j++){
            files[j] = NULL;
            free(files[j]);
        }
        files = NULL;
        free(files);
    }

    f = fopen(file, "r");
    getchar();
    symbol = reading(f);

    if( symbol !=   'A'){                        //проверка 1-го считанного
символа
        printf("Нет, это не СКОБКА (ожидалось, что строка начнется с
'A')\n");
        return 0;
    }
```

```
    printf("'%c' - верно\n", symbol);
    symbol = reading(f);

    if( symbol == '('){
        printf("'%c' - верно\n", symbol);
        if(parenthesis(&symbol, check+1, f) == 1){
            printf("\nДа, это СКОБКА\n");
        }
        else{
            printf("\nНет, это не СКОБКА\n");
        }
    }
    else if( symbol == ';'){
        printf("'%c' - верно\n", symbol);
        if(semicolon(&symbol, check+1, f) == 1){
            printf("\nДа, это СКОБКА\n");
        }
        else{
            printf("\nНет, это не СКОБКА\n");
        }

    }
    else if( symbol == '\n'){
        printf("\nДа, это СКОБКА\n");
    }
    else{
        printf("\nНет, это не СКОБКА (ожидалось, что вторым введеным
символом будет ';' или '(')\n");
    }

    file = NULL;
    free(file);
    return 0;
}
```