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

Студент гр. 8304 _____ Карабанов Р.Е

Преподаватель _____ Фирсов М.А

Санкт-Петербург

2019

Цель работы.

Ознакомиться с приёмами рекурсивного программирования, получить навыки программирования рекурсивных функций.

Задание.

Вариант № 24.

Построить синтаксический анализатор для понятия текст_со_скобками.
текст_со_скобками::=элемент или элемент текст_со_скобками
элемент::= A или B или (текст_со_скобками) или [текст_со_скобками] или
{ текст_со_скобками }

Описание алгоритма.

После вызова главной функции isText, принимающей строку для проверки функция проверяет строку на входящие в неё символы. При встрече элемента A или B функция пропустит его и перейдёт к проверке следующего, т.к строка с данными символами в любом количестве и месте по условию является текстом со скобками. При встрече открывающей скобки вызывается рекурсивная функция Brackets, которая при встрече элемента A или B также пропускает его, устанавливая в значении логической переменной AlphainScope истину. Если рекурсивная функция встречает элемент скобку, то она вызывает себя же и возвращает значение в переменную i (index). При удачном завершении возвращает значение не равное -1, в этом случае устанавливает значение AlphainScope в истину, иначе возвращает -1. Brackets вернёт значение не равное -1 если последняя открывающая скобка является парной к закрывающей - рассматриваемой. Проверяет парность скобок функция PairScope. При выполнении условий — PairScope вернула истину и AlphainScope является истиной функция Brackets возвращает индекс следующего элемента за закрывающей скобкой. В конце, в случае строки, не

являющейся текстом со скобками выводится символ и позиция символа в строке являющегося неверным.

Тестирование.

```
Enter text with scope: ABA[A(B)]
RecFun('[')
    RecFun('(')
    Найдена закрывающая скобка ')', удовлетворяющая условию, выход из рекурсивной функции
Найдена закрывающая скобка ']', удовлетворяющая условию, выход из рекурсивной функции
correct
```

Выводы.

В ходе лабораторной работы было изучены приёмамы рекурсивного программирования, получены навыки программирования рекурсивных функций.

Исходный код.

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

int Brackets(char* str, char sc, int i, int r);
bool isText(char* str);

bool PairScope(char* str, char sc, int i){ // функция проверки парности скобок,
        if (sc == '(' && str[i] == ')') {          // принимает указатель на строку, индекс закрывающей скобки и открывающую скобку
                return true;
        }
        else if (sc == '{' && str[i] == '}') {
                return true;
        }
        else if (sc == '[' && str[i] == ']') {
                return true;
        }
        else {
                return false;
        }
```

```
}

int main(int argc, char* argv[]) {
	char* str = (char*)calloc(200, sizeof(char));
	if (argc == 1){ // если не указана папка с тестами, то пользователь вводи строку
		printf("Enter text with scope: ");
		fgets(str, 199, stdin);
			if (isText(str))
				printf("correct\n");
			else
				printf("wrong\n");
	}
	else {
		char* path = (char*)calloc(strlen(argv[1]) + 8, sizeof(char)); // строка для
открытия файла
		char* estr; // указатель для отслеживания конца файла
		strcat(path, "Tests/");
		strcat(path, argv[1]);
		FILE* fileTests = fopen(path, "r"); // открываем файл
		if (!fileTests){
			printf("error open file\n");
			return 0;
		}
		else{
			while (1){
				estr = fgets(str, 199, fileTests); // считываем строку из файла
				if (estr == NULL){ // если указатель NULL то проверяем на
ошибку или конец файла
					if (feof(fileTests) != 0){
						printf("end file\n");
						break;
					}
					else {
						printf("error reading file\n");
						break;
					}
				}
				if (strlen(str) == 1){ // проверка на пустую строку
					printf("'Пустая строка'\n");
				}
```

```
                                        else{
                                                printf("%s\n", str);
                                        }
                                        if (isText(str)) // проверка на текст со скобками
                                                printf("correct >> %s", str);
                                        else
                                                printf("wrong >> %s", str);
                                }
                                free(path);
                        }
                        /*if (fclose(fileTests) == EOF)
                                printf("error closing file\n");*/
                }
                free(str);
                return 0;
        }

        bool isText(char* str) { // функция проверки, принимает указатель на строку
                int i = 0; // индекс символа
                while (str[i] != '\n' || str[i] == '\0') {
                        if (str[i] == 'A' || str[i] == 'B') { //пропуск букв тк их может сколько угодно идти подряд
                                i++;
                        }
                        else if (str[i] == '(' || str[i] == '[' || str[i] == '{') {
                                i++;
                                i = Brackets(str, str[i - 1], i, 0); // находим скобку и вызываем рекурсивную функцию, возвращаем индекс с которого продолжим
                                if (i == -1) { // если какая-либо ошибка вернётся -1
                                        return false;
                                }
                        }
                        else {
                    printf("Ошибка на символе номер %d - %c \n", i + 1, str[i]);
                                return false;
                }
                    }
                    return true && i; // возвращаем истину, тк дошли до конца без ошибок и проверяем на не пустоту строки.
        }
```

```
int Brackets(char* str, char sc, int i, int r) { /* рекурсивная функция принимает указатель на строку
		символ который является открывающей скобкой индекс текущего символа, глубину рекурсии */
		bool AlphainScope = false; // наличии буквы в скобках
		printf("%*.sRecFun('%c')\n", 4*r, " ", str[i - 1]); // отладочный вывод
		while (str[i] != '\n'|| str[i] == '\0') {
			if (str[i] == 'A' || str[i] == 'B') {
				AlphainScope = true;
				i++;
			}
			else if (str[i] == '(' || str[i] == '[' || str[i] == '{') { // аналогично работе в функции isText
				i++;
				i = Brackets(str, str[i - 1], i, r + 1); // возвращаем индекс с которого будем продолжать
				if (i != -1) {
					AlphainScope = true;
				}
				else {
					return -1;
				}
			}
			else if (PairScope(str, sc, i) && AlphainScope) { // проверка на парность скобок и вложенность в них символа
				i++;
				printf("%*.s", 4*r, " "); // отладочные выводы
				printf("Найдена закрывающая скобка '%c', удовлетворяющая условию, выход из рекурсивной функции\n", str[i - 1]);
				return i;
			}
			else {
				printf("Ошибка на символе номер %d - %c \n", i + 1, str[i]);
				break;
			}
		}
		return -1;
	}
```