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. 9304 _____ Ковалёв П. Д.

Преподаватель _____ Фиалковский М. С.

Санкт-Петербург
2020

Цель работы.

Изучить рекурсию, написать синтаксический анализатор.

Задание.

Вариант 21

Построить синтаксический анализатор для понятия скобки.

скобки::=квадратные | круглые

квадратные:: = [[квадратные](круглые)] | B

круглые::=((круглые)[квадратные]) | A

Выполнение работы.

На вход программе подается строка, состоящая из скобок, пробелов и букв *A* и *B*. Программа должна обработать строку и сообщить, удовлетворяет ли данная строка условию или нет. Для реализации программы были написаны функции *isBracket()*, *Round()* и *Square()*. Строка считывается из массива *argv*, а после она инициализирует строку типа *string*, с которой в дальнейшем будет работать программа.

В функции *isBracket()* сначала определялся тип первой скобки, а далее, в зависимости от него вызывалась функция *Round()*, если последовательность начиналась с круглой открывающейся скобки, или *Square()*, если последовательность начиналась с квадратной открывающейся скобки. Функция возвращает то значение, которое вернет вызванная функция, но если первый символ строки некорректен, то функция возвращает *false*.

Функция *Square()* определяет, удовлетворяет ли подпоследовательность символов понятию квадратной скобки. В данной функции осуществляется проход по строке, и анализ символов. В случае, если функция успешно доходит до символа *B*, функция идет дальше и проверяет, находится ли после него закрывающаяся квадратная скобка и вызывает функцию *Round()*; если же функция находит больше двух идущих через пробел открывающихся

квадратных скобок, то функция вызывает себя еще раз, так как имеется вложенная последовательность скобок. В случае, если встречается некорректный символ, или нарушается условие, что две открывающиеся скобки должны идти подряд через пробел, то функция пишет сообщение об ошибке, выводит часть строки, которую обработала, и возвращает *false*. Если же последовательность корректна, то функция вернет *true*.

Функция *Round()* определяет, удовлетворяет ли подпоследовательность символов понятию круглой скобки. В данной функции осуществляется проход по строке, и анализ символов. В случае, если функция успешно доходит до символа *A*, функция идет дальше и проверяет, находится ли после него закрывающаяся круглая скобка и вызывает функцию *Square()*; если же функция находит больше двух идущих через пробел открывающихся круглых скобок, то функция вызывает себя еще раз, так как имеется вложенная последовательность скобок. В случае, если встречается некорректный символ, или нарушается условие, что две открывающиеся скобки должны идти подряд через пробел, то функция пишет сообщение об ошибке, выводит часть строки, которую обработала, и возвращает *false*. Если же последовательность корректна, то функция вернет *true*.

Тестирование.

Запуск программы начинается с запуска команды *make* в терминале, что приведет к созданию исполняемого файла *lab1*. Запуск программы начинается с ввода команды *./lab1* в терминале в директории *lab1*. Тестирование же проводится с помощью скрипта *tester.py*, который запускается командой *python3 tester.py* в командной строке в директории *lab1*. В файле *test1.txt* лежат входные данные, которые обязательно должны удовлетворять условию задачи, а в файле *test2.txt* лежат входные данные, которые обязательно должны не удовлетворять условию задачи. Таким

образом, происходит проверка того, сможет ли программа корректно обработать правильные и неправильные входные данные.

Результаты тестирования представлены в приложении Б.

Выводы.

Ознакомились с основами рекурсивного программирования на языке программирования C++, освоили способы написания рекурсивных процедур и функций.

Реализовали синтаксический анализатор, который определяет, является ли входная последовательность скобками или нет.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <cstdlib>
#include <string>

bool Round(std::string & s, int& iter, int size, std::string &
err);

bool Square(std::string & s, int& iter, int size, std::string &
err);

bool isBracket(std::string & s);

int main(int argc, char** argv){
    std::string st(argv[1]);
    if(isBracket(st)){
     std::cout << st << '\n';
        std::cout << "It is bracket!" << '\n';
    }
    return 0;
}

bool isBracket(std::string & s){
    std::string err = "";
    int len = s.length();
    int iter = 0;
    err = err + s[iter];
    if(s[0] == '[') {
        iter = iter + 1;
        err = err + s[iter];
        return Square(s, iter, len, err);
    }else if(s[0] == '('){
        iter = iter + 1;
        err = err + s[iter];
        return Round(s, iter, len, err);
    }else{
        std::cout << err << '\n';
        std::cout << "Invalid symbol" << '\n';
        std::cout << "It is not bracket!" << '\n';
        return false;
    }
}

bool Square(std::string & s, int& iter, int size, std::string &
err){
        bool res;
        if(iter != size) {
            if (s[iter] == ' ') {
                iter = iter + 1;
                err = err + s[iter];
                if (s[iter] == '[') {
```

```
                    iter = iter + 1;
                    err = err + s[iter];
                    if (s[iter] == ' ') {
                        iter = iter + 1;
                        err = err + s[iter];
                        if (s[iter] == 'B') {
                            res = true;
                        } else if (s[iter] == '[') {
                            iter = iter + 1;
                            err = err + s[iter];
                            res = Square(s, iter, size, err);
                        } else {
                            std::cout << err << '\n';
                            std::cout << "Invalid symbol" << '\n';
                              std::cout << "It is not bracket!" << '\
n';
                            return false;
                        }
                        iter = iter + 1;
                        err = err + s[iter];
                        if(s[iter] == ' ') {
                            iter = iter + 1;
                            err = err + s[iter];
                            if (s[iter] == ']') {
                                iter = iter + 1;
                                err = err + s[iter];
                                if(s[iter] == ' ') {
                                    iter = iter + 1;
                                    if (s[iter] == '(') {
                                        iter = iter - 1;
                                           res = Round(s, iter, size,
err);
                                    } else if (s[iter] == ')') {
                                        return true;
                                    } else {
                                        std::cout << err << '\n';
                                        std::cout << "Invalid symbol"
<< '\n';
                                             std::cout << "It is not
bracket!" << '\n';
                                        return false;
                                    }
                                }else{
                                    std::cout << err << '\n';
                                        std::cout << "There must be a
probel!" << '\n';
                                    std::cout << "It is not bracket!"
<< '\n';
                                    return false;
                                }
                            } else {
                                std::cout << err << '\n';
                                  std::cout << "Invalid symbol" << '\
n';
                                 std::cout << "It is not bracket!" <<
'\n';
```

```
                                return false;
                            }
                        }else{
                            std::cout << err << '\n';
                            std::cout << "There must be a probel!" <<
'\n';
                              std::cout << "It is not bracket!" << '\
n';
                            return false;
                        }
                    }else{
                        std::cout << err << '\n';
                         std::cout << "There must be a probel!" << '\
n';
                        std::cout << "It is not bracket!" << '\n';
                        return false;
                    }
                }else{
                    std::cout << err << '\n';
                    std::cout << "Invalid symbol" << '\n';
                    std::cout << "It is not bracket!" << '\n';
                    return false;
                }
            }else{
                std::cout << err << '\n';
                std::cout << "There must be a probel!" << '\n';
                std::cout << "It is not bracket!" << '\n';
                return false;
            }
        }
        return res;
    }

    bool Round(std::string & s, int& iter, int size, std::string &
err){
        bool res;
        if(iter != size) {
            if (s[iter] == ' ') {
                iter = iter + 1;
                err = err + s[iter];
                if (s[iter] == '(') {
                    iter = iter + 1;
                    err = err + s[iter];
                    if (s[iter] == ' ') {
                        iter = iter + 1;
                        err = err + s[iter];
                        if (s[iter] == 'A') {
                            res = true;
                        } else if (s[iter] == '(') {
                            iter = iter + 1;
                            err = err + s[iter];
                            res = Round(s, iter, size, err);
                        } else {
                            std::cout << err << '\n';
                            std::cout << "Invalid symbol" << '\n';
```

```
                                std::cout << "It is not bracket!" << '\
n';
                            return false;
                        }
                        iter = iter + 1;
                        err = err + s[iter];
                        if(s[iter] == ' ') {
                            iter = iter + 1;
                            err = err + s[iter];
                            if (s[iter] == ')') {
                                iter = iter + 1;
                                err = err + s[iter];
                                if(s[iter] == ' ') {
                                    iter = iter + 1;
                                    if (s[iter] == '[') {
                                        iter = iter - 1;
                                          res = Square(s, iter, size,
err);
                                    } else if (s[iter] == ']') {
                                        return true;
                                    } else {
                                        std::cout << err << '\n';
                                        std::cout << "Invalid symbol"
<< '\n';
                                              std::cout << "It is not
bracket!" << '\n';
                                        return false;
                                    }
                                }else{
                                    std::cout << err << '\n';
                                        std::cout << "There must be a
probel!" << '\n';
                                    std::cout << "It is not bracket!"
<< '\n';
                                    return false;
                                }
                            } else {
                                std::cout << err << '\n';
                                  std::cout << "Invalid symbol" << '\
n';
                                   std::cout << "It is not bracket!" <<
'\n';
                                return false;
                            }
                        }else{
                            std::cout << err << '\n';
                            std::cout << "There must be a probel!" <<
'\n';
                              std::cout << "It is not bracket!" << '\
n';
                            return false;
                        }
                    }else{
                        std::cout << err << '\n';
                         std::cout << "There must be a probel!" << '\
n';
```

```
                    std::cout << "It is not bracket!" << '\n';
                    return false;
                }
            }else{
                std::cout << err << '\n';
                std::cout << "Invalid symbol" << '\n';
                std::cout << "It is not bracket!" << '\n';
                return false;
            }
        }else{
            std::cout << err << '\n';
            std::cout << "There must be a probel!" << '\n';
            std::cout << "It is not bracket!" << '\n';
            return false;
        }
    }
    return res;
}
```

ПРИЛОЖЕНИЕ Б
ТЕСТИРОВАНИЕ

Таблица Б.1 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	[[B] (A)]	[[B] (A)] It is bracket!	Последовательность правильная
2.	((A) [B])	((A) [B]) It is bracket!	Последовательность правильная
3.	[[[[B] (A)]] (A)]	[[[[B] (A)]] (A)] It is bracket!	Последовательность правильная
4.	((((A) [B])) [B])	((((A) [B])) [B]) It is bracket!	Последовательность правильная
5.	[[[[[[B] (A)]] (A)]] (A)]	[[[[[[B] (A)]] (A)]] (A)] It is bracket!	Последовательность правильная
6.	shdiusgif	s Invalid symbol It is not bracket!	Последовательность неправильная
7.	[[B] (B)]	[[B] (B Invalid symbol It is not bracket!	Последовательность неправильная
8.	((B) [A])	((B Invalid symbol It is not bracket!	Последовательность неправильная
9.		 Invalid symbol It is not bracket!	Последовательность неправильная
10.	[B] (A)	[B Invalid symbol It is not bracket!	Последовательность неправильная