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 Сорин А.В.

Преподаватель Филатов А.Ю.

Санкт-Петербург

2020

Цель работы.

Изучить основы работы с рекурсией на примере языка c++.

Задание.

Задание 16. Построить синтаксический анализатор для понятия скобки.

скобки::=A | B | (скобки скобки)

Формат входных и выходных данных.

На вход подается выражение. Например:

(A ((B A) B))

На выходе результат – подходит результат под определение скобок или нет.

Выполнение работы.

Для выполнения задания была создана функция IsBrackets, которая работает следующим образом: если встретился символ ‘A’ или символ ‘B’, функция возвращает true. Если встретилась ‘(‘, то функция вызывает себя рекурсивно, затем проверяет наличие пробела, после чего снова вызывает себя рекурсивно. После этого функция проверяет наличие ‘)’ и возвращает результат логического и от результатов вызова рекурсии. Во всех остальных случаях вызываются обработанные исключения. Функция main получает из потока ввода строку, после чего передает ее в функцию IsBrackets в виде потока.

Тестирование.

Тестирование проводится при помощи скрипта на python. При этом текст из входного файла выводится на экран и после его ввода, передается в поток ввода программы, а получившийся на выходе результат сравнивается в правильным и выводится, пройден тест, или нет. В таблице приведены результаты тестирования.

Таблица Б.1 – Результаты тестирования

№	Входные данные	Выходные данные	Комментарии
1	A	These are brackets	Скобки
2	(A B)	These are brackets	Скобки, выражение посложнее
3	((A B) ((B A) A))	These are brackets	Еще более сложное выражение со вложенностями
4	((	Error while entering expression	После скобки нет ничего, ошибка
5	(AB	Missed space	Отсутствует пробел, ошибка
6	(5B A)	Missed 'A', 'B' or '('	После скобки стоит неправильный символ, ошибка
7	((A B) (B B A)	Missed ')'	Отсутствует закрывающая скобка в нужном месте, ошибка
8	((((A (B A)) B) (A A)) (A (B B)))	These are brackets	Скобки. Очень сложное выражение.

Выводы.

В ходе выполнения работы были получены навыки использования рекурсии для решения поставленных задач. Была разработана программа на языке c++, которая анализирует скобочное выражение.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <string>
#include <iostream>
#include <stdexcept>
#include <sstream>

bool IsBrackets(std::stringstream& Stream) {
char c = 0;

if (!Stream.get(c))
     throw std::invalid_argument("Error while entering expression\n");
if (c == 'A') {
     return true;
}
else if (c == 'B') {
     return true;
}
else if (c == '(') {
     bool Res1, Res2;
     Res1 = IsBrackets(Stream);
     if (!Stream.get(c))
          throw std::invalid_argument("Error while entering
expression\n");
     if (c != ' ')
          throw std::invalid_argument("Missed space\n");
     Res2 = IsBrackets(Stream);
     if (!Stream.get(c))
          throw std::invalid_argument("Error while entering
expression\n");
     if (c != ')')
          throw std::invalid_argument("Missed \')\'\n");
```

```
	return Res1 && Res2;
}
else
	throw std::invalid_argument("Missed \'A\', \'B\' or \'(\'\n");
}

int main() {
try
{
	bool Res = 0;
	std::string Str;
	if (!std::getline(std::cin, Str))
		throw std::runtime_error("Error while reading from stream\n");
	std::stringstream Stream(Str);
	Res = IsBrackets(Stream);
	if (Res == true)
		std::cout << "These are brackets" << std::endl;
	else
		std::cout << "These are not brackets" << std::endl;
}
catch (const std::exception& Error)
{
	std::cout << Error.what();
}
return 0;
```

} **Название файла тестирующей программы: lab3_test.py**

```
import os

number_of_tests = 5
exec_file = './lab3'
path_to_tests = 'Tests/tests/test_'
path_to_answers = 'Tests/answers/answers_'
path_to_correct_answers = 'Tests/correct_answers/answers_'
```

```
exp = '.txt'


for i in range(number_of_tests):
    num = i + 1
    os.system(
        f'{exec_file} {path_to_tests}{num}{exp} >
{path_to_answers}{num}{exp}')


    ans_1 =
open(f'{path_to_correct_answers}{num}{exp}').readline().rstrip('\n')
    ans_2 = open(f'{path_to_answers}{num}{exp}').readline().rstrip('\n')
    str_test = open(f'{path_to_tests}{num}{exp}').readline()


    print(f'test_{num}:\n  string: {str_test}\n  answer: {ans_1}\n
result: ', end='')
    if ans_1 == ans_2:
        print(f'correct')
    else:
        print(f'incorrect')
```

Название файла: Makefile

```
lab1: ./src/main.cpp
     g++ -std=c++17 ./src/main.cpp -o lab1
```

Название файла: tests.py

```
import unittest
import subprocess



class tester(unittest.TestCase):


     def test1(self):
          with open('./Tests/test_1.txt', 'r') as file:
```

```
                print('\nenter', file.readline())
                res = "These are brackets\n"

                self.assertEqual(subprocess.check_output(["./lab1"],
universal_newlines=True), res)
                print(res)

    def test2(self):
        with open('./Tests/test_2.txt', 'r') as file:
                print('\nenter', file.readline())
                res = "These are brackets\n"

                self.assertEqual(subprocess.check_output(["./lab1"],
universal_newlines=True), res)
                print(res)

    def test3(self):
        with open('./Tests/test_3.txt', 'r') as file:
                print('\nenter', file.readline())
                res = "These are brackets\n"

                self.assertEqual(subprocess.check_output(["./lab1"],
universal_newlines=True), res)
                print(res)

    def test4(self):
        with open('./Tests/test_4.txt', 'r') as file:
                print('\nenter', file.readline())
                res = "Error while entering expression\n"

                self.assertEqual(subprocess.check_output(["./lab1"],
universal_newlines=True), res)
```

```
                print(res)


        def test5(self):
                with open('./Tests/test_5.txt', 'r') as file:
                        print('\nenter', file.readline())
                        res = "Missed space\n"


                        self.assertEqual(subprocess.check_output(["./lab1"],
universal_newlines=True), res)
                        print(res)


        def test6(self):
                with open('./Tests/test_6.txt', 'r') as file:
                        print('\nenter', file.readline())
                        res = "Missed \'A\', \'B\' or \'(\'\n"


                        self.assertEqual(subprocess.check_output(["./lab1"],
universal_newlines=True), res)
                        print(res)


        def test7(self):
                with open('./Tests/test_7.txt', 'r') as file:
                        print('\nenter', file.readline())
                        res = "Missed \')\'\n"


                        self.assertEqual(subprocess.check_output(["./lab1"],
universal_newlines=True), res)
                        print(res)


        def test8(self):
                with open('./Tests/test_8.txt', 'r') as file:
```

```
                print('\nenter', file.readline())
                res = "These are brackets\n"

                self.assertEqual(subprocess.check_output(["./lab1"],
universal_newlines=True), res)
                print(res)




if __name__ == '__main__':
     unittest.main()
```