**МИНОБРНАУКИ РОССИИ**

**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**

**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**

**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**

**Кафедра МОЭВМ**

**ОТЧЕТ**

**По лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема**: **Рекурсия**

|  |  |  |
| --- | --- | --- |
| Студент гр. 9381 |  | Судаков Е.В. |
| Преподаватель |  | Фирсов М.А. |

Санкт-Петербург

2020

**1. Цель работы**.

Ознакомиться с рекурсивными алгоритмами и реализовать рекурсивную функцию соответствующую условию.

**2. Задание.**

16. Построить синтаксический анализатор для понятия скобки.

скобки::=А | B | ( скобки скобки )

# 3. Пример работы программы:

|  |  |
| --- | --- |
| input | output |
| (AB) | Parsed A  Parsed B  Parsed (AB)  (AB) is skobki! |
| ((AB)(AB)) | Parsed A  Parsed B  Parsed (AB)  Parsed A  Parsed B  Parsed (AB)  Parsed ((AB)(AB))  ((AB)(AB)) is skobki! |
| ((AB)B) | Parsed A  Parsed B  Parsed (AB)  Parsed B  Parsed ((AB)B)  ((AB)B) is skobki! |
| ((A(AB))B) | Parsed A  Parsed A  Parsed B  Parsed (AB)  Parsed (A(AB))  Parsed B  Parsed ((A(AB))B)  ((A(AB))B) is skobki! |
| (A(B)) | Parsed A  Parsed B  Нет второй скобки в () на позиции 4  (A(B))not a skobki |
| (((AB)(AB))((AB)(AB))) | Parsed A  Parsed B  Parsed (AB)  Parsed A  Parsed B  Parsed (AB)  Parsed ((AB)(AB))  Parsed A  Parsed B  Parsed (AB)  Parsed A  Parsed B  Parsed (AB)  Parsed ((AB)(AB))  Parsed (((AB)(AB))((AB)(AB)))  (((AB)(AB))((AB)(AB))) is skobki! |

# 4. Выполнение работы:

Для ввода информации из файла необходимо ввести “1” в начале работы программы.

Для ввода информации через консоль необходимо ввести “0” в начале работы программы.

Программе подается на вход единственная строка для разбора.

После ввода строки вызывается функция parser(string, int&, int), которая управлет всем разбором : находит скобки, ищет ошибки, запускает вывод “лестницы” разбора.  
После завершения работы функции parser(), функция main выводит результат работы алгоритма

# 5. Описание функций:

printDepth(string s, int k, int depth ) - функция выводы элементов лестницы разбора.

s - исходная строка, k - индекс, указывающий на конец скобки, depth - индекс начала скобки.

parser(string s, int &k, int depth) - функция управления разбором, соответствующая условию

**скобки::=А | B | ( скобки скобки )**

Задача заключается в том, чтобы в каждой паре скобок () находить по две другие скобки, если это не базовый случай, когда скобка = A | B.

getNextSkobki(string s, int &k, int depth) - рекурсивная функция поиска следующей скобки.

error(string s, int &k, int c) - процедура вывода текста ошибки

**6. Идея алгоритма:**

В синтаксическом разборе есть 2 возможных состояния(исключая заведомо ошибочные):

1. Базовый случай. скобки::=А | B |. По достижении этого состояния либо заканчивается разбор(если строка и есть А|B), либо парсеру передается одна из скобок в ().
2. Строка начинается с символа (. Это означает, что парсеру необходимо найти две соседних скобки. Для этого 2 раза вызывается функция getNextSkobki(), которая именно этим и занимается. Аналогично, если и в самой getNextSkobki() встретится символ (, тогда опять таки необходимо найти две скобки.

# 7. Вывод:

В ходе выполнения лабораторной работы был изучен такой вид алгоритмов, как синтаксические анализаторы. Была реализована программа, которая анализирует строку рекурсивным методом, определяя соответствие определению.

# 8. Код программы

#include <bits/stdc++.h>

#define RESET "\033[0m"

#define FAILURE "\033[1m\033[31m"

#define SUCCES "\033[1m\033[32m"

using namespace std;

//скобки::=А | B | ( скобки скобки )

int getNextSkobki(string s, int &k, int depth);

void error(string, int, int);

void printDepth(string s, int k, int depth) {

for (int i = 0; i < depth; i++) cout << " ";

cout << "Parsed ";

for (int i = depth; i <= k; i++) cout << s[i];

cout << "\n";

}

int parser(string s, int &k, int depth) {

if (s[k] == 'A' || s[k] == 'B') {

printDepth(s, k, k);

return 1;

}

// Внутри () ждем 2 скобки

depth = k;

if (s[k] == '(') {

if (getNextSkobki(s, ++k, depth + 1)) {

if (getNextSkobki(s, ++k, depth + 1)) {

if (s[++k] == ')') {

printDepth(s, k, 0);

return 1;

} else

error(s, k, 4);

} else

error(s, k, 3);

} else

error(s, k, 2);

}

return 0;

}

int getNextSkobki(string s, int &k, int depth) {

if (s[k] == 'A' || s[k] == 'B') {

printDepth(s, k, k);

return 1;

}

depth = k;

if (s[k] == '(') {

if (getNextSkobki(s, ++k, depth + 1)) {

if (getNextSkobki(s, ++k, depth + 1)) {

if (s[++k] == ')') {

printDepth(s, k, depth);

return 1;

}

}

}

}

return 0;

}

void error(string s, int k, int c) {

cout << FAILURE;

switch (c) {

case 1:

cout << "Недопустимый символ в скобке: " << s[k] << " на позиции " << k << "\n";

break;

case 2:

cout << "НЕ скобка после ( на позиции " << k << "\n";

break;

case 3:

cout << "Нет второй скобки в () на позиции " << k << "\n";

break;

case 4:

cout << "Нет закрывающей ) в скобке на позиции " << k << "\n";

break;

}

}

void runTests() {

int cnt = 0;

vector<string> tests;

tests.emplace\_back("A");

tests.emplace\_back("(AB)");

tests.emplace\_back("((AB)(AB))");

tests.emplace\_back("((AB)B)");

tests.emplace\_back("((A(AB))B)");

tests.emplace\_back("((A((A(AB))B))B)");

tests.emplace\_back("(((AB)(AB))((AB)(AB)))");

tests.emplace\_back("(B(AA))");

tests.emplace\_back("((AB)(A(((AB)(AB))((AB)(AB)))))");

// НЕ СКОБКИ

tests.emplace\_back("d");

tests.emplace\_back("(A)");

tests.emplace\_back("(A(B))");

for (const auto &t : tests) {

int k = 0, depth = 0;

if (parser(t, k, depth)) {

cnt++;

cout << SUCCES << t << " passed\n";

} else {

cout << FAILURE << t << " not passed\n";

}

cout << RESET;

}

cout << SUCCES << cnt << " tests PASSED" << FAILURE << " " << tests.size() - cnt << " NOT PASSED \n";

}

int main() {

int f;

cout << "if you want to input from file press 1 else press 0\n";

cin >> f;

if (f) {

freopen("input.txt", "r", stdin);

freopen("output.txt", "w", stdout);

}

cout << "Введите строку:\n";

string s;

cin >> s;

int k = 0, depth = 0;

if (parser(s, k, depth)) {

cout << SUCCES << s << " is skobki!\n";

} else cout << FAILURE << s << "not a skobki\n";

// runTests();

return 0;

}