**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 |  | Тиняков С.А. |
| Преподаватель |  | Фиалковский М. С. |

Санкт-Петербург

2020

## Цель работы.

Научиться применять рекурсию в программировании.

## Задание.

Вариант 20.

Построить синтаксический анализатор понятия список\_параметров.

список\_параметров::= параметр | параметр , список\_параметров

параметр::= имя=цифра цифра | имя=(список\_параметров)

имя::= буква буква буква

## Выполнение работы.

Программа считывает данные из входного файла. Алгоритм в своей сути прост: он рекурсивно идёт по понятиям. Всё начинается с понятия список\_параметров. Он, точно, начинается с понятия параметр. Алгоритм переходит к проверки понятия параметр. Он, точно, начинается с понятия имя, и алгоритм переходит к проверке этого понятия. Если после символо «=» в понятии параметр идет символ «(«, то алгоритм идёт проверять понятие список\_параметров. Аналогично, если в понятии список\_параметров после понятия параметр идёт запятая, то вызывается проверка для понятия список\_параметров.

На вход и выход программе подаются файлы через аргументы командной строки. Пробелы, символы табуляции и переноса строки и т. п. игнорируются. Во входном файле должно быть только то, что необходимо проанализировать. Никаких других предложений, символов и прочего быть не должно. В выходном файле выводится проанализированное понятие и то, что оно корректно, если понятие корректно. Иначе выводятся символы до ошибки(включая символ, на котором возникла она) и описание ошибки.

Класс *ReaderWriter* отвечает за ввод и вывод данных. В себе имеет два потока: один для входных данных, второй для выходных. Метод *GetNextChar* возвращает следующий символ из входного потока. Если достигнут конец, то возвращается нулевой символ. Метод *GetAndWriteNextChar* делает тоже самое, только дополнительно записывает символ в выходной поток(нулевой символ не записывается). Метов *IsEOF()* сообщает, достигнут ли конец входного потока. Методы *WriteChar* и *WriteString* записывают в выходной файл соответсвенно символ и строку.

Класс *Analyzer* проверяет корректность понятия список\_параметров. Метод *CheckListParam* проверяет корректность понятия список\_параметров. Метод *ChecParam* проверяет корректность понятия параметр. Метод *ChecName* проверяет корректность понятия имя. Метод *StartAnalyz* запускает проверку входных данных.

Разработанный программный код см. в приложении А.

Результаты тестирования см. в приложении Б.

## Выводы.

Научились применять рекурсию в программировании.

Была разработана программа для проверки понятия список\_параметров. Реализация проверки сделана через рекурсию. Класс *ReaderWriter*, который отвечает за входные и выходные данные, сделан через потоки, что позволяет при помощи класса *Analyzer* проанализировать любой объект, с которым можно взаимодействовать через интрефейс потоков(например строки). Метод *WriteString* принимает аргумент типа *string\_view*, что позволяет передовать методу как класс *string*, так и массив символов.Для проверки правильности работы программы были сделаны тесты.

# Приложение А Исходный код программы

Название файла: Source/lab1.cpp

#include<iostream>

#include<fstream>

#include<string\_view>

/\* For analyz from string \*/

// #include<sstream>

class ReaderWriter{

private:

std::istream& input;

std::ostream& output;

public:

ReaderWriter(std::istream& in, std::ostream& out):

input(in),

output(out)

{

}

~ReaderWriter(){

output << '\n';

}

char GetNextChar(){

char c;

input >> c;

if(IsEOF()) return '\0';

return c;

}

char GetAndWriteNextChar(){

char c;

input >> c;

if(IsEOF()) return '\0';

output << c;

return c;

}

bool IsEOF(){

return input.eof();

}

void WriteChar(char c){

output << c;

}

void WriteString(std::string\_view str){

output << str;

}

};

class Analyzer{

private:

ReaderWriter\* reader\_writer;

char last\_char;

std::string error\_msg;

bool CheckListParam(){

;

if(CheckParam()){

char next = reader\_writer->GetAndWriteNextChar();

if(next == ',') return CheckListParam();

else if(next == ')'){

last\_char = next;

return true;

}

else if(!next) return true;

error\_msg = std::string("List of parameters Error: Expected character \',\', but were given \'") + next + "\'.\n";

return false;

}

return false;

}

bool CheckParam(){

;

if(CheckName()){

char next = reader\_writer->GetAndWriteNextChar();

if(next == '='){

next = reader\_writer->GetAndWriteNextChar();

if(next == '('){

if(CheckListParam()){

if(last\_char == ')'){

last\_char = 0;

return true;

}

next = reader\_writer->GetAndWriteNextChar();

if(next == ')') return true;

if(next) error\_msg = std::string("Parameter Error: Expected character \')\', but were given \'") + next + "\'.\n";

else error\_msg = "Parameter Error: Expected character \')\', but end of input were reached.\n";

return false;

}

return false;

}else{

if(!(next >= '0' && next <='9')){

if(next) error\_msg = std::string("Parameter Error: Invalid define, character \'") + next + "\' is not digit.\n";

else error\_msg = "Parameter Error: Invalid define, expected digit, but end of input were reached.\n";

return false;

}

next = reader\_writer->GetAndWriteNextChar();

if(!(next >= '0' && next <='9')){

if(next) error\_msg = std::string("Parameter Error: Invalid define, character \'") + next + "\' is not digit.\n";

else error\_msg = "Parameter Error: Invalid define, expected digit, but end of input were reached.\n";

return false;

}

return true;

}

}

if(next) error\_msg = std::string("Parameter Error: Expected character \'=\', but were given \'") + next + "\'.\n";

else error\_msg = "Parameter Error: Expected character \'=\', but end of input were reached.\n";

return false;

}

return false;

}

bool CheckName(){

int count = 0;

while(count < 3){

char next = reader\_writer->GetAndWriteNextChar();

if(!(next >= 'A' && next <= 'z')){

if(next) error\_msg = std::string("Name Error: Invalid name, character \'") + next + "\' is not letter.\n";

else error\_msg ="Name Error: Invalid name, excepted letter, but end of input were reached.\n";

return false;

}

count++;

}

return true;

}

public:

Analyzer(ReaderWriter\* reader\_writer){

this->reader\_writer = reader\_writer;

last\_char = 0;

}

~Analyzer(){}

void StartAnalyz(){

bool correct = CheckListParam();

char next = reader\_writer->GetNextChar();

if(last\_char != 0 && error\_msg.empty()){

error\_msg = std::string("List of parameters Error: Expected character \',\', but were given \'") + last\_char + "\'.\n";

correct = false;

}

if(!correct){

reader\_writer->WriteString("\nIncorrect. ");

reader\_writer->WriteString(error\_msg);

}else reader\_writer->WriteString("\nCorrect.\n");

}

};

int main(int argc, char\*\* argv){

if(argc < 3){

std::cout << "Usage: param\_analyzer /path/to/input /path/to/output\n";

return 1;

}

try{

std::ifstream in(argv[1]);

std::ofstream out(argv[2]);

if(!in.is\_open()){

std::cout << "Can't open file " << argv[1] << "\n";

return 2;

}

if(!out.is\_open()){

std::cout << "Can't open/create file " << argv[2] << "\n";

return 2;

}

/\* Example with string \*/

// std::stringstream in("AAA = 15\n");

{

ReaderWriter rw(in, out);

Analyzer analyzer(&rw);

analyzer.StartAnalyz();

}

in.close();

out.close();

}catch(std::exception& e){

std::cout << e.what() << "\n";

return 3;

}

return 0;

}

Название файла: Makefile

lab1: Source/lab1.cpp

g++ Source/lab1.cpp -std=c++17 -o lab1

run\_tests: lab1

python3 test.py

Название файла: test.py

import unittest

import subprocess

import os

class TestParamAnalyzer(unittest.TestCase):

cwd = os.getcwd();

def test\_0\_basic(self):

subtests = [

['AAA = 15', "AAA=15\nCorrect.\n\n"],

['AA4 = 23','AA4\nIncorrect. Name Error: Invalid name, character \'4\' is not letter.\n\n'],

['AAw = 2P','AAw=2P\nIncorrect. Parameter Error: Invalid define, character \'P\' is not digit.\n\n'],

['BBB= \n (RRR =\n56)', "BBB=(RRR=56)\nCorrect.\n\n"],

['AAA = BBB = 45','AAA=B\nIncorrect. Parameter Error: Invalid define, character \'B\' is not digit.\n\n'],

['AAA = (BBB = 13)','AAA=(BBB=13)\nCorrect.\n\n'],

['AAA = (BBB = 13','AAA=(BBB=13\nIncorrect. Parameter Error: Expected character \')\', but end of input were reached.\n\n'],

['AAA = (BBB = 1','AAA=(BBB=1\nIncorrect. Parameter Error: Invalid define, expected digit, but end of input were reached.\n\n'],

['AAA\n= 14)','AAA=14)\nIncorrect. List of parameters Error: Expected character \',\', but were given \')\'.\n\n'],

['AAA\n= 14o','AAA=14o\nIncorrect. List of parameters Error: Expected character \',\', but were given \'o\'.\n\n'],

['AAA\n= 14,','AAA=14,\nIncorrect. Name Error: Invalid name, excepted letter, but end of input were reached.\n\n'],

['AAA 14','AAA1\nIncorrect. Parameter Error: Expected character \'=\', but were given \'1\'.\n\n'],

['AAA ','AAA\nIncorrect. Parameter Error: Expected character \'=\', but end of input were reached.\n\n']

]

for test in subtests:

with open('input', 'w') as f:

f.write(test[0])

p = subprocess.run(['./lab1','input','output'], cwd = self.cwd)

with open('output', 'r') as f:

line = f.read()

self.assertEqual(line, test[1])

def test\_1(self):

subtests = [

['AAA = 15 , BBB = (AaA = (AAA = (AAA = 12)), FFF =10)\n','AAA=15,BBB=(AaA=(AAA=(AAA=12)),FFF=10)\nCorrect.\n\n'],

['AAA = 15 , BBB = (AaA = (AAA = (AAA = 12))), FFF =10\n','AAA=15,BBB=(AaA=(AAA=(AAA=12))),FFF=10\nCorrect.\n\n'],

['AAA = 15 , BBB = (AaA = (AAA = (AAA = 12))), FFF =10)\n','AAA=15,BBB=(AaA=(AAA=(AAA=12))),FFF=10)\nIncorrect. List of parameters Error: Expected character \',\', but were given \')\'.\n\n'],

['AAA = 15 , BBB = (AaA = (AAA = (AAA = 12)), FFF =10\n','AAA=15,BBB=(AaA=(AAA=(AAA=12)),FFF=10\nIncorrect. Parameter Error: Expected character \')\', but end of input were reached.\n\n'],

['AAA = 15 , BBB = AaA = AAA = AAA = 12, FFF =10\n','AAA=15,BBB=A\nIncorrect. Parameter Error: Invalid define, character \'A\' is not digit.\n\n'],

]

for test in subtests:

with open('input', 'w') as f:

f.write(test[0])

p = subprocess.run(['./lab1','input','output'], cwd = self.cwd)

with open('output', 'r') as f:

line = f.read()

self.assertEqual(line, test[1])

def test\_2(self):

subtests = [

['Ssf = (AOE = 12,\n FWA = 14,\n ASF = 16,\n VER = 20\n ),\nFFF = 04, QWE=(AFA = (aas = 13,\n vsd = ( yet = (faw = ( ker = ( ort = 95, not = 00,\n trr = ( zzz = (kkk = (uuu = 14)))))))))','Ssf=(AOE=12,FWA=14,ASF=16,VER=20),FFF=04,QWE=(AFA=(aas=13,vsd=(yet=(faw=(ker=(ort=95,not=00,trr=(zzz=(kkk=(uuu=14)))))))))\nCorrect.\n\n'],

['asd = 00, fbk = 11, AAA = (OOO = 13),\nFFF = 12, BBB = 94, QQQ = 93, KKK = 21,\nMMM = 21, NNN = 21, DDD = (EEE = (kad = 94)),\nDDD = 64, VVV = 82,HHH=65, LLL = 08, ret = 42','asd=00,fbk=11,AAA=(OOO=13),FFF=12,BBB=94,QQQ=93,KKK=21,MMM=21,NNN=21,DDD=(EEE=(kad=94)),DDD=64,VVV=82,HHH=65,LLL=08,ret=42\nCorrect.\n\n']

]

for test in subtests:

with open('input', 'w') as f:

f.write(test[0])

p = subprocess.run(['./lab1','input','output'], cwd = self.cwd)

with open('output', 'r') as f:

line = f.read()

self.assertEqual(line, test[1])

def test\_3(self):

p = subprocess.run(['./lab1','input'], cwd = self.cwd, stdout = subprocess.PIPE)

self.assertEqual(p.returncode, 1);

p = subprocess.run(['./lab1','non\_exist\_file','output'], cwd = self.cwd, stdout = subprocess.PIPE)

self.assertEqual(p.returncode, 2);

def tearDown(self):

if os.path.isfile('./input'):

os.remove('./input')

if os.path.isfile('./output'):

os.remove('./output')

if \_\_name\_\_ == "\_\_main\_\_":

unittest.main()

# Приложение Б Тестирование

Результаты тестирования представлены в таблице Б.1

Таблица Б.1 — Результаты тестирования

|  |  |  |
| --- | --- | --- |
| № п/п | Входные данные | Выходные данные |
|  | AAA = 15 | AAA=15  Correct. |
|  | AA4 = 23 | AA4  Incorrect. Name Error: Invalid name, character '4' is not letter. |
|  | AAw = 2P | Aaw=2P  Incorrect. Parameter Error: Invalid define, character 'P' is not digit. |
|  | BBB=  (RRR =  56) | BBB=(RRR=56)  Correct. |
|  | AAA = BBB = 45 | AAA=B  Incorrect. Parameter Error: Invalid define, character 'B' is not digit. |
|  | AAA = (BBB = 13) | AAA=(BBB=13)  Correct. |
|  | AAA = (BBB = 13 | AAA=(BBB=13  Incorrect. Parameter Error: Expected character ')', but end of input were reached. |
|  | AAA = (BBB = 1 | AAA=(BBB=1  Incorrect. Parameter Error: Expected character ')', but end of input were reached. |
|  | AAA  = 14) | AAA=14)  Incorrect. List of parameters Error: Expected character ',', but were given ')'. |
|  | AAA  = 14o | AAA=14o  Incorrect. List of parameters Error: Expected character ',', but were given 'o'. |
|  | AAA  = 14, | AAA=14,  Incorrect. Name Error: Invalid name, excepted letter, but end of input were reached. |
|  | AAA 14 | AAA1  Incorrect. Parameter Error: Expected character '=', but were given '1'. |
|  | AAA | AAA  Incorrect. Parameter Error: Expected character '=', but end of input were reached. |
|  | AAA = 15 , BBB = (AaA = (AAA = (AAA = 12)), FFF =10 | AAA=15,BBB=(AaA=(AAA=(AAA=12)),FFF=10)  Correct. |
|  | AAA = 15 , BBB = (AaA = (AAA = (AAA = 12))), FFF =10 | AAA=15,BBB=(AaA=(AAA=(AAA=12))),FFF=10  Correct. |
|  | AAA = 15 , BBB = (AaA = (AAA = (AAA = 12))), FFF =10) | AAA=15,BBB=(AaA=(AAA=(AAA=12))),FFF=10)  Incorrect. List of parameters Error: Expected character ',', but were given ')'. |
|  | AAA = 15 , BBB = (AaA = (AAA = (AAA = 12)), FFF =10 | AAA=15,BBB=(AaA=(AAA=(AAA=12)),FFF=10  Incorrect. Parameter Error: Expected character ')', but end of input were reached. |
|  | AAA = 15 , BBB = AaA = AAA = AAA = 12, FFF =10 | AAA=15,BBB=A  Incorrect. Parameter Error: Invalid define, character 'A' is not digit. |
|  | Ssf = (AOE = 12,  FWA = 14,  ASF = 16,  VER = 20  ),  FFF = 04, QWE=(AFA = (aas = 13,  vsd = ( yet = (faw = ( ker = ( ort = 95, not = 00,  trr = ( zzz = (kkk = (uuu = 14))))))))) | Ssf=(AOE=12,FWA=14,ASF=16,VER=20),FFF=04,QWE=(AFA=(aas=13,vsd=(yet=(faw=(ker=(ort=95,not=00,trr=(zzz=(kkk=(uuu=14)))))))))  Correct. |
|  | asd = 00, fbk = 11, AAA = (OOO = 13),  FFF = 12, BBB = 94, QQQ = 93, KKK = 21,  MMM = 21, NNN = 21, DDD = (EEE = (kad = 94)),  DDD = 64, VVV = 82,HHH=65, LLL = 08, ret = 42 | asd=00,fbk=11,AAA=(OOO=13),FFF=12,BBB=94,QQQ=93,KKK=21,MMM=21,NNN=21,DDD=(EEE=(kad=94)),DDD=64,VVV=82,HHH=65,LLL=08,ret=42  Correct. |