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.

Для проверки правильности работы были созданы тесты. Тесты проверяют, что анализ происходит правильно. Сначала тестируются базовые случаи, например: проверка корректности имени, отсутвие лишних символов, присутсвие запятой между параметрами в понятии список_параметров и тд. Тестирование происходит при помощи *python*-скрипта. Запустить проверку тестов можно вручную командой *python3 test.py* или же при помощи утилиты *make: make run_tests*.

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>
#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");
            }

            void AnalyzString(const std::string& str){
                ReaderWriter* old_reader_writer = reader_writer;
                std::istringstream stream_str(str);
                ReaderWriter new_reader_writer(stream_str, std::cout);
```

```
            reader_writer = &new_reader_writer;
            StartAnalyz();
            reader_writer = old_reader_writer;
        }
    };

    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=1
3,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	Коментарии
1.	AAA = 15	AAA=15 Correct.	Этот случай проверяет, что программа корректно проверит тривиальный случай
2.	AA4 = 23	AA4 Incorrect. Name Error: Invalid name, character '4' is not letter.	Этот случай проверяет, что программа найдёт ошибку в понятии имя
3.	AAw = 2P	AAw=2P Incorrect. Parameter Error: Invalid define, character 'P' is not digit.	Этот случай проверяет, что программа найдёт в понятии параметр, а конкретнее в цифрах
4.	BBB= (RRR = 56)	BBB=(RRR=56) Correct.	Этот случай проверяет, что программа корректно проводит проверку понятия параметр=(список_параметров)
5.	AAA = BBB = 45	AAA=B Incorrect. Parameter Error: Invalid define, character 'B' is not digit.	Этот случай проверяет, что программа выдаст ошибку, если после символа «=» не следуют цифры
6.	AAA = (BBB = 13)	AAA=(BBB=13) Correct.	Этот случай проверяет, что все правильно. Входные данные являются правильной

			версией предыдущего случая
7.	AAA = (BBB = 13	AAA=(BBB=13 Incorrect. Parameter Error: Expected character ')', but end of input were reached.	Этот случай проверяет, что программа выдаст ошибку, если не закрыть все скобки
8.	AAA = (BBB = 1	AAA=(BBB=1 Incorrect. Parameter Error: Expected character ')', but end of input were reached.	Этот случай проверяет, что программа выдаст ошибку, если не закрыть все скобки
9.	AAA = 14)	AAA=14) Incorrect. List of parameters Error: Expected character ',', but were given ')'.	Этот случай, проверяет, что программа выдаст ошибку, если во входных данных будут лишние символы
10.	AAA = 14o	AAA=14o Incorrect. List of parameters Error: Expected character ',', but were given 'o'.	Этот случай, проверяет, что программа выдаст ошибку, если во входных данных будут лишние символы
11.	AAA = 14,	AAA=14, Incorrect. Name Error: Invalid name, excepted letter, but end of input were reached.	Этот случай, проверяет, что программа выдаст ошибку, если во входных данных будут лишние символы
12.	AAA 14	AAA1 Incorrect. Parameter Error: Expected character '=', but were given '1'.	Этот случай, проверяет, что прогрмма выдаст ошибку, если между именем и цифрами нет символа «=»
13.	AAA	AAA	Этот случай, проверяет,

		Incorrect. Parameter Error: Expected character '=', but end of input were reached.	что программа выдаст ошибку, если передать только понятие имя
14.	AAA = 15 , BBB = (AaA = (AAA = (AAA = 12)), FFF =10)	AAA=15,BBB=(AaA=(AAA=(AAA=12)),FFF=10) Correct.	Этот и следующие три случая связанны. В зависимости от растоновки скобок, по разному будет происходить проверка. В первых двух случаях программа не должна выдать ошибок
15.	AAA = 15 , BBB = (AaA = (AAA = (AAA = 12))), FFF =10	AAA=15,BBB=(AaA=(AAA=(AAA=12))),FFF=10 Correct.	См. описание предыдущего случая
16.	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 ')'.	См. описание случая №14. В этом случае проверяется, что программа выдаст ошибку, так как есть лишняя скобка
17.	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.	См. описание случая №14. В этом случае проверяется, что программа выдаст ошибку, так как не все скобки закрыты
18.	AAA = 15 , BBB = AaA = AAA = AAA = 12, FFF =10	AAA=15,BBB=A Incorrect. Parameter Error: Invalid define, character 'A' is not digit.	Этот случай, проверяет, что прогрмма выдаст ошибку, если в понятии параметр после символа «=» будет

			следовать имя, а не цифры
19.	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.	Этот случай, проверяет, что прогрмма не выдаст ошибку, если будет больше вложенности в понятии параметр.
20.	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.	Этот случай, проверяет, что прогрмма не выдаст ошибку, если будет больше вложенности в понятии список_параметр.