

**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. 7381

Лукашев Р.С.

Преподаватель

Фирсов М.А.

Санкт-Петербург

2018

**Цель работы:** ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++.

### **Задание.**

12. Построить синтаксический анализатор для понятия скобки.

скобки ::= квадратные | круглые | фигурные

квадратные ::= [ круглые фигурные ] | +

круглые ::= ( фигурные квадратные ) | -

фигурные ::= { квадратные круглые } | 0

**Дополнительное требование к 1-ой работе:** промежуточные данные должны выводиться с отступами, соответствующими глубине рекурсии. Должна выводиться информация о вызовах и завершениях рекурсивных функций. Желательно также явно выводить глубину рекурсии.

### **Пояснение задания.**

На вход программе подаётся последовательность символов, необходимо проверить, является ли она “скобками”, используя рекурсивный алгоритм.

### **Описание алгоритма.**

Основная часть этой программы реализована как булевская функция `isBracket`, которая вызывает три другие булевские функции `isRound`, `isSquare` и `isFigure`, определяющие, является ли текущая подпоследовательность символов частью круглые, квадратные или фигурные соответственно. Каждая из функций `isRound`, `isSquare` и `isFigure` в свою очередь может вызывать две другие.

### **Описание функций и структур данных.**

1. **bool isBracket()** – главная функция синтаксического анализатора. Сначала считывается первый символ входной последовательности (и возвращается в

поток ввода с помощью функции `ungetc()`). В зависимости от этого символа возвращается результат одной из функций `isRound()`, `isSquare()`, `isFigure()`, либо, если этот символ является некорректным, вызывается функция `error(INVALID_SC)`, выводящая сообщение о некорректности символа, и возвращается 0 (FALSE).

2. **bool isSquare()** – функция, проверяющая, является ли текущая подпоследовательность символов частью “квадратные”. После проверки конца ввода, считывается символ из потока, проверяется, является ли этот символ в прямом определении “квадратные” (в нашем случае это '+'). Далее идет проверка, попадает ли этот символ в рекурсивное определение (через “круглые” и “фигурные”) “квадратные”, т.е. если символ является '[', то поочередно вызываются функции `isRound()` и `isFigure()`, если их результат удовлетворяет определению, то происходит проверка на закрывающую скобку (']'), в случае ее наличия, возвращается TRUE, в ином случае вызывается `error(MISS_S_CL)`, выводящая сообщение об отсутствии закрывающей скобки. Если символ не является ни '+' ни '[', вызывается `error(MISS_S)`, выводящая сообщение об отсутствии этих ожидаемых символов, и возвращается FALSE. В начале функции и перед завершением функции вызывается функция `interDataGen()` с аргументами `ENTER_S` и `EXIT_S` соответственно (вывод сообщения о входе в функцию `isSquare()` и выходе из нее в файл `Data.txt`), также меняется значение переменной `depth`, характеризующей глубину рекурсии в начале вызова и конце работы функции.
3. **bool isRound()** – функция, проверяющая, является ли текущая подпоследовательность символов частью “круглые”. Функция симметрична функции `isSquare()` с изменениями в соответствии с определением “круглые” и своими аргументами функций `error()` и `interDataGen()`.
4. **bool isFigure()** – функция, проверяющая, является ли текущая подпоследовательность символов частью “фигурные”. Функция симметрична функции `isSquare()` с изменениями в соответствии с определением “фигурные” и своими аргументами функций `error()` и `interDataGen()`.

5. **void error(error\_code code)** – функция, выводящая на консоль и в файл вывода сообщение об ошибке в соответствии с кодом ошибки (code).
6. **void interDataGen(internal\_op\_code op\_code)** – дополнительная функция, выводящая в файл Data.txt информацию о входе и выходе из главных рекурсивных функций и о ошибке ввода (op\_code) в соответствии с дополнительными требованиями к лабораторной работе.
7. **void displayFileContents()** – дополнительная функция, выводящая на консоль содержимое файла ввода. Необходима для удобства работы с программой.

### Тестирование.

Входной поток данных сразу проверяется на пустоту, в случае пустого ввода выводится сообщение об ошибке (см. тест 1). Программа в процессе анализа входной последовательности выводит её либо целиком, если она правильная (см., например, тесты 2 и 3), либо до того символа, который является ошибочным (см., например, тесты 4, 5, 6 и 7). Сообщение об ошибке выводится с новой строки после знака «!».

Таблица 1 - Результат тестирования программы.

№ теста	Исходные данные:	Результат:
1		<b>!-Empty input.</b>
2	+	<b>+</b> <b>This is a Bracket</b>
3	{[(+++)0](+-)[-0]}	<b>{[(+++)0](+-)[-0]}</b> <b>This is a Bracket</b>
4	[--]	<b>[--</b> <b>!-'{' or '0' is missing.</b> <b>This is not a Bracket</b>
5	{[(++A+)0](+-)[-0]}	<b>{[(++A</b> <b>!-'(' or '-' is missing.</b> <b>This is not a Bracket</b>

Продолжение Таблицы 1.

№ теста	Исходные данные:	Результат:
6	<code>[-{+-}](</code>	<code>[-{+-}]</code> <b>!-Excess characters.</b> <b>This is not a Bracket.</b>
7	<code>A[-{+-}]</code>	<b>A</b> <b>!-Invalid starting character.</b> <b>This is not a Bracket</b>

Тест 4. В программе вызывается функция `isBracket`, считывается символ '[', и возвращается в поток ввода, затем этой функцией возвращается результат работы `isSquare`, в которой считывается тот-же символ '[', в условии вызывается `isRound`, в которой в свою очередь считывается следующий символ '-', и возвращается 1, поскольку '-' является "круглые". Условие в функции `isSquare` выполнено, и следует вызов во втором условии `isSquare` функции `isFigure`. В этой функции считывается следующий символ '-', происходят две проверки на соответствие этого символа части "фигурные", поскольку этот символ не является частью выражения, происходит вызов `error(MISS_F)`, печатающей сообщение о ошибке, и возвращается 0. Второе условие в `isSquare` не выполнено, поэтому возвращается 0. В результате функцией `isBracket` возвращается 0, и программа выводит сообщение о том, что строка не является "скобками".

### **Вывод.**

В процессе выполнения лабораторной работы были изучены и применены на практике основные приёмы и понятия рекурсивного программирования для создания программы-лексического анализатора для понятия "скобки". Программа была написана на языке C++.

## Приложение А. Код программы.

```
//Syntax analysis for brackets
#include <iostream>
#include <stdio.h>
#include <fstream>

//Global variables
int depth = 0; //recursive depth
std::ofstream interData("Data.txt", std::ofstream::out); //internal data
file
std::ofstream output("output.txt", std::ofstream::out); //output file
enum error_code {EMPTY, EXCESS, INVALID_SC, MISS_S_CL, MISS_S, MISS_R_CL,
MISS_R, MISS_F_CL, MISS_F};
enum internal_op_code {ERROR, ENTER_B, EXIT_B, ENTER_S, EXIT_S, ENTER_R,
EXIT_R, ENTER_F, EXIT_F};
//.....
//Function definitions
void error(error_code code); //Displays error message
void interDataGen(internal_op_code op_code); //Fills file Data.txt with
intermediate data (algorithm work)
void displayFileContents(); //Displays input file contents

bool isBracket(); //Bracket ::= Square | Round | Figure
bool isSquare(); //Square ::= [Round Figure] | +
bool isRound(); //Round ::= (Figure Square) | -
bool isFigure(); //Figure ::= {Square Round} | 0
//.....
//Function implementations
bool isBracket(){
    interDataGen(ENTER_B);
    char c;
    c = std::cin.get();
    std::ungetc(c, stdin);
    if(c == '[' || c == '+'){ return isSquare(); }
    if(c == '(' || c == '-'){ return isRound(); }
    if(c == '{' || c == '0'){ return isFigure(); }
    std::cout << c;
    output << c;
    error(INVALID_SC);
    return 0;
}

bool isSquare(){
    depth++;
    interDataGen(ENTER_S);
```

```

if(!std::cin.eof()){
    char input;
    input = std::cin.get();
    std::cout << input;
    output << input;

    if(input == '+'){
        interDataGen(EXIT_S);
        depth--;
        return 1;
    }

    if(input == '['){
        if(isRound()){
            if(isFigure()){
                if(!std::cin.eof()){
                    input = std::cin.get();
                    std::cout << input;
                    output << input;
                    if(input==']'){
                        interDataGen(EXIT_S);
                        depth--;
                        return 1;
                    } else error(MISS_S_CL); // "]" is
missing
                }
            }
        }
    } else error(MISS_S); // "[" or "+" is missing
}

interDataGen(EXIT_S);
depth--;
return 0;
}

bool isRound(){
    depth++;
    interDataGen(ENTER_R);

    if(!std::cin.eof()){
        char input;
        input = std::cin.get();
        std::cout << input;

```

```

        output << input;

        if(input == '-'){
            interDataGen(EXIT_R);
            depth--;
            return 1;
        }

        if(input == '('){
            if(isFigure()){
                if(isSquare()){
                    if(!std::cin.eof()){
                        input = std::cin.get();
                        std::cout << input;
                        output << input;
                        if(input==' '){
                            interDataGen(EXIT_R);
                            depth--;
                            return 1;
                        } else error(MISS_R_CL); // ")" is
missing
                    }
                }
            }
        } else error(MISS_R); // "-" or "(" is missing
    }

    interDataGen(EXIT_R);
    depth--;
    return 0;
}

bool isFigure(){
    depth++;
    interDataGen(ENTER_F);

    if(!std::cin.eof()){
        char input;
        input = std::cin.get();
        std::cout << input;
        output << input;

        if(input == '0'){
            interDataGen(EXIT_F);
            depth--;

```



```

        return 1;
    }

    if(input == '{'){
        if(isSquare()){
            if(isRound()){
                if(!std::cin.eof()){
                    input = std::cin.get();
                    std::cout << input;
                    output << input;
                    if(input=='}'){
                        interDataGen(EXIT_F);
                        depth--;
                        return 1;
                    } else error(MISS_F_CL); // "}" is
missing
                }
            }
        }
    } else error(MISS_F); // "0" or "{" is missing
}

interDataGen(EXIT_F);
depth--;
return 0;
}

void error(error_code code){
    interDataGen(ERROR);
    switch(code){
        case EMPTY: std::cout << "!-Empty input.";
                    output << "!-Empty input.";
                    break;
        case EXCESS: std::cout << "\n!-Excess characters.\nThis is not a
Bracket.";
                    output << "\n!-Excess characters.\nThis is not a Bracket.";
                    break;
        case INVALID_SC: std::cout << "\n!-Invalid starting character.";
                        output << "\n!-Invalid starting character.";
                        break;
        case MISS_S_CL: std::cout << "\n!-']' is missing.";
                        output << "\n!-']' is missing.";
                        break;
        case MISS_S: std::cout << "\n!- '[' or '+' is missing.";
                     output << "\n!- '[' or '+' is missing.";
    }
}

```

```

        break;
case MISS_R_CL: std::cout << "\n!-')' is missing.";
    output << "\n!-')' is missing.";
    break;
case MISS_R: std::cout << "\n!- '(' or '-' is missing.";
    output << "\n!- '(' or '-' is missing.";
    break;
case MISS_F_CL: std::cout << "\n!-'}' is missing.";
    output << "\n!-'}' is missing.";
    break;
case MISS_F: std::cout << "\n!- '{' or '0' is missing.";
    output << "\n!- '{' or '0' is missing.";
    break;
default: break;
}
}

void interDataGen(internal_op_code op_code){
    interData << "Depth: " << std::dec << depth << ".\t";
    for(int i = 0; i < depth; i++) interData << ".\t";
    switch(op_code){
case ERROR: interData << "INVALID INPUT\n"; break;
case ENTER_B: interData << "Entering BRACKET\n"; break;
case EXIT_B: interData << "Exiting BRACKET\n"; break;
case ENTER_S: interData << "Entering SQUARE\n"; break;
case EXIT_S: interData << "Exiting SQUARE\n"; break;
case ENTER_R: interData << "Entering ROUND\n"; break;
case EXIT_R: interData << "Exiting ROUND\n"; break;
case ENTER_F: interData << "Entering FIGURE\n"; break;
case EXIT_F: interData << "Exiting FIGURE\n"; break;
    }
}

void displayFileContents(){
    std::cout << "File contents:\n";
    std::cout << "-begin-\n";
    char c;
    c = std::cin.get();
    while(!std::cin.eof()){
        std::cout << c;
        c = std::cin.get();
    }
    std::cout << "\n-end-\n";
}
//.....

```

```

int main(int argc, char* argv[]){
    if(argc == 2){ //true: input from file. false: input from console
        std::cout << "The program is launched in file input mode.\n";
        std::cout << "To launch a program in console input mode, drag
input file on .exe file.\n";

        //redirect input stream into a file
        std::freopen(argv[1], "r", stdin);

        displayFileContents();

        //reset input stream
        std::cin.clear();
        rewind(stdin);
    } else { //input from console
        std::cout << "The program is launched in console input mode.\n";
        std::cout << "Enter a string of characters to start a syntax
analysis for Brackets:\n";
    }
    char c = std::cin.get();
    if(std::cin.eof() || (c == '\n' && argc != 2)){ //true: empty input.
false: input is not empty
        error(EMPTY); //empty input
    } else { //input is not empty, proceed.
        std::ungetc(c, stdin);
        if (isBracket()){
            interDataGen(EXIT_B);
            c = std::cin.get();
            if((!std::cin.eof() && argc == 2) || (c != '\n' && argc !=
2)) { std::cout << c; output << c; error(EXCESS); } //excess characters
            else{
                std::cout << "\nThis is a Bracket";
                output << "\nThis is a Bracket";
            }
        }
        else{
            interDataGen(EXIT_B);
            std::cout << "\nThis is not a Bracket";
            output << "\nThis is not a Bracket";
        }
    }
}

if(argc == 2)
    fclose(stdin);

```

```

std::cout << '\n';
#ifdef _WIN32
std::freopen("CONIN$", "r", stdin);
system("PAUSE");
#endif

interData.close();
output.close();
return 0;
}

```

### Приложение Б. Makefile.

```

CODE    = ./Source/lab1.cpp
OBJ      = lab1.o
EXE      = lab1
CXX      = g++
CFLAGS   = -Wall -Wextra -c -static

```

```

all: $(OBJ)
    $(CXX) $(OBJ) -o $(EXE)

```

```

$(OBJ): $(CODE)
    $(CXX) $(CFLAGS) $(CODE)

```

```

clean:
    rm $(OBJ) $(EXE)

```

### Приложение В. checker.sh.

```

#!/bin/bash
touch checker_res.txt
cp /dev/null checker_res.txt
for cfile in Tests/Correct/*; do
    echo "running test: \"Tests/Correct/$cfile\" ";
    echo "correct test $cfile:" >>checker_res.txt;
    ./lab1 $cfile >>checker_res.txt;
    echo " " >>checker_res.txt;
done;
for icfile in Tests/Incorrect/*; do
    echo "running test: \"Tests/Incorrect/$icfile\" ";
    echo "incorrect test $icfile:" >>checker_res.txt;
    ./lab1 $icfile >>checker_res.txt;
    echo " " >>checker_res.txt;
done;

```