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: Рекурсивные алгоритмы

Студент гр. 9381 _____ Чухарев И.А.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург
2020

Цель работы.

Изучить основные понятия и приёмы рекурсивного программирования; реализовать рекурсивный алгоритм на языке C++.

Задание.

Вариант 21.

Построить синтаксический анализатор для понятия *скобки*.
скобки::=квадратные | *круглые*
квадратные:: = [[*квадратные*] (*круглые*)] | B
круглые::=((*круглые*) [*квадратные*]) | A

Основные теоретические положения.

Рекурсивным называется объект, содержащий сам себя или определенный с помощью самого себя.

Выполнение работы.

Для решения поставленной задачи были написаны функции *isChar*, *isRound*, *isSquare*, *isBracket*, для вывода основной и промежуточной информации на экран и в файл — функция *log*. В данной программе функции *isRound* и *isSquare* являются взаимно рекурсивными. Для тестирования работы программы была написана функция *test*. Были разработаны следующие функции:

Функция isChar.

Принимает на вход четыре аргумента: *str* — ссылка на указатель начала строки, *c* — символ для проверки, *logfile* — файл для вывода логов (нужен для передачи его в функцию *log*) и *indent* — глубина рекурсии. Проверяет, находится ли символ *c* в начале строки. Если да, то к указателю *str* прибавляется 1 (сдвигаемся в строке на следующий символ), и функция возвращает *true*. Иначе функция возвращает *false*.

Функция isRound.

Взаимно рекурсивная функция (с функцией *isSquare*). Принимает на вход три аргумента: *str* — ссылка на указатель начала строки, *logfile* — файл для вывода логов (нужен для передачи его в функцию *log*) и *indent* — глубина рекурсии. Проверяет, находится ли выражение, соответствующее БНФ-понятию *круглая скобка*, в начале строки *str*. В начале объявляется переменная *result* со значением *false* — хранит результат работы функции. Если в начале строки стоит буква *«A»*, то переменной *result* присваивается *true* и происходит выход из функции. Иначе проверяется, нет ли в начале строки str выражение вида *„((круглые)[квадратные])“* при помощи функций *isChar* (проверка символов «(», «)», «[», «]»), *isRound* (проверка *круглые скобки*) и *isSquare* (проверка *квадратные скобки*). Если выражение в начале строки соответствует данному шаблону, то все вызваемые для проверки функции вернут *true* — в этом случае переменной *result* присваивается значение *true*. Функция возвращает переменную *result* — определяет, находятся ли в начале строки *str* круглые скобки.

Функция isSquare.

Взаимно рекурсивная функция (с функцией *isRound*). Принимает на вход три аргумента: *str* — ссылка на указатель начала строки, *logfile* — файл для вывода логов (нужен для передачи его в функцию *log*) и *indent* — глубина рекурсии. Проверяет, находится ли выражение, соответствующее БНФ-понятию *квадратная скобка*, в начале строки *str*. В начале объявляется переменная *result* со значением *false* — хранит результат работы функции. Если в начале строки стоит буква *«B»*, то переменной *result* присваивается *true* и происходит выход из функции. Иначе проверяется, нет ли в начале строки str выражение вида *„[[квадратные](круглые)]“* при помощи функций *isChar* (проверка символов «(», «)», «[», «]»), *isRound* (проверка *круглые скобки*) и *isSquare* (проверка *квадратные скобки*). Если выражение в начале строки соответствует данному шаблону, то все вызваемые для проверки функции вернут *true* — в этом случае переменной *result*

присваивается значение *true*. Функция возвращает переменную *result* — определяет, находятся ли в начале строки *str* квадратные скобки.

Функция isBrackets.

Принимает на вход два аргумента: *expression* — строка для проверки соответствия БНФ-понятию скобок и *logfile* — файл для вывода логов (нужен для передачи его в функцию *log*). Проверяет, соответствует ли строка *expression* БНФ-понятию скобок. Для проверки вызываются функции *isRound и isSquare* — если одна из этих функций вернет true, то в начале строки есть выражение, соответствующее БНФ-понятию круглых или квадратных скобок. Далее проверяется, является ли найденное выражение строкой *expression*. Если да, то функция возвращает *true*, иначе *false*.

Функция log.

Принимает на вход три аргумента: *message* — сообщение, *file* — файл для вывода сообщений и *indent* — количество отступов. Выводит сообщение *message* на экран и в файл file с некоторым отступом в размере *4 * indent* пробелов от начала строки. Ничего не возвращает.

Функция test.

Проводит тестирование программы при помощи заготовленных тестов, находящихся в файле. На вход принимает *path* — путь к файлу с тестами. Для начала открывает файл, если не удалось открыть — происходит выход из функции. Далее из файла тестов происходит считывание входных данных и правильного результата работы, которые находятся на одной строке, разделенные символом «|», и их проверка на тестируемой функции с выводом информации о результатах. В конце происходит закрытие файла и выход из функции.

Функция main.

Для начала объявляются следующе переменные:

- *logfile* — файл для вывода логов.
- *expression* — хранит строку для проверки соответствия БНФ-понятию.

Далее происходит считывание выбранного пользователем действия: 1 – ввести выражение с консоли, 2 – считать выражение с файла, 3 – выполнить автоматическое тестирование программы.

Далее в зависимости от значения переменной *action* происходит либо тестирование программы при помощи функции *test,* после чего происходит выход из программы, либо считывание входных данных с консоли, либо считывание входных данных из файла. После получения анализируемой строки *expression*, происходит вызов функции *isBracket* для получения результата для поставленной задачи. В конце происходит вывод результата и завершение работы программы.

Разработанный программный код см. в приложении А.

Результаты тестирования см. в приложении Б.

Выводы.

Были изучены основные понятия и приёмы рекурсивного программирования, получены навыки программирования рекурсивных функций на языке программирования C++.

Разработан синтаксический анализатор, выполняющий проверку соответствия БНФ-понятию скобок некоторого выражения, подаваемого на вход программе. Две функции из набора функций проверки получились взаимно рекурсивными.

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <conio.h>
#include "functions.h"

void test(const std::string& path, std::ofstream& logfile) {
    size_t testCount = 0; // Общее количество тестов
    size_t testCountSuccess = 0; // Количество успешных тестов
    std::ifstream file(path);

    // Проверка открытия файла
    if (!file.is_open()) {
        log("Cannot open file: " + path + "\n", logfile);
        return;
    }

    log("File with tests: " + path + "\n", logfile);

    while (!file.eof()) { // Пока не пройдемся по всем строкам файла
        std::string line;
        file >> line;

        // Поиск и проверка разделителя
        size_t separatorIndex = line.find('|');
        if (separatorIndex != -1) {
            std::string expression = line.substr(0, separatorIndex); //
Входная строка
            bool correctResult = line.substr(separatorIndex + 1,
separatorIndex + 2) == "T"; // Корректный результат теста
            bool result = isBrackets(expression, logfile); // Результат
теста

            // Вывод сравнения результата с корректным результатом теста
            if (correctResult == result) {
                testCountSuccess++;
                log("[Test №" + std::to_string(++testCount) + " OK] ",
logfile);
            } else {
                log("[Test №" + std::to_string(++testCount) + " WRONG] ",
logfile);
            }

            // Вывод результата теста
            if (result) {
                log("Result: '" + expression + "' is brackets.\n\n",
logfile);
            } else {
                log("Result: '" + expression + "' is not brackets.\n\n",
logfile);
            }
        }
    }

    log("Passed tests: " + std::to_string(testCountSuccess) + "/" +
std::to_string(testCount) + "\n", logfile);
}
```

```
int main(int argc, char* argv[]) {
    std::ofstream logfile("log.txt");
    std::string expression;

    if (!logfile.is_open()) {
        std::cout << "Cannot open file: log.txt\n";
        _getch();
        return 0;
    }

    // Считывание выбора действия пользователя
    log("Available actions:\n\n  1) Read expression from console.\n  2)
Read expression from file.\n  3) Run testing.\n\nSelect the action: ", logfile);

    int action = -1;
    std::cin >> action;

    while (action < 1 || action > 3) {
        log("Incorrect action. Select the action again: ", logfile);
        std::cin >> action;
    }

    // Тестирование алгоритма при помощи набора тестов
    if (action == 3) {
        test("tests.txt", logfile);
        _getch();
        return 0;
    }

    if (action == 2) { // Ввод выражения из файла
        std::ifstream file("input.txt");

        // Проверка открытия файла
        if (!file.is_open()) {
            log("Cannot open file: input.txt\n", logfile);
            _getch();
            return 0;
        } else {
            file >> expression;
            log("Expression from file: " + expression + "\n", logfile);
        }

    } else { // Ввод выражения с клавиатуры
        std::cout << "[Enter expression] ";
        std::cin >> expression;
        log("Entered expression: " + expression + "\n", logfile);
    }

    // Вывод результата работы программы на экран
    if (isBrackets(expression, logfile)) {
        log("Result: '" + expression + "' is brackets.\n", logfile);
    } else {
        log("Result: '" + expression + "' is not brackets.\n", logfile);
    }

    _getch();
    return 0;
}
```

Название файла: functions.h

```
#ifndef FUNCTIONS_H
```

```
#define FUNCTIONS_H

#include <string>

void log(const std::string& message, std::ofstream& file, int indent = 0);
bool isChar(const char*& str, char c, std::ofstream& logfile, int indent =
0);
bool isRound(const char*& str, std::ofstream& logfile, int indent = 0);
bool isSquare(const char*& str, std::ofstream& logfile, int indent = 0);
bool isBrackets(const std::string& expression, std::ofstream& logfile);

#endif // FUNCTIONS_H
```

Название файла: functions.cpp

```
#include <iostream>
#include <fstream>
#include "functions.h"

// Построить синтаксический анализатор для понятия скобки.
// скобки::=квадратные | круглые
// квадратные:: = [ [ квадратные ] ( круглые ) ] | B
// круглые::=( ( круглые ) [ квадратные ] ) | A

void log(const std::string& message, std::ofstream& file, int indent) {
    std::cout << std::string(4 * indent, ' ') << message;
    file << std::string(4 * indent, ' ') << message;
}

bool isChar(const char*& str, char c, std::ofstream& logfile, int indent)
{
    const char* start = str; // Первый символ в строке до начала работы
функции
    bool result = false;

    log("Function isChar('" + std::string(1, *str) + "', '" + c + "') is
called [deep=" + std::to_string(indent) + "]\n", logfile, indent);

    if (*str == c) {
        str++; // Идем на следующий символ
        result = true;
    }

    // Вывод результата
    if (result) {
        log("Result: '" + std::string(1, *start) + "' is '" + c + "'\n",
logfile, indent);
    } else {
        log("Result: '" + std::string(1, *start) + "' is not '" + c +
"'\n", logfile, indent);
    }

    return result;
}

bool isRound(const char*& str, std::ofstream& logfile, int indent) {
    const char* start = str; // Первый символ в строке до начала работы
функции
    bool result = false;

    log("Function isRound() is called [deep=" + std::to_string(indent) +
"]\n", logfile, indent);
```

```
          if (isChar(str, 'A', logfile, indent + 1)) {
              result = true;
          } else if (isChar(str, '(', logfile, indent + 1) && isChar(str, '(',
logfile, indent + 1) &&
                     isRound(str, logfile, indent + 1) && isChar(str, ')',
logfile, indent + 1) &&
                     isChar(str, '[', logfile, indent + 1) && isSquare(str,
logfile, indent + 1) &&
                     isChar(str, ']', logfile, indent + 1) && isChar(str, ')',
logfile, indent + 1)) {
              result = true;
          }

          if (!result) {
              str = start;
          }

          std::string round(start, str - start); // Найденные функцией круглые
скобки

          // Вывод результата
          if (result) {
              log("Result: '" + round + "' is round\n", logfile, indent);
          } else {
              log("Result: is not round\n", logfile, indent);
          }

          return result;
      }

      bool isSquare(const char*& str, std::ofstream& logfile, int indent) {
          const char* start = str; // Первый символ в строке до начала работы
функции
          bool result = false;

          log("Function isSquare() is called [deep=" + std::to_string(indent) +
"]\n", logfile, indent);

          if (isChar(str, 'B', logfile, indent + 1)) {
              result = true;
          } else if (isChar(str, '[', logfile, indent + 1) && isChar(str, '[',
logfile, indent + 1) &&
                     isSquare(str, logfile, indent + 1) && isChar(str, ']',
logfile, indent + 1) &&
                     isChar(str, '(', logfile, indent + 1) && isRound(str,
logfile, indent + 1) &&
                     isChar(str, ')', logfile, indent + 1) && isChar(str, ']',
logfile, indent + 1))
          {
              result = true;
          }

          if (!result) {
              str = start;
          }

          std::string square(start, str - start); // Найденные функцией
квадратные скобки

          // Вывод результата
          if (result) {
              log("Result: '" + square + "' is square\n", logfile, indent);
          } else {
```

```
            log("Result: is not square\n", logfile, indent);
        }

        return result;
    }

    bool isBrackets(const std::string& expression, std::ofstream& logfile) {
        const char* end = expression.c_str(); // Получение C-style строки
        bool result = false;

        log("Function isSquare() is called [deep=0]\n", logfile);

        if (isRound(end, logfile, 1) || isSquare(end, logfile, 1)) {
            if (*end == '\0') { // Проверяем, что вся строка соответствует
определению
                result = true;
            }
        }

        return result;
    }
```

ПРИЛОЖЕНИЕ Б

ТЕСТИРОВАНИЕ

Таблица Б.1 - Примеры тестовых случаев

№ п/п	Входные данные	Выходные данные	Комментарии
1.	UNCORRECT_DATA	Result: 'UNCORRECT_DATA' is not brackets.	
2.	[]	Result: '[]' is not brackets.	
3.	()	Result: '()' is not brackets.	
4.	[A]	Result: '[A]' is not brackets.	
5.	(B)	Result: '(B)' is not brackets.	
6.	[[]()]	Result: '[[]()]' is not brackets.	
7.	[()[]]	Result: '[()[]]' is not brackets.	
8.	(()[])	Result: '(()[])' is not brackets.	
9.	([]())	Result: '([]())' is not brackets.	
10.	[[A](B)]	Result: '[[A](B)]' is not brackets.	
11.	[[B](A)]	Result: '[[B](A)]' is brackets.	
12.	[(A)[B]]	Result: '[(A)[B]]' is not brackets.	
13.	[(B)[A]]	Result: '[(B)[A]]' is not brackets.	
14.	((A)[B])	Result: '((A)[B])' is brackets.	
15.	((B)[A])	Result: '((B)[A])' is not brackets.	
16.	([A](B))	Result: '([A](B))' is not brackets.	
17.	([B](A))	Result: '([B](A))' is not brackets.	
18.	((((B)[A]))[B])	Result: '((((B)[A]))[B])' is not brackets.	
19.	((A)[[[B](A)]])	Result: '((A)[[[B](A)]])' is brackets.	

Файл с тестами: tests.txt

```
UNCORRECT_DATA|F
[]|F
()|F
[A]|F
(B)|F
[[]()]|F
[()[]]|F
(()[])|F
([]())|F
[[A](B)]|F
[[B](A)]|T
[(A)[B]]|F
[(B)[A]]|F
((A)[B])|T
((B)[A])|F
([A](B))|F
([B](A))|F
((((B)[A]))[B])|F
```

```
((A)[[[B](A)]])|T
```