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 _____ Соболева К.С.

Преподаватель _____ Фирсов М.А.

Санкт-Петербург

2020

Цель работы.

Ознакомиться с принципами рекурсии, реализовать рекурсивный алгоритм на языке C++.

Задание.

Вариант 15.

Построить синтаксический анализатор для понятия скобки.

скобки::=A | A (ряд_скобок)

ряд_скобок::= скобки | скобки ; ряд_скобок

Описание алгоритма.

В основе работы синтаксического анализатора для понятия «скобки» лежит поверка двух двух понятий: скобки и ряд. Для этого реализованы 2 взаимно-рекурсивные функции, возвращающие 4 понятия: СКОБКИ, НЕСКОБКИ, РЯД, НЕРЯД. Функции устроены таким образом, что первой, исходя из задания, будет вызываться функция проверки для «скобки», и соответственно выход из данных взаимно-рекурсивных рекурсивных функций будет с результатом СКОБКИ или НЕСКОБКИ. А понятия РЯД и НЕРЯД будут выполнять вспомогательную функцию для всего анализатора.

Для **понятия «скобки»** происходят соответственно проверки на:

«А» : если совпадает — идет дальше, нет — НЕСКО;

«(» : если совпадает – идет дальше, если не «(», то считается, что текущее понятие ограничивается символом "А" и возвращается результат СКОБКИ, а следующий символ будет обработан функцией уровнем выше;

«ряд»: если совпадает — дальше, нет — НЕСКО;

«)» : если совпадает — СКОБКИ, нет — НЕСКО.

Для **понятия «ряд»** происходят соответственно проверки на:

«скобки» : вызывается функция для понятия «скобки», если результат СКОБКИ — идет дальше, если НЕСКОБКИ — НЕРЯД;

«;» : если совпадает – идет дальше, если не «;», то происходит ситуация аналогичная «(» для понятия «скобки»

«ряд»: вызывается функция для понятия «ряд», если результат РЯД — возвращает РЯД, если НЕРЯД — НЕРЯД.

После такого как отработают данные 2 взаимно-рекурсивные функции необходимо исключить оставшийся плохой вариант – когда возможна строка наподобие СКОБКИ…, где … - какие-то еще символы. Для этого необходимо проверить все ли символы, переданной в анализатор строки, проверены. Если да, то выводится окончательный результат – СКОБКИ, иначе – НЕ СКОБКИ.

Функции и структуры данных.

Перечисления:

- enum Name{NOTBRACKETS, BRACKETS, NOTROW, ROW, EMPTY}; соответственно: НЕСКОБКИ, СКОБКИ, НЕРЯД, РЯД, ПУСТАЯ; Необходимо для обработки результата работы анализатора.

Функции:

- void printTask()

  Печать условий задания.

- void readConsole(std::ofstream& fileOut)

  Обрабатывает n введенных с клавиатуры строк, где число n вводит пользователь. Считывается строка при помощи функции getline, а затем передается в функцию processingResults для обработки. При этом так же происходит проверка на пустую(EMPTY) строку. Если пустая — она не будет входит в n строк.

- void readFile(std::ifstream& file, std::ofstream& fileOut);

  Считываются строки из файла при помощи функции getline, а затем передаются в функцию processingResults для обработки.

- int processingResults(const std::string& str, std::ofstream& fileOut); Печать полученного результата из функции processingSentence(скобки или не скобки) в консоль и в файл, так же возможно получение результата пустая строка.

- int processingSentence(const std::string& str, std::ofstream& fileOut);

  Происходит проверка на нулевую строку, при положительном результате такие строки не обрабатываются. Далее вызывается уже функция непосредственно необходимая для проверки понятия «скобки». Так же необходимо проверить все ли символы в строке обработаны, т.е. исключить все случаи вроде СКОБКИ…, где … - какие-то еще символы. Если положительно выполняются эти два условия, функция возвращает результат BRACKETS, иначе NOTBRACKETS .

- int brackets(const std::string& str, int& pos, int depthRec, std::ofstream& fileOut);

Рекурсивная функция для понятия СКОБКИ.

Если символ не равен «А», то возвращается не скобки. Далее проверка на «(», при отрицательном результате выход из функции с результатом скобки, но далее этот результат так же будет обработан в функции, предшествующей данной. Сделано так из-за того, что скобки могут быть не только формата «А(ряд)», но и просто «А», поэтому отсутствие «(» еще не говорит об отрицательном результате, просто необходима дальнейшая проверка. В случае «(» далее идет проверка на ряд(т.е. вызов еще одной рекурсивной функции). Если не ряд, то возврат не скобки, в противном — проверка на «)». Если не проходит эта проверка, то возврат не скобки, иначе возврат скобки.

На протяжении всей функции несколько раз вызывается функция printDepth для вывода промежуточных результатов.

- int row(const std::string&, int& pos, int depthRec, std::ofstream& fileOut);

  Рекурсивная функция для понятия РЯД_СКОБОК.

  Происходит проверка на понятие скобки(т.е. вызов функции brackets), при отрицательном результате происходит возврат не ряд. Иначе дальше идет проверка на «;». Если не «;», то возвращает ряд, но далее этот результат так же будет обработан в функции, предшествующей данной(по аналогии с таким же пунктом из функции brackets). Последней будет проверка на понятие «ряд»: если проходит — ряд, нет — не ряд.

  На протяжении всей функции несколько раз вызывается функция printDepth для вывода промежуточных результатов.

- void printDepth(const std::string& str, int& pos, int depthRec, std::ofstream& fileOut);

  Функция для печати(в консоль и файл) промежуточных результатов. Std::string(depthRec, '\t') позволяет сделать вывод с отступами, соответствующими глубине рекурсии.

- int main()

  Пользователю предлагается способ ввода: консоль или файл.

В случае файлового ввода пользователю предлагается ввести имена файлов для чтения и вывода. Происходит проверка корректного открытия файла для чтения. Если произошла ошибка, то пользователю предлагается ввести новое имя. Так же тут присутствует возможность досрочно завершить программу путем ввода определенной команды. В случае корректного открытия файла для чтения далее происходит вызов функции, запускающей обработку файлового ввода.

В случае консольного ввода необходимо ввести только файл для записи. Далее вызов функции, запускающей обработку консольного ввода.

После отработки программа предложит пользователю повторить обработку, и тогда все начнется с самого начала, или закончить тестирование.

Тестирование.

№	Входные данные	Вывод
1	A	STR: A Глубина рекурсии: 1 A Скобки
2	A(A)	STR: A(A) Глубина рекурсии: 1 A Глубина рекурсии: 1 A(Глубина рекурсии: 3 A(A Глубина рекурсии: 1 A(A) Скобки
3	A(A;A)	STR: A(A;A) Глубина рекурсии: 1 A Глубина рекурсии: 1 A(Глубина рекурсии: 3 A(A Глубина рекурсии: 2 A(A; Глубина рекурсии: 4 A(A;A

		Глубина рекурсии: 1 A(A;A) Скобки
4	A(A(A(A)))	STR: A(A(A(A))) Глубина рекурсии: 1 A Глубина рекурсии: 1 A(Глубина рекурсии: 3 A(A Глубина рекурсии: 3 A(A(Глубина рекурсии: 5 A(A(A Глубина рекурсии: 5 A(A(A(Глубина рекурсии: 7 A(A(A(A Глубина рекурсии: 5 A(A(A(A) Глубина рекурсии: 3 A(A(A(A)) Глубина рекурсии: 1 A(A(A(A))) Скобки
5	A(A(A;A))	STR: A(A(A;A)) Глубина рекурсии: 1 A Глубина рекурсии: 1 A(Глубина рекурсии: 3 A(A Глубина рекурсии: 3 A(A(Глубина рекурсии: 5 A(A(A Глубина рекурсии: 4 A(A(A; Глубина рекурсии: 6 A(A(A;A Глубина рекурсии: 3 A(A(A;A) Глубина рекурсии: 1 A(A(A;A)) Скобки
6	A(A;A);	STR: A(A;A); Глубина рекурсии: 1 A Глубина рекурсии: 1 A(Глубина рекурсии: 3 A(A Глубина рекурсии: 2 A(A; Глубина рекурсии: 4 A(A;A Глубина рекурсии: 1 A(A;A) Не скобки

7	A(A;	STR: A(A; Глубина рекурсии: 1 A Глубина рекурсии: 1 A(Глубина рекурсии: 3 A(A Глубина рекурсии: 2 A(A; Не скобки
8	c	STR: c Не скобки
9	A(A()A)	STR: A(A()A) Глубина рекурсии: 1 A Глубина рекурсии: 1 A(Глубина рекурсии: 3 A(A Глубина рекурсии: 3 A(A(Не скобки

Выводы.

Были изучены принципы рекурсии. Построен синтаксический анализатор для понятия скобки с использованием рекурсивного алгоритма на языке C++.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>
#include <string>
#include <iomanip>
//НЕСКОБКИ, СКОБКИ, НЕРЯД, РЯД, ПУСТАЯ - в зависимости от этих элементов
перечисления происходит обработка результата
enum Name{NOTBRACKETS, BRACKETS, NOTROW, ROW, EMPTY};

void printTask();
void readConsole(std::ofstream& fileOut);//считываются строки с клавиатуры
и передаются в функцию для обработки
void readFile(std::ifstream& file, std::ofstream& fileOut);//считываются
строки из файла и передаются в функцию для обработки
int processingResults(const std::string& str, std::ofstream&
fileOut);//печать полученного результата из функции processingSentence
int processingSentence(const std::string& str, std::ofstream&
fileOut);//функция, передающая строку в анализатор скобок и возвращающая
результат - скобки, не скобки или пустая строка
int brackets(const std::string& str, int& pos, int depthRec,
std::ofstream& fileOut);//рекурсивная функция для понятия СКОБКИ
int row(const std::string&, int& pos, int depthRec, std::ofstream&
fileOut);//рекурсивная функция для понятия РЯД_СКОБОК
void printDepth(const std::string& str, int& pos, int depthRec,
std::ofstream& fileOut);//функция для печати промежуточных результатов

void printTask(){
    std::cout<<"скобки::=A | A ( ряд_скобок )\n"
               "ряд_скобок::= скобки | скобки ; ряд_скобок\n";
}

void readConsole(std::ofstream& fileOut){
    int n; // количество строк
    std::string str;
    std::cout<<"Введите количество строк\n";
    std::cin>>n;
    for(int i=0; i < n; i++){
        getline(std::cin, str);
        if (processingResults(str, fileOut) == EMPTY){ //проверка на
пустую строку
            i--;
        }

    }
}

void readFile(std::ifstream& file, std::ofstream& fileOut){
    std::string str;
    while (file) { //для построчного считывания из файла; пока есть строки
- идет считывание
        getline(file, str); //считывается обрабатываемая строка
        processingResults(str, fileOut);
    }
}


int processingResults(const std::string& str, std::ofstream& fileOut){
    switch (processingSentence(str, fileOut)){
```

```
        case BRACKETS:
            std::cout<<"Скобки\n\n";
            fileOut<<"Скобки\n\n";
            return BRACKETS;
        case NOTBRACKETS:
            std::cout << "Не скобки\n\n";
            fileOut << "Не скобки\n\n";
            return NOTBRACKETS;
        case EMPTY:
            return EMPTY;
        default:
            return NOTBRACKETS;
    }
}

int processingSentence(const std::string& str, std::ofstream& fileOut){
    if (str.length()==0) { // нулевые строки не обрабатываются
        return EMPTY;
    }
    std::cout << "STR: " << str << "\n";
    fileOut << "STR: " << str << "\n";
    int pos = 0; // обрабатываемая позиция
    int depthRec = 1;//глубина рекурсии
    if (brackets(str, pos, depthRec, fileOut) == BRACKETS && pos ==
(str.length())) { //смотрит результат анализатора и проверяет, все ли
символы
        return BRACKETS;
//обработаны, т.е. не вышла ли функция, не обработав последние символы
    } else {
        return NOTBRACKETS;
    }
}


int brackets(const std::string& str, int& pos, int depthRec,
std::ofstream& fileOut){ //анализатор СКОБКИ
    if (str[pos++] != 'A'){
        return NOTBRACKETS;
    }
    printDepth(str, pos, depthRec, fileOut);  //печать промежуточного
результата
    if (str[pos] != '('){
        return BRACKETS;
    }
    pos++;
    printDepth(str, pos, depthRec, fileOut);  //печать промежуточного
результата
    if (row(str, pos, depthRec+1, fileOut)==NOTROW) { //вызов рекурсивной
функции для проверки на понятие РЯД
        return NOTBRACKETS;
    }
    if(str[pos++] != ')'){
        return NOTBRACKETS;
    }
    printDepth(str, pos, depthRec, fileOut);  //печать промежуточного
результата
    return BRACKETS;
}

int row(const std::string& str, int& pos, int depthRec, std::ofstream&
fileOut){
    if (brackets(str, pos, depthRec+1, fileOut)==NOTBRACKETS){ //вызов
рекурсивной функции для проверки на понятие СКОБКИ
        return NOTROW;
```

```
    }
    if (str[pos] != ';'){
        return ROW;
    }
    pos++;
    printDepth(str, pos, depthRec, fileOut);  //печать промежуточного
результата
    if (row(str, pos, depthRec+1, fileOut)==NOTROW){ //вызов рекурсивной
функции для проверки на понятие РЯД
        return NOTROW;
    }
    return ROW;
}

void printDepth(const std::string& str, int& pos, int depthRec,
std::ofstream& fileOut){
    std::cout<<std::string(depthRec, '\t')<<"Глубина
рекурсии:"<<std::setw(2)<<depthRec<<"\t"<<str.substr(0, pos)<<"\n";
    fileOut<<std::string(depthRec, '\t')<<"Глубина
рекурсии:"<<std::setw(2)<<depthRec<<"\t"<<str.substr(0, pos)<<"\n";
}

int main() {
    int flag = 1;
    while (flag==1) {
        printTask();
        int input = 0;
        std::string nameIn, nameOut;
        std::ifstream fileIn;
        std::ofstream fileOut;
        std::cout << "\nВыберите вариант ввода текста:\n"
                     "1 - файловый ввод\n"
                     "2 - консольный ввод\n";

        std::cin >> input;
        switch (input) {
            case 1: //файловый
                std::cout << "Введите имена файлов для считывания и записи
\n";
                std::cin >> nameIn >> nameOut;
                fileIn.open(nameIn);
                fileOut.open(nameOut);
                while (!fileIn) { //проверка существования файла
                    std::cout
                            << "Не получилось открыть файл для считывания.
Введите новое имя файла. Для выхода введите \'0\'\n";
                    std::cin >> nameIn;
                    if (nameIn == "0") {//команда для выхода из цикла
ввода файла
                        return 0;
                    }
                    fileIn.open(nameIn); //открытие файла

                }
                readFile(fileIn, fileOut); //запуск построчного чтения и
анализатора
                fileIn.close(); //закрытие файла для чтения
                fileOut.close();//закрытие файла для записи
                break;
            case 2: //консоль
                std::cout << "Введите имя файла для записи \n";
                std::cin >> nameOut;
                fileOut.open(nameOut);
                readConsole(fileOut);
```

```
                fileOut.close();//закрытие файла для записи
                break;
            default:
                break;
        }
        std::cout<<"Хотите повторить?\n"
                   "1 - да\n"
                   "0 - нет\n";
        std::cin >> flag;
    }

    return 0;
}
```