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: Программирование рекурсивных алгоритмов

Студент гр. 9303 _____ Куршев Е.О

Преподаватель _____ Филатов Ар.Ю

Санкт-Петербург

2020

Цель работы.

Создание рекурсивного алгоритма для анализа понятия «скобки».

Задание

10. Построить синтаксический анализатор для определяемого далее понятия константное_выражение.

константное_выражение::=ряд_цифр|

константное_выражение знак_операции константное_выражение

знак_операции::=+|-|*

ряд_цифр::=цифра|цифра ряд_цифр

Основные теоретические положения.

Рекурсивным называется объект, содержащий сам себя или определенный с помощью самого себя. Мощность рекурсии связана с тем, что она позволяет определить бесконечное множество объектов с помощью конечного высказывания. Точно так же бесконечные вычисления можно описать с помощью конечной рекурсивной программы. Рекурсивные алгоритмы лучше всего использовать, когда решаемая задача, вычисляемая функция или обрабатываемая структура данных определены с помощью рекурсии. Если процедура (функция) P содержит явное обращение к самой себе, она называется прямо рекурсивной. Если P содержит обращение к процедуре (функции) Q, которая содержит (прямо или косвенно) обращение к P, то P называется косвенно рекурсивной. Многие известные функции могут быть определены рекурсивно. Например факториал, который присутствует практически во всех учебниках по программированию, а также наибольший общий делитель, числа Фибоначчи, степенная функция и др.

Выполнение работы

В ходе выполнения работы были реализованы следующие функции для работы анализатора:

1. bool is_operation_sign(std::string &str, std::ofstream &file); - функция, проверяющая текущий символ на предмет того, является ли он одним из трёх знаков операции ("+", "-", "*").
2. bool is_row_of_numbers(std::string &str, std::ofstream &file); - функция, проверяющая кусок строки на предмет того, является ли это последовательность рядом цифр.
3. bool analyzer(std::string &str, std::ofstream &file); - стартовая точка программы. В ней происходит проверка первого символа, который обязательно должен быть цифрой.
4. void Errors(int error, std::string &str, std::ofstream &file); - функция вывода на экран ошибок во входных строках.

Выводы.

Была реализована программа, включающая в себя рекурсивные функции is_operation_sign и is_row_of_numbers для синтаксического анализа выражения «константное выражение». Также было проведено тестирование программы. Были удовлетворены следующие требования: информация на вход подается из файла, выводятся текущие сведения программы, также происходит запись всех сведений, полученных в ходе выполнения программы, в файл с названием "Output.txt".

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <fstream>

static int current_index = 0;

bool is_operation_sign(std::string &str, std::ofstream &file);
bool is_row_of_numbers(std::string &str, std::ofstream &file);
bool analyzer(std::string &str, std::ofstream &file);
void Errors(int error, std::string &str, std::ofstream &file);

int main(){
    std::string file_name, parsed_string;
    std::cin >> file_name;
    std::fstream input_file(file_name);
    std::ofstream out;
    out.open("Output.txt");

    if (input_file.is_open())
        std::cout << "Success! File open!" << std::endl;
    else{
        std::cout << "Error! Wrong name of input file!!" <<
std::endl;
        return 0;
    }

    while(getline(input_file, parsed_string)){
        current_index = 0;
        out << "\"" << parsed_string << "\":\n";
        std::cout << "\"" << parsed_string << "\":\n";
        if (analyzer(parsed_string, out)){
            out << "Wow! Expression \"" << parsed_string <<
"\" is a constant expression!\n\n";
```

```
                std::cout << "Wow! Expression \"" << parsed_string
<< "\" is a constant expression!\n\n";
            }
        }
        out.close();
        return 0;
    }

    bool analyzer(std::string &str, std::ofstream &file){
        if(!isdigit(str[0])){
            Errors(3, str, file);
            return false;
        }
        else
            return is_row_of_numbers(str, file);
    }

    bool is_operation_sign(std::string &str, std::ofstream &file){
        if ((str[current_index] == '*') || (str[current_index] ==
'+') || (str[current_index] == '-')) {
            std::cout << "Current symbol: operation sign \"" <<
str[current_index] << "\"\n";
            file << "Current symbol: operation sign \"" <<
str[current_index] << "\"\n";
            if((str[current_index - 1] == '*') ||
(str[current_index - 1] == '+') || (str[current_index - 1] ==
'-')){
                Errors(1, str, file);
                return false;
            }
            current_index += 1;
            if(current_index == str.length()){
                Errors(2, str, file);
                return false;
            }
```

```
            return is_row_of_numbers(str, file);
        }
        else{
            file << "Current symbol: unknown symbol \"" <<
str[current_index] << "\"\n";
            std::cout << "Current symbol: unknown symbol \"" <<
str[current_index] << "\"\n";
            Errors(3, str, file);
            return false;
        }
    }

    bool is_row_of_numbers(std::string &str, std::ofstream &file){
        if(current_index == str.length())
            return true;
        if(isdigit(str[current_index])){
            file << "Current symbol: numeral \"" <<
str[current_index] << "\"\n";
            std::cout << "Current symbol: numeral \"" <<
str[current_index] << "\"\n";
            current_index += 1;
            return is_row_of_numbers(str, file);
        }
        else
            return is_operation_sign(str, file);
    }

    void Errors(int error, std::string &str, std::ofstream &file){
        switch(error){
            case 1:
                file << "Error! Two signs of operation in a row!\
n\n";
                std::cout << "Error! Two signs of operation in a
row!\n\n";
                break;
```

```
            case 2:
                file << "Error! The expression ends with an
operation character!\n\n";
                std::cout << "Error! The expression ends with an
operation character!\n\n";
                break;
            default:
                file << "Error! An unknown symbol has been
encountered!\n\n";
                std::cout << "Error! An unknown symbol has been
encountered!\n\n";
        }
    }
```

ПРИЛОЖЕНИЕ А
ИСХОДНЫЙ КОД ПРОГРАММЫ

Входной файл: input.txt

```
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BBBBBBBBB
3+7*5-4
333
3+6+8-2843989*849023-5
3+7*5-4
8****************
8+39*j-0320+12
23456+
12++----12
12--12+3289-22
-12+34
122*1+
```

Выходной файл: Output.txt

```
"AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA":
Error! An unknown symbol has been encountered!

"BBBBBBBBB":
Error! An unknown symbol has been encountered!

"3+7*5-4":
Current symbol: numeral "3"
Current symbol: operation sign "+"
Current symbol: numeral "7"
Current symbol: operation sign "*"
Current symbol: numeral "5"
Current symbol: operation sign "-"
Current symbol: numeral "4"
Wow! Expression "3+7*5-4" is a constant expression!
```

```
"333":
Current symbol: numeral "3"
Current symbol: numeral "3"
Current symbol: numeral "3"
Wow! Expression "333" is a constant expression!

"3+6+8-2843989*849023-5":
Current symbol: numeral "3"
Current symbol: operation sign "+"
Current symbol: numeral "6"
Current symbol: operation sign "+"
Current symbol: numeral "8"
Current symbol: operation sign "-"
Current symbol: numeral "2"
Current symbol: numeral "8"
Current symbol: numeral "4"
Current symbol: numeral "3"
Current symbol: numeral "9"
Current symbol: numeral "8"
Current symbol: numeral "9"
Current symbol: operation sign "*"
Current symbol: numeral "8"
Current symbol: numeral "4"
Current symbol: numeral "9"
Current symbol: numeral "0"
Current symbol: numeral "2"
Current symbol: numeral "3"
Current symbol: operation sign "-"
Current symbol: numeral "5"
Wow! Expression "3+6+8-2843989*849023-5" is a constant
expression!

"3+7*5-4":
Current symbol: numeral "3"
Current symbol: operation sign "+"
```

```
Current symbol: numeral "7"
Current symbol: operation sign "*"
Current symbol: numeral "5"
Current symbol: operation sign "-"
Current symbol: numeral "4"
Wow! Expression "3+7*5-4" is a constant expression!

"8***************":
Current symbol: numeral "8"
Current symbol: operation sign "*"
Current symbol: operation sign "*"
Error! Two signs of operation in a row!

"8+39*j-0320+12":
Current symbol: numeral "8"
Current symbol: operation sign "+"
Current symbol: numeral "3"
Current symbol: numeral "9"
Current symbol: operation sign "*"
Current symbol: unknown symbol "j"
Error! An unknown symbol has been encountered!

"23456+":
Current symbol: numeral "2"
Current symbol: numeral "3"
Current symbol: numeral "4"
Current symbol: numeral "5"
Current symbol: numeral "6"
Current symbol: operation sign "+"
Error! The expression ends with an operation character!

"12+++----12":
Current symbol: numeral "1"
Current symbol: numeral "2"
Current symbol: operation sign "+"
```

```
Current symbol: operation sign "+"
Error! Two signs of operation in a row!

"12--12+3289-22":
Current symbol: numeral "1"
Current symbol: numeral "2"
Current symbol: operation sign "-"
Current symbol: operation sign "-"
Error! Two signs of operation in a row!

"-12+34":
Error! An unknown symbol has been encountered!

"122*1+":
Current symbol: numeral "1"
Current symbol: numeral "2"
Current symbol: numeral "2"
Current symbol: operation sign "*"
Current symbol: numeral "1"
Current symbol: operation sign "+"
Error! The expression ends with an operation character!
```