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

Студентка гр. 8381 _____ Звегинцева Е.Н.

Преподаватель _____ Жангиров Т.Р.

Санкт-Петербург
2019

Цель работы.

Познакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++. Разработать программу, использующую рекурсию.

Задание.

Вариант 6.

Построить синтаксический анализатор для понятия *простое выражение*.
Простое_выражение:: = простой_идентификатор | (простое_выражение знак_операции простое_выражение)
Простой_идентификатор:: = буква
Знак_операции:: = - | + | *

Основные теоретические положения.

Любая функция (метод) в своем теле может вызывать сама себя. *Рекурсия* – это такой способ определения функции, при котором результат возврата из функции для данного значения аргумента определяется на основе результата возврата из той же функции для предыдущего (меньшего или большего) значения аргумента.

Если функция (метод) вызывает сам себя, то такой вызов называется рекурсивным вызовом функции. При каждом рекурсивном вызове запоминаются предыдущие значения внутренних локальных переменных и полученных параметров функции. Чтобы следующий шаг рекурсивного вызова отличался от предыдущего, значение как-минимум одного из параметров функции должно быть изменено. Остановка процесса рекурсивных вызовов функции происходит, когда изменяемый параметр достиг некоторого конечного значения, например, обработан последний элемент в массиве.

Рекурсивное обращение к функции может быть осуществлено, если алгоритм определен рекурсивно.

Чтобы циклический процесс преобразовать в рекурсивный, нужно уметь определить (выделить) три важных момента:

- условие прекращения последовательности рекурсивных вызовов функции
- формулу следующего элемента или итератора, который используется в рекурсивном процессе
- список параметров, которые передаются в рекурсивную функцию. Один из параметров обязательно есть итератор (счетчик) изменяющий свое значение.

Выполнение работы.

Написание работы производилось на базе операционной системы Windows 10 в среде разработки VisualStudio(где также производились сборка, отладка и тестирование программы).

Программе передается строка (в зависимости от выбора пользователя - либо с консоли, либо из файла). Далее эта строка передается в функцию rec(), являющуюся рекурсивной. Изначально функция проверяет, является ли строка просто буквой, и если это так, то выходит из функции. В противном случае строка должна начинаться со скобки, при наличии которой мы идем дальше, удаляя ее с помощью функции substr() и снова вызывая функцию rec(). Каждый раз, когда мы встречаем скобки, мы увеличиваем соответственный счетчик (opening_bracket или closing_bracket), чтобы в дальнейшем, при последней проверке в main() выявить корректность их наличия.

Далее происходит проверка выражения в скобках. Это может быть буква, символ и открывающая скобка; буква, символ и буква. В этом случае мы посимвольно их удаляем. Если же строка не соответствует, то нам выдает ошибку и программа завершает работу. Так, открывая скобки и поочередно удаляя символы, мы проверяем подходит ли строка под понятие *простое выражение*.

Вывод.

В ходе выполнения лабораторной работы был изучен такой вид программ, как синтаксические анализаторы. Была реализована программа, которая анализирует строку рекурсивным методом, определяя соответствие определению.

Тестовые задания.

Входные данные	**Результат**
((a+b)-(c+b))	((a+b)-(c+b)) Correct
A	A Correct
(a+b))	(a+b)) Invalid brackets
(--)	(--) Error3
(((a+b)-c)*(a-b))	(((a+b)-c)*(a-b)) Correct
((((v+h))))	((((v+h)))) Ooops

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

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

using namespace std;

int opening_bracket = 0;
int closing_bracket = 0;
int simb = 0;
int str_s = 0;

int rec(string& str, ofstream& output)
{

	while (str.size())
	{
		//output << str << endl;
		cout << str << endl;
		cout << opening_bracket << endl;
		cout << closing_bracket << endl;

		if (((str[0] >= 97 && str[0] <= 122) || (str[0] >= 65 && str[0] <=
90)) && (str.size() == 1))
			return 0;

		if (str[0] == '(')
		{
			opening_bracket++;
			str = str.substr(1, str.size() - 1);
			int k = rec(str, output);
			str = str.substr(k, str.size() - k);
			if (str[0] == ')')
				continue;
			if (str[0] == '\0')
				return 0;
			if ((str[0] == '-' || str[0] == '*' || str[0] == '+') &&
((str[1] >= 97 && str[1] <= 122) || str[1] == '(' || (str[0] >= 65 && str[0] <=
90)))
			{
				if (str[1] == '(') {
					opening_bracket++;
					simb++;
				}
				if (str[1] != '(') {
					simb+=2;
				}
				str = str.substr(2, str.size() - 2);
				continue;
			}

			if (str[0] != '*' && str[0] != '+' && str[0] != '-')
			{

				output << "Error1" << endl;
				cout << "Error1" << endl;
				exit(0);
```

```
                  }

            }
            else
            {

                  if (str[0] == ')')
                  {
                        if (str[1] >= 97 && str[1])
                        {
                              output << "Error2" << endl;
                              cout << "Error2" << endl;
                              exit(0);
                        }

                        closing_bracket++;
                        str = str.substr(1, str.size() - 1);

                        if ((str[0] == '-' || str[0] == '*' || str[0] == '+') &&
((str[1] >= 97 && str[1] <= 122) || str[1] == '(' || (str[0] >= 65 && str[0] <=
90)))
                        {
                              if (str[1] == '(') {
                                    opening_bracket++;
                                    simb++;
                              }
                              if (str[1] != '(') {
                                    simb+=2;
                              }
                              str = str.substr(2, str.size() - 2);

                              continue;
                        }
                        continue;
                  }
                  if ((str[0] >= 97 && str[0] <= 122) || (str[0] >= 65 && str[0]
<= 90)) {
                        if (str[1] == '*' || str[1] == '+' || str[1] == '-') {
                              if ((str[2] >= 97 && str[2] <= 122) || (str[2] >= 65
&& str[2] <= 90)) {
                                    simb += 3;
                                    return 3;
                              }
                              else {
                                    str = str.substr(2, str.size() - 2);
                                    simb += 2;
                                    continue;
                              }
                        }
                        else {
                              simb++;
                              return 1;
                        }
                  }
                  output << "Error3" << endl;
                  cout << "Error3" << endl;
                  //output << str << endl;
                  cout << str << endl;
                  exit(0);

            }
      }
```

```
	return 0;
}

int main() {
	{

		ofstream output("/Users/Elizaveta/source/repos/lr1/output.txt");
		string com, str;

		cout << R"(Would u like to get input from "input.txt"? Y/N)" << endl;
		cin >> com;

		if (com[0] == 'Y') {
			cout << R"(Searching for it...)" << endl;
			ifstream input("input.txt", ifstream::in);
			if (input.fail()) {
				cout << "File is not certain." << endl;
				return 0;
			}
			getline(input, str);
		}
		else if (com[0] == 'N') {
			cout << R"(Then type the string!)" << endl;
			cin >> str;
		}
		else {
			cout << "Try once again..." << endl;
			return 0;
		}

		output << str << endl;
		cout << str << endl;
		str_s = str.size();
		rec(str, output);

		if (opening_bracket == closing_bracket) {
			if (simb >= (2 * opening_bracket + 1)) {
				output << "Correct" << endl;
				cout << "Correct" << endl;
			}
			else {
				output << "Ooops" << endl;
				cout << "Ooops" << endl;
			}
		}
		else {
			output << "Invalid brackets" << endl;
			cout << "Invalid brackets" << endl;
		}
		return 0;
	}
}
```