**МИНОБРНАУКИ РОССИИ**
**САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ**
**ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ**
**«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)**
**Кафедра МОЭВМ**

# ОТЧЕТ

**по лабораторной работе №1**

**по дисциплине «Алгоритмы и структуры данных»**

**Тема: Рекурсия**

Студент гр. 8304 ____ Нам Ё Себ

Преподаватель ____ Фирсов М.А.

Санкт-Петербург

2019

**Цель работы.**

Изучить основы рекурсии и составления эффективных алгоритмов.

**Постановка задачи.**

1) Разработать программу, использующую рекурсию;
2) Сопоставить рекурсивное и итеративное решение задачи;
3) Сделать вывод о целесообразности и эффективности рекурсивного подхода для решения данной задачи.

Вариант 16

Построение синтаксического анализатора для понятия *скобки*.

*скобки*::= A | B | (*скобки скобки*)

**Описание алгоритма.**

Для решения поставленной задачи была реализована рекурсивная функция check, которая заменяет каждую подстроку “(x x)” полученной строки на подстроку “x” и рекурсивно вызывает саму себя от измененной строки. Выход из рекурсии осуществляется посредством двух проверок. Первая – проверка на то, что полученная строка равна строке “x” (так как дальнейшая замена бесполезна) и возвращает значение True, вторая – проверка на подстроки (x x) в полученной строке, если на одном из шагов таковой не было найдено, то функция возвращает значение False.

**Спецификация программы.**

Программа предназначена для синтаксического анализа выражения методом рекурсии.

Программа написана на языке C++. Входными данными является либо строка, либо путь до файла содержащего строку (строки). Выходными данными являются промежуточные значения входной после замены описанной выше и глубина рекурсии.

**Описание функций.**

1) Функция CheckInputData.

Объявление функции:

```
bool CheckInputData(std::string inputData);
```

Данная функция осуществляет проверку на отсутствие нешаблонных символов во входной строке. Если такие имеются возвращаемое значение – False, иначе – True. Обход по принятой строке осуществляется при помощи цикла for, проверка на нешаблонные символы осуществляется при помощи условного оператора if.

2) Функция Analyzer.

Объявление функции:

```
bool Analyzer(std::string& _data_, std::ostream& out, int i)
```

Данная функция осуществляет проверку на то, удовлетворяет ли входная строка поставленной задаче. Функция принимает проверяемую строку, ссылку на поток вывода и имеет параметр i, который по умолчанию равен 0, он используется для подсчета глубины рекурсии. Сначала функция проверяет входную строку на равенство строке “x”, если это условие выполнилось, то функция завершается, возвращая True. Далее при помощи метода find класса std::string осуществляется поиск подстроки “( x x ) ” (возвращаемое значение сохраняется в переменную obj), если данная подстрока не была найдена (то есть функция вернула специальный параметр std::string::npos), то функция завершается, возвращая False. Если проверка последняя проверка не выполнилась, то подстрока “( x x )” была найдена и индекс ее первого вхождения в строку _data_ содержится в переменной obj. Далее при помощи метода erase класса std::string из строки _data_ удаляется найденная ранее подстрока “( x x )”. Далее при помощи метода insert класса std::string осуществляется “вставка” подстроки “x” в строку _data_ (места вставки и удаления определяются при помощи ранее найденного индекса). В конце функция рекурсивно вызывает саму себя от уже измененной строки.

**Вывод.**

Был получен опыт работы с рекурсией и с построением синтаксического анализатора. На мой взгляд, итеративное решение поставленной задачи более эффективно.

## ПРИЛОЖЕНИЕ

### 1) ТЕСТИРОВАНИЕ:

Работа программы для строки (A ((A B) (A A))):

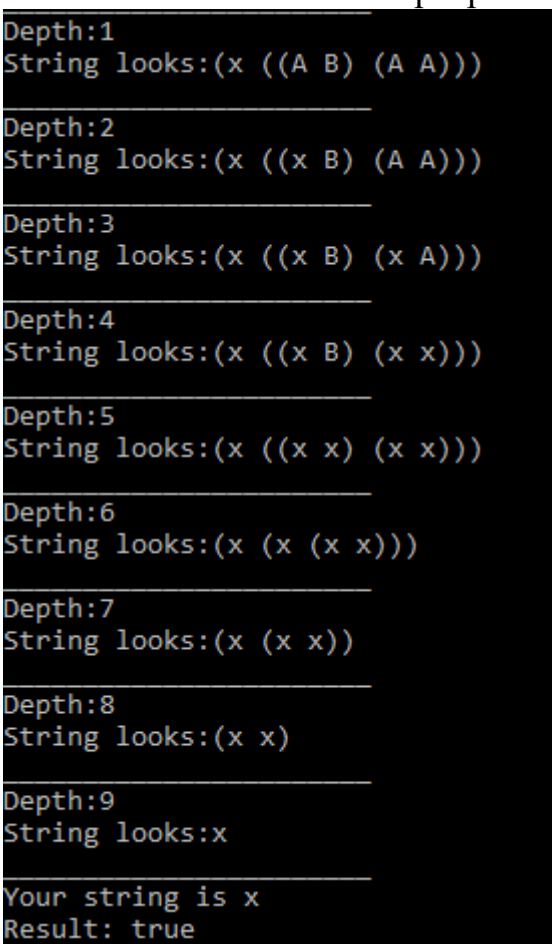


Таблица результатов ввода/вывода тестирования программы:

| Ввод | Вывод |
|---|---|
| (A B) | true |
| ((A C)) | false |
| ))A B(( | false |
| (A (B (A (A (A B))))) | true |
| (A A (B B (A A (B A)))) | false |
| A (B A) B | false |
| (A B (C A) W)) | false |
| (A ((A B) (A A))) | true |
| ((B ((B A) (B B))) (A ((A B) (A A)))) | true |
| (B A (A B) (A B) A B ((A B) A B) A B) A B) | false |

## 2) ИСХОДНЫЙ КОД:

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

// Функция, ведущая диалог с пользователем по поводу формата вывода(Сделана для того, чтобы в случае ошибки, пользователь мог ввести данные заново);
std::string outDialog(int option);

// Функция, ведущая диалог с пользователем по поводу формата ввода(Сделана для того, чтобы в случае ошибки, пользователь мог ввести данные заново);
std::string InDialog(int option);

// Функция, ведущая общий диалог и объединяющая введенные данные из outDialog и InDialog;
void Dialog();

// Функция, проверяющая введенную строку на наличие наших символов;
bool CheckInputData(std::string inputData);

// Функция, выводящая результат, а так же вызывающая функцию Analyzer;
void Result(std::string _data_, std::ostream& out, int i);

// Рекурсивная фунция, обрабатывающая введеную пользователем строку;
bool Analyzer(std::string& _data_, std::ostream& out, int i);

std::string outDialog(int option) {
	std::cout << "\nChoise your output format(Press a number and after press Enter) :\n\n1)Console\n2)Your file(Write a path for your file or write name of file and it will be created in a directory with a code)\n3)Back\nYour choise: ";
	std::string inputData;
	std::cin >> option;
	// Вывод с консоли;
	if (option == 1)
	{
		return "cout";
	}
	// Вывод в файл;
	else if (option == 2)
	{
		std::cout << "\nInput path for your file.txt and press Enter: ";
		std::cin.ignore();
		std::getline(std::cin, inputData);
		size_t pos = inputData.find(".txt");
		if (pos != std::string::npos)
			inputData.append(".txt");
		std::ofstream outFile;
		outFile.open(inputData);
		if (!(outFile.is_open()))
		{
			std::cout << "Incorrect path! Please, try again!\n";
			return outDialog(option);
		}
		return inputData;
	}
	// Вернуться назад, знаю, что это не очень хорошо, но зато работает и есть возможность передумать ;) P.S.: Только много раз не нажимайте на "3";
	else if (option == 3)
	{
		Dialog();
```

```
	}
}

std::string InDialog(int option)
{
	std::string inputData;
	// Вввод с консоли;
	if (option == 1)
	{
		std::cout << "\nInput your string and press Enter: ";
		std::getline(std::cin, inputData);
		if (!CheckInputData(inputData))
		{
			std::cout << "\nIncorrect symbols, please, try again!\n";
			return InDialog(option);
		}
		else
			return inputData;
	}
	// Ввод из файла;
	else if (option == 2)
	{
		std::cout << "\nInput path for your file.txt and press Enter:";
		std::getline(std::cin, inputData);
		std::ifstream inputFile(inputData);
		if (!(inputFile.is_open()))
		{
			std::cout << "Incorrect path! Please, try again!\n";
			return InDialog(option);
		}
		else return inputData;
	}
}

void Dialog()
{
	std::cout << "Choise your input format(Press a number and after press
Enter):\n\n1)Console\n2)Open your file(Write the path to your file)\n3)Exit\n4)For sensei ;)\nYour
choise:";
	int inOption = 0;
	int outOption = 0;
	std::ofstream out;
	std::string inputData;
	std::string outPath;
	std::cin >> inOption;
	// Нужно, чтобы std::getline() работал корректно;
	std::cin.ignore();
	// Ввод с консоли;
	if (inOption == 1)
	{
		inputData = InDialog(1);
		outPath = outDialog(outOption);
		if (outPath == "cout")
		{
			std::ostream& workStream = std::cout;
			Result(inputData, workStream, 0);
		}
		else
		{
			out.open(outPath);
			std::ostream& workStream = out;
			Result(inputData, workStream, 0);
		}
	}
	// Ввод из файла;
```

```
	else if (inOption == 2)
	{
		inputData = InDialog(2);
		std::string dataFile;
		std::ifstream inputFile(inputData);
		outPath = outDialog(outOption);
		if (outPath == "cout")
		{
			std::ostream& workStream = std::cout;
			while (std::getline(inputFile, dataFile))
			{
				if (!CheckInputData(dataFile))
				{
					workStream << std::endl <<
"_____________________________________________________________" << std::endl << "Your
input: " + dataFile << std::endl << "Result: False - incorrect symbols\n" << std::endl <<
"_____________________________________________________________";
					continue;
				}
				Result(dataFile, workStream, 0);
			}
		}
		else
		{
			out.open(outPath);
			std::ostream& workStream = out;
			while (std::getline(inputFile, dataFile))
			{
				if (!CheckInputData(dataFile))
				{
					workStream << std::endl <<
"_____________________________________________________________" << std::endl << "Your
input: " + dataFile << std::endl << "Result: False - incorrect symbols\n" << std::endl <<
"_____________________________________________________________";
					continue;
				}
				Result(dataFile, workStream, 0);
			}
		}

	}
	// Выход из программы;
	else if (inOption == 3)
	{
		std::cout << "Goodbye, CYA later!";
		return;
	}
	// В случае неккоректной выбранной опции;
	else
	{
		std::cout << "\nPlease press not like a dunmb, only four numbers're here -_-\n";
		Dialog();
	}

	char quest;
	std::cout << "Mission complete!" << std::endl;
	std::cout << std::endl << "Do you want to retry?(y/n)" << std::endl;
	std::cin >> quest;
	if (quest == 'y')
		Dialog();
	else
		std::cout << std::endl << "Goodbye!=)";
}

bool CheckInputData(std::string inputData)
```

```
{
	for (int i = 0; i < inputData.size(); i++)
		if (inputData[i] != 'A' && inputData[i] != 'B' && inputData[i] != '(' && inputData[i]
!= ')' && inputData[i] != ' ')
			return 0;
	return 1;
}

void Result(std::string _data_, std::ostream& out, int i)
{
	out << std::endl <<
"_____________________________________________________________________" << std::endl;
	out << std::endl << "Your string is " + _data_ + "\nResult: " << std::boolalpha <<
Analyzer(_data_, out, i);
	out << std::endl <<
"_____________________________________________________________________" << std::endl;
}

bool Analyzer(std::string& _data_, std::ostream& out, int i)
{
	out << std::endl << "Depth:" << i << std::endl << "String looks:" << _data_ << std::endl <<
"_____________________";
	if (_data_ == "x")
		std::cout << "qwer";
		return 1;
	std::size_t objA = _data_.find("A");

	if (objA != std::string::npos)
	{
		_data_.erase(objA + _data_.begin(), objA + _data_.begin() + 1);
		_data_.insert(objA + _data_.begin(), 'x');
	}
	std::size_t objB = _data_.find("B");
	else if (objB != std::string::npos)
	{
		_data_.erase(objB + _data_.begin(), objB + _data_.begin() + 1);
		_data_.insert(objB + _data_.begin(), 'x');
	}
	else
	{
		size_t obj = _data_.find("(x x)");
		if (obj == std::string::npos)
			return 0;
		_data_.erase(obj + _data_.begin(), obj + _data_.begin() + 5);
		_data_.insert(obj + _data_.begin(), 'x');
	}
	return Analyzer(_data_, out, ++i);
}

int main()
{
	std::cout << "\tHELLO! THIS IS SUPER SYNTAX ANALYZER!" << "THIS IS PROGRAMM FOR ANALYZE YOUR
STRING IF IT IS x::= A|B|(xx)\n\n";
	Dialog();
	return 0;
   }
```