

МИНОБРНАУКИ РОССИИ
САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ
ЭЛЕКТРОТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ
«ЛЭТИ» ИМ. В.И. УЛЬЯНОВА (ЛЕНИНА)
Кафедра математического обеспечения и применения ЭВМ

ОТЧЕТ
по лабораторной работе №1
по дисциплине «Алгоритмы и структуры данных»
Тема: Рекурсия

Студент гр. 9304

Силкин В.А.

Преподаватель

Фиалковский М.С.

Санкт-Петербург

2020

Цель работы.

Научиться использовать рекурсивный подход решения задачи на языке C++.

Задание.

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.

В ходе выполнения практической работы необходимо использовать GNU Octave – программу для произведения математических расчётов и построения графиков. В ней также присутствует широкий спектр разнообразных функций для работы с матрицами, которые будут использованы в этой работе.

Выполнение работы.

Для выполнения задачи используются функции для определения простого выражения, идентификатора, знака операции и конкретно 2-го вида простого выражения. Для всего этого служат функции `is_expression`, `is_iden`, `is_sign` и `is_bracketEx` соответственно. `is_expression` имеет перегрузку для рекурсии.

Изначально в `is_expression` подаётся строка, которую надо проверить, является ли она простым выражением. Там проверяются оба случая простого выражения. В первом случае вызывается лишь `is_iden`, а во втором `is_bracketEx`, которое в свою очередь дважды вызывает перегрузку `is_expression`, и один раз `is_sign`. Таким образом получается, что `is_expression` и `is_bracketEx` входят во взаимную рекурсию, пока `is_expression` не вызовет `is_iden`, который в свою очередь не является рекурсивным.

Для обхода по строке используется итератор стандартной библиотеки. Если подаётся пустая строка, её отлавливает функция `is_bracketEx`. Если в выражении присутствуют пробелы, функция видит его как неправильное. Функция обрабатывает только первое выражение в строке, и выводит специальное сообщение, если символов больше чем включается в это выражение, но только если оно написано верно.

Разработанный программный код см. в приложении А.

Тестирование.

Компиляция выполняется командой `make` (или `make lab`). Для тестирования написан `bash`-скрипт, лежащий в корневой папке лабораторной работы, он запускается через `make testing`, либо вручную через файл `test_script`. Если до этого компиляции не было, программа сама скомпилирует файл для тестов.

Выводы.

Была проведена работа по ознакомлению с рекурсией, и выполнению поставленных задач с её помощью на основе языка программирования C++.

ПРИЛОЖЕНИЕ А

ИСХОДНЫЙ КОД ПРОГРАММЫ

Название файла: main.cpp

```
#include <iostream>
#include <cctype>
#include <string>

std::string is_expression(std::string);
bool is_iden(const std::string::iterator);
bool is_sign(const std::string::iterator);
bool is_bracketEx(std::string::iterator&);
bool is_expression(std::string::iterator&);

std::string is_expression(std::string in) {
    auto iter = in.begin();
    if(is_iden(iter) || is_bracketEx(iter)) {
        iter++;
        if(*iter) {
            return "There is more symbols than one expression\n";
        }
        return "One expression detected\n";
    }
    return "There is no expressions\n";
}

bool is_expression(std::string::iterator &iter) {
    if(is_iden(iter) || is_bracketEx(iter)) {
        return true;
    }
    return false;
}

bool is_bracketEx(std::string::iterator &iter)
{
    if(!(*iter)) {
        return false;
    }
    if(*iter != '(') {
        return false;
    }
    iter++;
    if(!(is_expression(iter))) {
        return false;
    }
    iter++;
    if(!(is_sign(iter))) {
        return false;
    }
}
```

```

    }
    iter++;
    while(isspace(*iter)) iter++;
    if(!is_expression(iter)) {
        return false;
    }
    iter++;
    if(*iter != ')') {
        return false;
    }
    return true;
}

bool is_iden (const std::string::iterator iter) {
    return std::isalpha(*iter);
}

bool is_sign (const std::string::iterator iter) {
    std::string signs = "-+*";
    for(auto it = signs.begin(); it != signs.end(); it++) {
        if(*it == *iter) {
            return true;
        }
    }
    return false;
}

int main() {
    std::string in;
    std::cout << "Enter an expression\n";
    std::cin >> in;
    std::cout << is_expression(in);
    return 0;
}

```

ПРИЛОЖЕНИЕ В

РЕЗУЛЬТАТЫ ТЕСТИРОВАНИЯ

№	Входные данные	Выходные данные	Комментарии
1	$((M-a)+(n*u))$	Enter an expression One expression detected	3 раза вызывается is_bracketEx
2	n	Enter an expression One expression detected	Простой вызов is_iden
3	(((((((Enter an expression There is no expressions	После открытой скобки не идёт простое выражение
4	$(a*b)n$	Enter an expression There is more symbols than one expression	Выражение $(a*b)$ верное, но на вход подаются и следующие символы