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

Студент гр. 9304 Шуняев А.В.

Преподаватель Фиалковский М.С.

Санкт-Петербург
2020

Цель работы.

Ознакомиться с основными понятиями и приёмами рекурсивного программирования, получить навыки программирования рекурсивных процедур и функций на языке программирования C++.

Задание.

Требования и рекомендации к выполнению задания:

1. проанализировать полученное задание, выделив рекурсивно определяемые информационные объекты и (или) действия;
2. разработать программу, использующую рекурсию;
3. сопоставить рекурсивное решение с итеративным решением задачи;
4. сделать вывод о целесообразности и эффективности рекурсивного решения данной задачи.

13. Построить синтаксический анализатор для понятия скобки.

скобки::=A | скобка скобки

скобка::= (B скобки)

Выполнение работы.

Описание алгоритма работы программы:

На вход программы подаются строки неограниченной длинны. Данные строки записываются в список строк. После этого вызывается рекурсивный метод, который обрабатывает строки и возвращает булевое значение. Сама обработка заключается в проверки текущего символа и следующего за ним. Если следующий символ является одним из возможных для текущего, то обработка продолжается, иначе возвращается false. Данное булевое значение определяет ответ, который будет записан в файл вывода.

Формат входных и выходных данных:

Входные данные представлены в виде строк, каждый символ которых может быть одним из символов ‘(‘, ‘)’, ‘A’, ‘B’. Они считываются из файла input.txt. Выходными данные являются те же строки, плюс строки-индикаторы “ – THIS IS A BRACKETS” и “ – THIS IS NOT A BRACKETS”, которые указывают является ли строка скобками или нет. Выходные данные записываются в файлы output.txt.

Описание основных структур данных и функций (кратко):

Реализованы 2 класса Data и Recursion. Класс Data хранит в себе список строк, полученных на вход. Также в нем реализован статический метод StartDataProcessing, который запускается из функции main. Данный метод вызывает все необходимые методы для выполнения алгоритма, создает объекты классов и выполняет запись результат. Также в данном классе реализован метод SpaceErase, который удаляет все пробелы в строке.

В классе Recursion реализована основная логика алгоритма. В нем реализован рекурсивный метод IsBrackets. Данный метод обрабатывает строки посимвольно.

Тестирование:

Входные данные	Выходные данные	Результат теста
A	A - THIS IS A BRACKETS	True!
(B A) A	(B A) A - THIS IS A BRACKETS	True!
(B (B A) A) A	(B (B A) A) A - THIS IS A BRACKETS	True!
AA	AA - THIS IS NOT A BRACKETS	True!
((B A) A	((B A) A - THIS IS NOT A BRACKETS	True!
(BBA)A	(BBA)A - THIS IS NOT A BRACKETS	True!
B A) A	B A) A - THIS IS NOT A BRACKETS	True!
(B (B A) (B A) A) A	(B (B A) (B A) A) A - THIS IS A BRACKETS	True!

(B (B (B A) A) A) A	(B (B (B A) A) A) A - THIS IS A BRACKETS	True!

Выводы.

Сопоставляя рекурсивное решение с итеративным, очевидно, что данная задача легко решается без рекурсии. Я считаю, применение рекурсии в решение является целесообразным только в целях обучения т.к. многократный вызов рекурсивной функции требует больше ресурсов чем цикл т.к. каждый вызов функции складывает на стек все параметры функции, локальные переменные и т.д. При большой глубине рекурсии может произойти переполнение стека.