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(ка) гр. 0000 _____ Ахримов А.М.

Преподаватель _____ Филатов А.Ю.

Санкт-Петербург
2020

Цель работы.

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

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.

Задание

3. Имеется n городов, пронумерованных от 1 до n. Некоторые пары городов соединены дорогами. Определить, можно ли попасть по этим дорогам из одного заданного города в другой заданный город. Входная информация о дорогах задаётся в виде последовательности пар чисел i и j ($i < j$ и $i, j \in 1..n$), указывающих, что i-й и j-й города соединены дорогами.

Выполнение работы.

Для решения данной задачи была применена теория графов. Города представляют собой вершины графа, а дороги – рёбра. Для поиска пути применяется алгоритм поиска “в глубину“:

1. Начинаем с заданного города, помечаем его. Переходим к шагу 2.

2. Смотрим, есть ли ещё не отмеченный город, соединенный ребром с текущим. Если таковой найдётся, переходим в него и возвращаемся к шагу 1. Если такого города нет, завершаем работу. И если это искомый город, завершаем работу.

Из алгоритма видно, что функция, написанная на его основе, прямо рекурсивна. Функция принимает в качестве параметров номера городов, которые проверяются на наличие между ними дорог. Возвращаемое значение - объект логического типа bool (true – между городами существует дорога, false – дороги между городами нет).

Для данной программы были разработаны два класса: City и ArrayOfCity. В классе City есть поля с номером города, со списком дорог в другие города и поле для “отметки” города. Также в нем определен метод для проверки существования дороги к конкретному городу checkRoad. В классе ArrayOfCity основное поле – массив из объектов класса City. Данный класс выполняет ключевую роль – один из его методов checkRoads и есть рекурсивная функция, ищущая путь из одного города в другой.

Информация о количестве городов и дорогах между ними записывается в программу из файла, который нужно указать при запуске программы. Далее пользователь может сколь угодно задавать города для проверки пути между ними.

Тестирование

(numberdata – количество дорог)

№	Основные входные данные	Данные, вводимые пользователем	Результат программы
1.1	8 (n) 8(numberdata) 1 2 1 3	2 4	2 Start function checkRoads(2, 5) 1 Start function checkRoads(1, 5) 3 Start function checkRoads(3, 5) 3 Finish function checkRoads(3, 5) 1 Finish function checkRoads(1, 5) 2 Finish function checkRoads(2, 5)

	2 3		Road exist.
1.2	3 4 3 5 6 7 7 8 8 6	3 8	3 Start function checkRoads(3, 8) 1 Start function checkRoads(1, 8) 2 Start function checkRoads(2, 8) 2 Finish function checkRoads(2, 8) 1 Finish function checkRoads(1, 8) 4 Start function checkRoads(4, 8) 4 Finish function checkRoads(4, 8) 5 Start function checkRoads(5, 8) 5 Finish function checkRoads(5, 8) 3 Finish function checkRoads(3, 8) Road don't exist.
1.3		1 5	1 Start function checkRoads(1, 5) 2 Start function checkRoads(2, 5) 3 Start function checkRoads(3, 5) 3 Finish function checkRoads(3, 5) 2 Finish function checkRoads(2, 5) 1 Finish function checkRoads(1, 5) Road exist.
2.1	9(n) 8(numberdata) 1 2 2 3 3 4	1 4	1 Start function checkRoads(1, 4) 2 Start function checkRoads(2, 4) 3 Start function checkRoads(3, 4) 3 Finish function checkRoads(3, 4) 2 Finish function checkRoads(2, 4) 1 Finish function checkRoads(1, 4) Road exist.
2.2	5 6 5 9 6 7 7 8 8 9	2 7	2 Start function checkRoads(2, 7) 1 Start function checkRoads(1, 7) 1 Finish function checkRoads(1, 7) 3 Start function checkRoads(3, 7) 4 Start function checkRoads(4, 7) 4 Finish function checkRoads(4, 7) 3 Finish function checkRoads(3, 7) 2 Finish function checkRoads(2, 7) Road don't exist.

Выводы.

В ходе выполнения работы были приобретены навыки по работе с рекурсивными функциями. Были изучены базовые понятия о рекурсии и рекурсивном программировании.