**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. 6304 |  | Виноградов К.А. |
| Преподаватель |  | Балтрашевич В.Э. |

Санкт-Петербург

2017

**Цель работы.**

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

**Текст задания**

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∈1..n), указывающих, что i-й и j-й города соединены дорогами.

**Ход работы**

Сначала написан код, выполняющийся открытие отдельного окна выбора файла по нажатию вкладки «Open file», созданной с помощью встроенного в Qt редактора. В данном окне из специального поля «treeView», которое также создано с помощью редактора, считывается путь к файлу с исходными данными. Если по указанному пути файл не найден, то выводится соответствующее сообщение об ошибке. Если же файл найден, то эта строка преобразуется в формат QString для того, чтобы ее можно было использовать для открытия файла. Код данной функции представлен в приложении А.

Из считанного из файла текста формируется список элементов, которые преобразуются в объекты класса int и подаются в функцию поиска дорог графа graph. В функции graph элементы проходят обработку, а именно сравниваются с задаваемой точкой начала движения, которая также передается в функцию graph, и если какой либо из них эквивалентен этой точке, то рекурсивно вызывается функция graph, для которой стартовой точкой является уже пара эквивалентного элемента. Все пройденные точки сохраняются в отдельном массиве, который после выполнения функции graph передается обратно в main, где его содержимое сравнивается с требуемой точкой завершения пути в графе. Также для выделения памяти используется рекурсивная функция roads. Код вышеназванных функций представлен в приложении Б.

На основании результата работы этой пары функций в поле «textEdit» выводится соответствующее сообщение, после чего файл закрывается и программа ожидает повторного запроса или завершения.

Далее протестируем программу. Для этого выберем файл по пути «C:Qt\input.txt» и будем, изменяя содержимое файла, отслеживать результат.

Таблица 1. Результаты тестирования программы

|  |  |
| --- | --- |
| Содержимое input.txt | Результат в outputEdit |
| 5 1 2 2 3 3 4 4 5 5 6 1 6 | You entered wrong coordinates |
| 5 1 2 1 3 1 4 1 5 3 2 | There is a road between these towns |
| 5 1 2 2 3 3 4 4 5 1 5 | There is a road between these towns |
| 5 1 2 2 3 3 4 4 5 5 6 6 7 7 8 8 9 9 1 11 1 5 | Error, there is a problem with input data |
| 0 | Error, there is a problem with input data |

**Вывод.**

В ходе данной работы был с помощью рекурсивных функций реализован логический анализатор для понятия скобки, написана программа на языке C++ с использованием фреймворка Qt для создания простейшего интерфейса.

**ПРИЛОЖЕНИЕ А**

**СОДЕРЖАНИЕ MAINWINDOW::READTOFILE**

QString MainWindow::ReadToFile(const QString &FilePath)

{

QFile input(FilePath);

if(!input.*open*(QFile::ReadOnly|QFile::Text))

{

QMessageBox::information(this, "Error", "Path is not correct!");

return buff;

}

QTextStream stream(&input);

int p=input.*size*();

if(p==0)

{

input.*close*();

return buff;

}

buff=stream.readAll();

input.*close*();

flag=1;

return buff;

}

**ПРИЛОЖЕНИЕ Б**

**СОДЕРЖАНИЕ ФУНКЦИЙ GRAPH И ROADS**

int roads(int n)

{

if(n>0)

{

n=(n-1)+roads(n-1);

}

return n;

}

int\* graph(int k, int\* roads1, int\* roads2, int\* result, int j, int y, int towns)

{

int t=0;

for(int i=0; i<towns; i++)

{

if((roads1[i]==k)&&(y!=roads2[i]))

{

y=roads1[i];

t=roads2[i];

while((result[j]!=0)&&(result[j]<towns))

j++;

result[j]=t;

j++;

graph(t, roads1, roads2, result, j, y, towns);

}

else if((roads2[i]==k)&&(y!=roads1[i]))

{

y=roads2[i];

t=roads1[i];

while((result[j]!=0)&&(result[j]<towns))

j++;

result[j]=t;

j++;

graph(t, roads1, roads2, result, j, y, towns);

}

}

return result;

}