dfs.cpp

```cpp
/**
 * depth-first search
 * @author Jon Beck
 * @version 23 March 2021
 */
#include <iostream>
#include <list>
#include <vector>
#include "graph.h"
using namespace std;

/**
 * the dfs algorithm proper
 * @param g the adjacency lists graph
 * @param start_vertex the starting vertex
 * @param visited the list of already-visited vertices
 */
void dfs(const Graph& g, size_t start_vertex, vector<bool>& visited);

/**
 * find all connected componenets and run dfs on each one
 * @param g the graph to run dfs on
 */
void explore_connected_components(const Graph& g);

/**
 * previsit: for now, just report
 * @param vertex the vertex we are previsiting
 */
void previsit(size_t vertex);

/**
 * postvisit: for now, just report
 * @param vertex the vertex we are postvisiting
 */
void postvisit(size_t vertex);

/**
 * utility function to provide a "global" counter, incremented
 * each time the function is called
 * @return the next counter value
 */
size_t get_clock();

int main()
{
  // declare a graph and read it in from standard input
  Graph g;
  g.read_graph();

  g.dump(); // just so we can see the contents of the graph

  explore_connected_components(g);
  return 0;
}
//
```

dfs.cpp

```
57  void dfs(const Graph& g, size_t start_vertex, vector<bool>& visited)
58  {
59    // set this vertex as visited, and get its adjacency list
60    visited.at(start_vertex) = true;
61    previsit(start_vertex);
62
63    list<size_t> list_of_adjacent_vertices {g.get_list(start_vertex)};
64
65    // go through start_vertex's adjacency list, exploring each unvisited
66    // vertex one by one
67    for (auto vertex : list_of_adjacent_vertices)
68    {
69      if (!visited.at(vertex))
70      {
71        dfs(g, vertex, visited);
72      }
73    }
74    postvisit(start_vertex);
75  }
76
77  void explore_connected_components(const Graph& g)
78  {
79    cout << "starting the graph";
80    vector<bool> visited(g.size(), false);
81
82    size_t connected_component_counter {0};
83
84    for (size_t vertex = 0; vertex < g.size(); vertex++)
85    {
86      if (!visited.at(vertex))
87      {
88        cout << endl << "Connected component: " <<
89          connected_component_counter << endl;
90        connected_component_counter++;
91        dfs(g, vertex, visited);
92      }
93    }
94    cout << endl << "finished the graph" << endl;
95  }
96
97  void previsit(size_t vertex)
98  {
99    cout << '+'<< vertex << ' ' << to_string(get_clock()) << ' ';
100 }
101
102 void postvisit(size_t vertex)
103 {
104   cout << '-'<< vertex << ' ' << to_string(get_clock()) << ' ';
105 }
106
107 size_t get_clock()
108 {
109   static size_t clock = 0;
110   return clock++;
111 }
```