```cpp
#include "graph.h"
#include <algorithm>
#include <array>
#include <cassert>
#include <fstream>
#include <iostream>
#include <stdexcept>
#include <string>
#include <vector>
using namespace std;

void read_edges_from_file(const string &file_name, vector<array<int, 2>> &v)
{
    ifstream fin(file_name);           // Oeffne das File im ASCII-Modus
    if ( fin.is_open() ) {             // File gefunden:
        v.clear();                     // Vektor leeren
        int k,l;
        while ( fin >> k >> l) {v.push_back({k,l});} // Einlesen
        if (!fin.eof()) {
            // Fehlerbehandlung
            cout << " Error handling \n";
            if ( fin.bad() )  {throw runtime_error("Schwerer Fehler in istr");}
            if ( fin.fail() ) { // Versuch des Aufraeumens
                cout << " Failed in reading all data.\n";
                fin.clear();
            }
        }
        v.shrink_to_fit();
    }
    else {                             // File nicht gefunden:
        cout << "\nFile " << file_name << " has not been found.\n\n" ;
        assert( fin.is_open() && "File not found." );       // exeption handling for the poor programmer
    }
    //return;
}


vector<vector<int>> get_node2nodes(vector<array<int,2>> const & edges)
{
    // we assume that the nodes are continuously numbered from 0 to n-1
    // determine number of nodes
    int nnode=-1;
    for (size_t k=0; k<edges.size(); ++k)
    {
        for (size_t j=0; j<edges[k].size(); ++j)
        {
            nnode=max(nnode,edges[k][j]);
        }
    }
    ++nnode;                           // number of nodes

    // Determine the neighborhood for each vertex
    vector<vector<int>> n2n(nnode);
    for (size_t k=0; k<edges.size(); ++k)
    {
        const int v0 = edges[k][0];
        const int v1 = edges[k][1];
        n2n[v0].push_back(v1);         // add v1 to neighborhood of v0
        n2n[v1].push_back(v0);         //     and vice versa
    }
    // ascending sort of entries per node
    for (size_t k=0; k<n2n.size(); ++k)
    {
        sort(n2n[k].begin(),n2n[k].end());
    }


    return n2n;
}


```

```
 1: #pragma once                 // substitutes header guarding
 2:
 3: #include <array>
 4: #include <string>
 5: #include <vector>
 6:
 7: /**
 8:   This function opens the ASCII-file named @p file_name and reads the
 9:   int data into the C++ vector @p v.
10:   If the file  @p file_name does not exist then the code stops with an appropriate message.
11:   @param[in]    file_name   name of the ASCII-file
12:   @param[out]   v           C++ vector with edge vertices
13: */
14:
15: void read_edges_from_file(const std::string& file_name, std::vector<std::array<int,2>>& v);
16:
17:
18: /**
19:   Determines the neighboring vertices for each node from the edge definition @p edges .
20:   The node itself is not contained in the neighboring vertices.
21:
22:   @param[in]    edges   vector[ne][2] with edge vertices
23:   @return       vector[nn][*] with all neighboring vertices for each node
24:
25:   @warnung We assume that the nodes are continuously numbered from 0 to nn-1.
26:            Otherwise we have to use map< int, vector<int> > as data structure.
27: */
28: std::vector<std::vector<int>> get_node2nodes(std::vector<std::array<int,2>> const & edges);
29:
30:
```

```cpp
//graph
#include "graph.h"
#include <array>
#include <iostream>
#include <string>
#include <vector>
using namespace std;

int main()
{
    cout << "Hello Graph!" << endl;
    const string name{"g_1.txt"};

    // read the edges
    vector<array<int,2>> edges;
    read_edges_from_file(name, edges);

    cout << "\n --  Edges  --\n";
    for (size_t k=0; k<edges.size(); ++k)
    {
        cout << k << " : ";
        for (size_t j=0; j<edges[k].size(); ++j)
        {
            cout << edges[k][j] << " ";
        }
        cout << endl;
    }

    // construct mapping nodes to nodes
    auto n2n=get_node2nodes(edges);

    cout << "\n --  Nodes to Node  --\n";
    for (size_t k=0; k<n2n.size(); ++k)
    {
        cout << k << " : ";
        for (size_t j=0; j<n2n[k].size(); ++j)
        {
            cout << n2n[k].at(j) << " ";
        }
        cout << endl;
    }


    return 0;
}
```