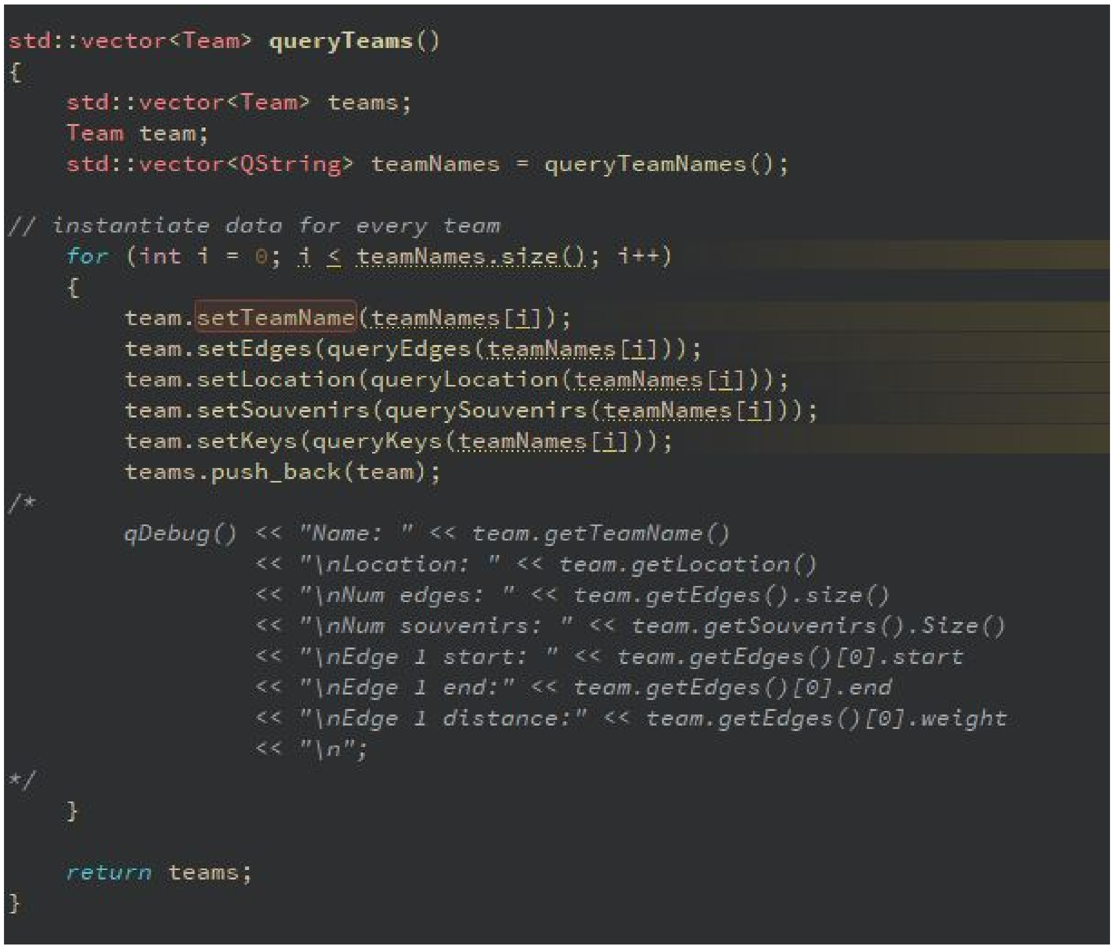
Data Structures Used

Vector:

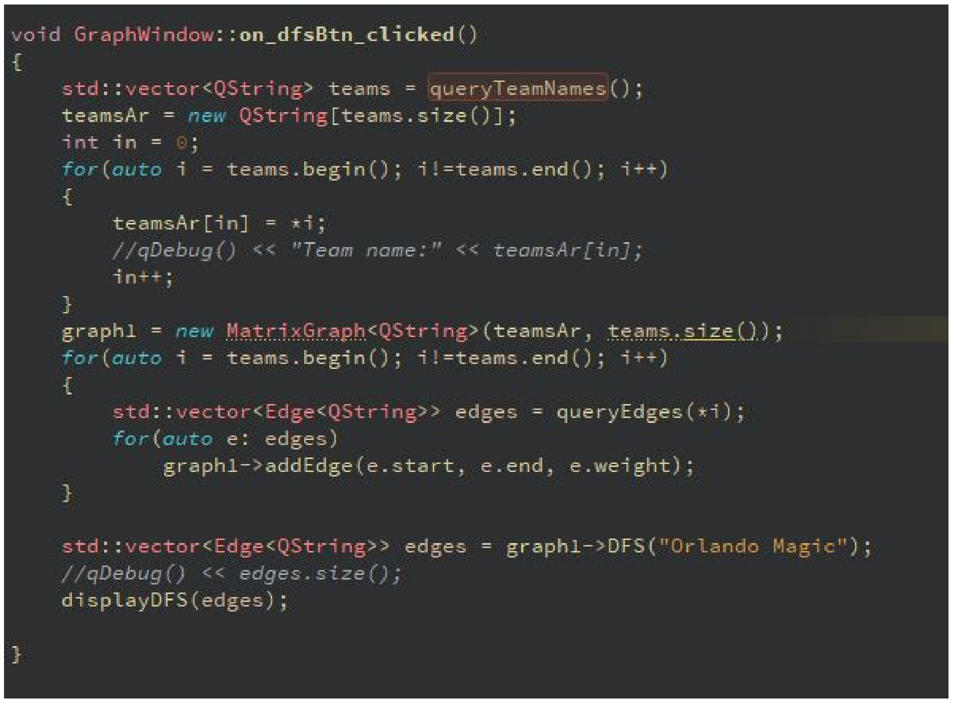
Found in queryTeams function - database.cpp - line 55



```
std::vector<Team> queryTeams()
{
    std::vector<Team> teams;
    Team team;
    std::vector<QString> teamNames = queryTeamNames();

// instantiate data for every team
    for (int i = 0; i < teamNames.size(); i++)
    {
        team.setTeamName(teamNames[i]);
        team.setEdges(queryEdges(teamNames[i]));
        team.setLocation(queryLocation(teamNames[i]));
        team.setSouvenirs(querySouvenirs(teamNames[i]));
        team.setKeys(queryKeys(teamNames[i]));
        teams.push_back(team);
/*
        qDebug() << "Name: " << team.getTeamName()
                 << "\nLocation: " << team.getLocation()
                 << "\nNum edges: " << team.getEdges().size()
                 << "\nNum souvenirs: " << team.getSouvenirs().Size()
                 << "\nEdge 1 start: " << team.getEdges()[0].start
                 << "\nEdge 1 end:" << team.getEdges()[0].end
                 << "\nEdge 1 distance:" << team.getEdges()[0].weight
                 << "\n";
*/
    }

    return teams;
}
```

Graph (Using Adjacency matrix):

Found in graphwindow.cpp - line 60

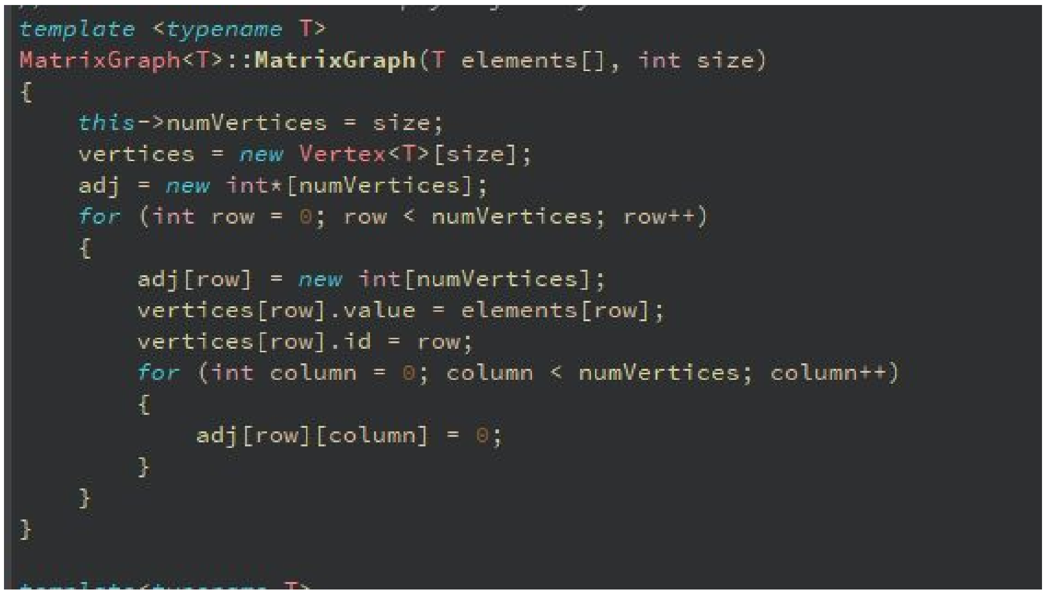


```
void GraphWindow::on_dfsBtn_clicked()
{
    std::vector<QString> teams = queryTeamNames();
    teamsAr = new QString[teams.size()];
    int in = 0;
    for(auto i = teams.begin(); i!=teams.end(); i++)
    {
        teamsAr[in] = *i;
        //qDebug() << "Team name:" << teamsAr[in];
        in++;
    }
    graph1 = new MatrixGraph<QString>(teamsAr, teams.size());
    for(auto i = teams.begin(); i!=teams.end(); i++)
    {
        std::vector<Edge<QString>> edges = queryEdges(*i);
        for(auto e: edges)
            graph1->addEdge(e.start, e.end, e.weight);
    }

    std::vector<Edge<QString>> edges = graph1->DFS("Orlando Magic");
    //qDebug() << edges.size();
    displayDFS(edges);

}
```

Array:

Dynamic 2 dimensional array Found on matrixgraph.h - line 166

Used to create adjacency matrix

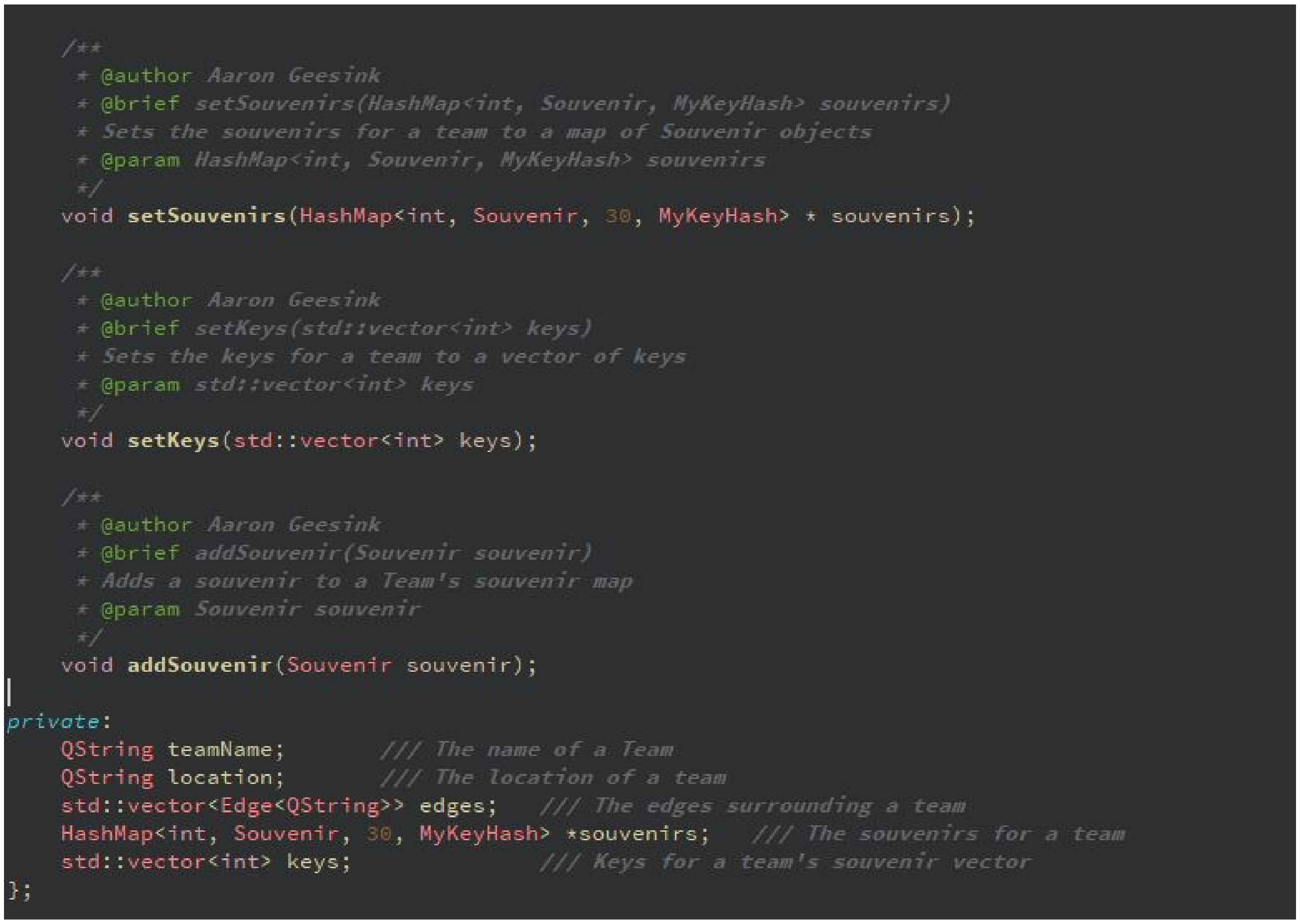


```
template <typename T>
MatrixGraph<T>::MatrixGraph(T elements[], int size)
{
    this->numVertices = size;
    vertices = new Vertex<T>[size];
    adj = new int*[numVertices];
    for (int row = 0; row < numVertices; row++)
    {
        adj[row] = new int[numVertices];
        vertices[row].value = elements[row];
        vertices[row].id = row;
        for (int column = 0; column < numVertices; column++)
        {
            adj[row][column] = 0;
        }
    }
}
```

Map (using hashing function):

Private data member to store souvenirs within the Team class

Found in team.h - line 127



```
    /**
     * @author Aaron Geesink
     * @brief setSouvenirs(HashMap<int, Souvenir, MyKeyHash> souvenirs)
     * Sets the souvenirs for a team to a map of Souvenir objects
     * @param HashMap<int, Souvenir, MyKeyHash> souvenirs
     */
    void setSouvenirs(HashMap<int, Souvenir, 30, MyKeyHash> * souvenirs);

    /**
     * @author Aaron Geesink
     * @brief setKeys(std::vector<int> keys)
     * Sets the keys for a team to a vector of keys
     * @param std::vector<int> keys
     */
    void setKeys(std::vector<int> keys);

    /**
     * @author Aaron Geesink
     * @brief addSouvenir(Souvenir souvenir)
     * Adds a souvenir to a Team's souvenir map
     * @param Souvenir souvenir
     */
    void addSouvenir(Souvenir souvenir);

private:
    QString teamName;       /// The name of a Team
    QString location;       /// The location of a team
    std::vector<Edge<QString>> edges;   /// The edges surrounding a team
    HashMap<int, Souvenir, 30, MyKeyHash> *souvenirs;   /// The souvenirs for a team
    std::vector<int> keys;              /// Keys for a team's souvenir vector
};
```