WATER SUPPLY MANAGEMENT

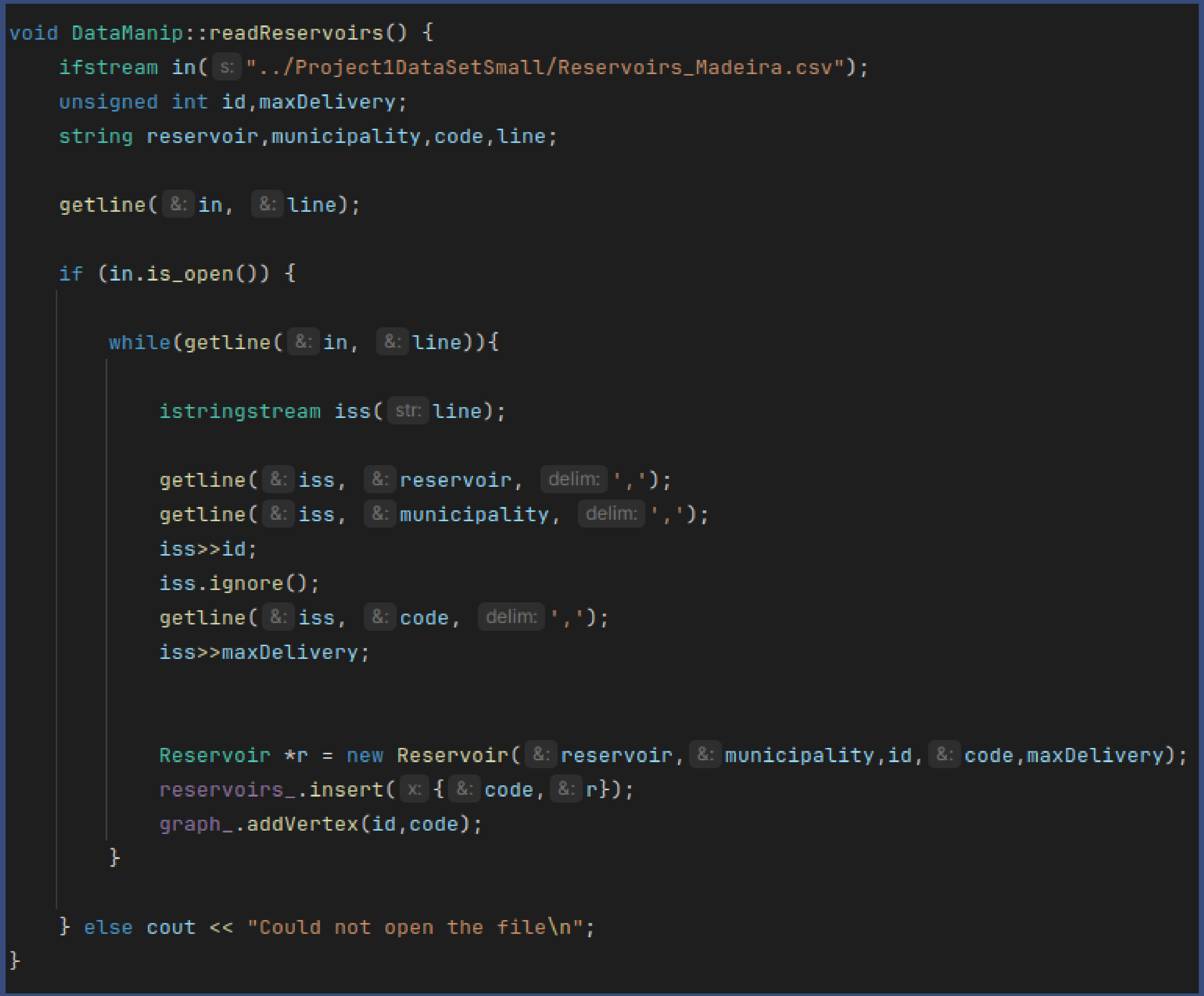
Presented By

- Ana Sofia Baptista - up202207334
- Eduardo Santos - up202207521
- Pedro Pedro - up202206961

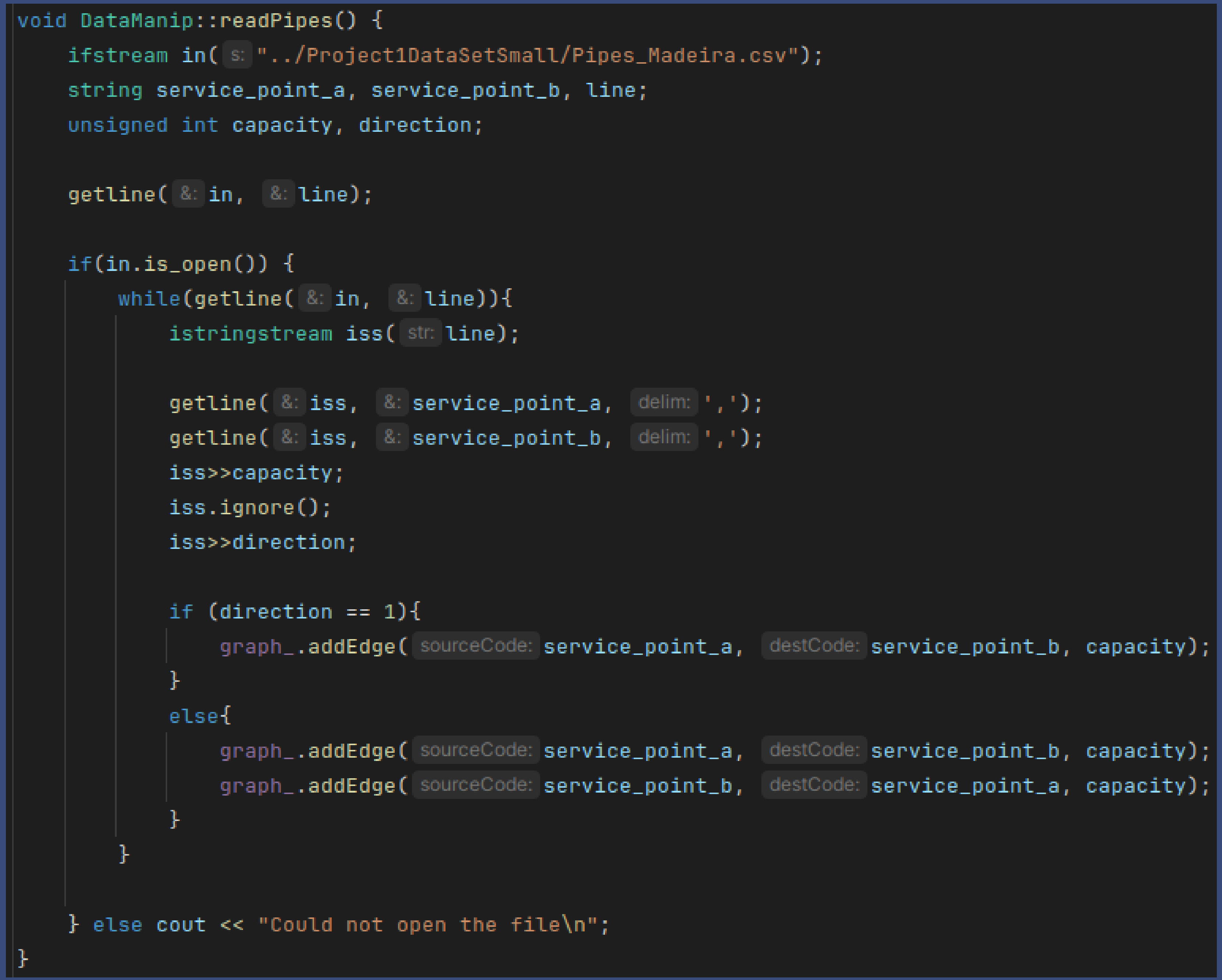
CLASSES

- Reservoir - code, name, municipality, id, max delivery
- Station - code, id
- City - name, code, id, population, demand, flow
- Graph - (class Vertex - lista de Edges, lista de incomings, id, code, boolenano visited, booleano processing, unsigned indegree, Edge path), (class Edge - destino, origem, capacidade, flow)
- DataManip - map de reservoirs, map de stations, map de cities code, map de cities name
- Menu - com um Data, onde são criados vários menus para utilização do user

LEITURA DATASET

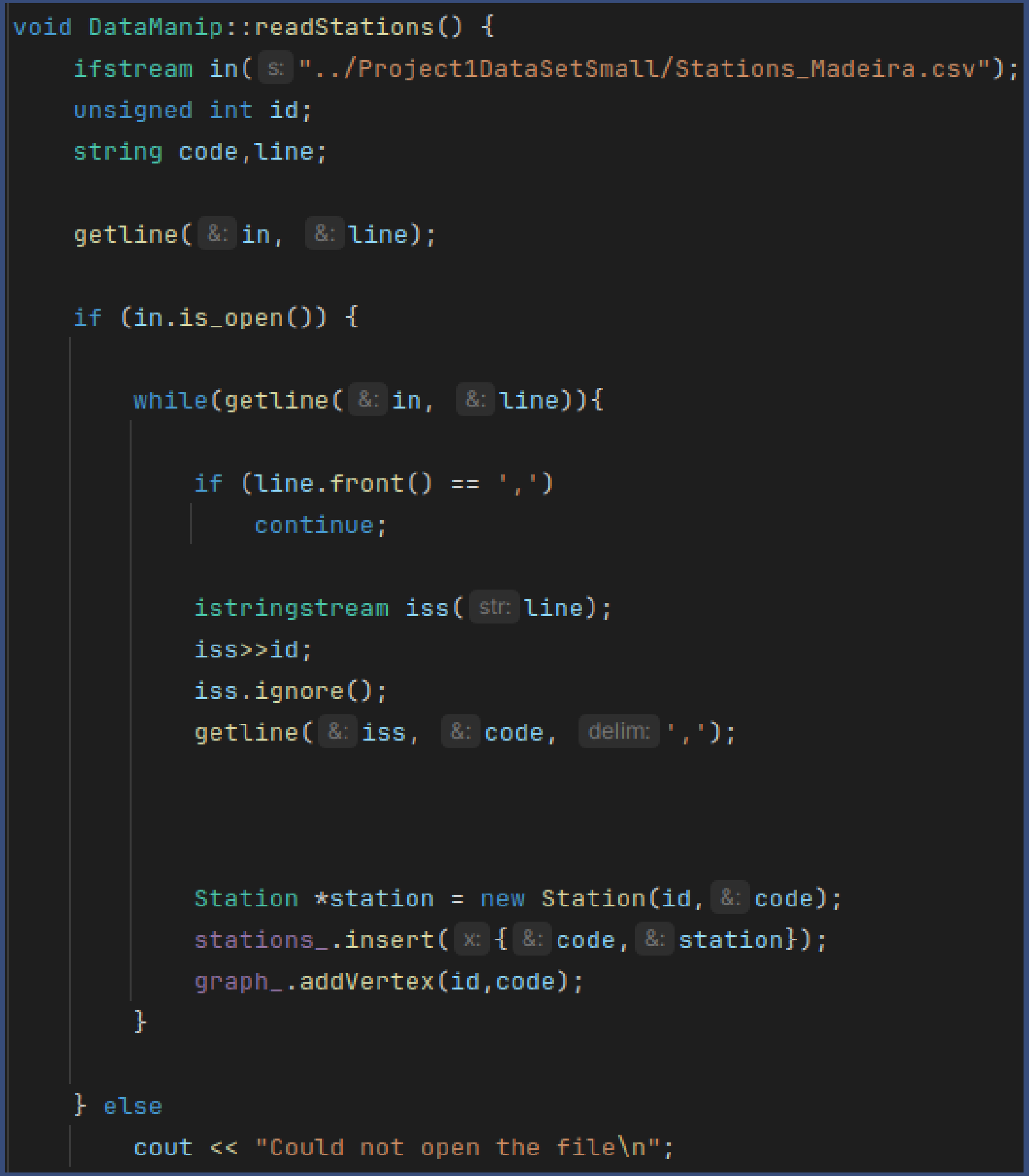


```
void DataManip::readReservoirs() {
    ifstream in( s: "../Project1DataSetSmall/Reservoirs_Madeira.csv");
    unsigned int id,maxDelivery;
    string reservoir,municipality,code,line;

    getline( &: in,  &: line);

    if (in.is_open()) {

        while(getline( &: in,  &: line)){

            istringstream iss( str: line);

            getline( &: iss,  &: reservoir,  delim: ',');
            getline( &: iss,  &: municipality,  delim: ',');
            iss>>id;
            iss.ignore();
            getline( &: iss,  &: code,  delim: ',');
            iss>>maxDelivery;


            Reservoir *r = new Reservoir( &: reservoir, &: municipality,id, &: code,maxDelivery);
            reservoirs_.insert( x: { &: code,  &: r});
            graph_.addVertex(id,code);
        }

    } else cout << "Could not open the file\n";
}
```

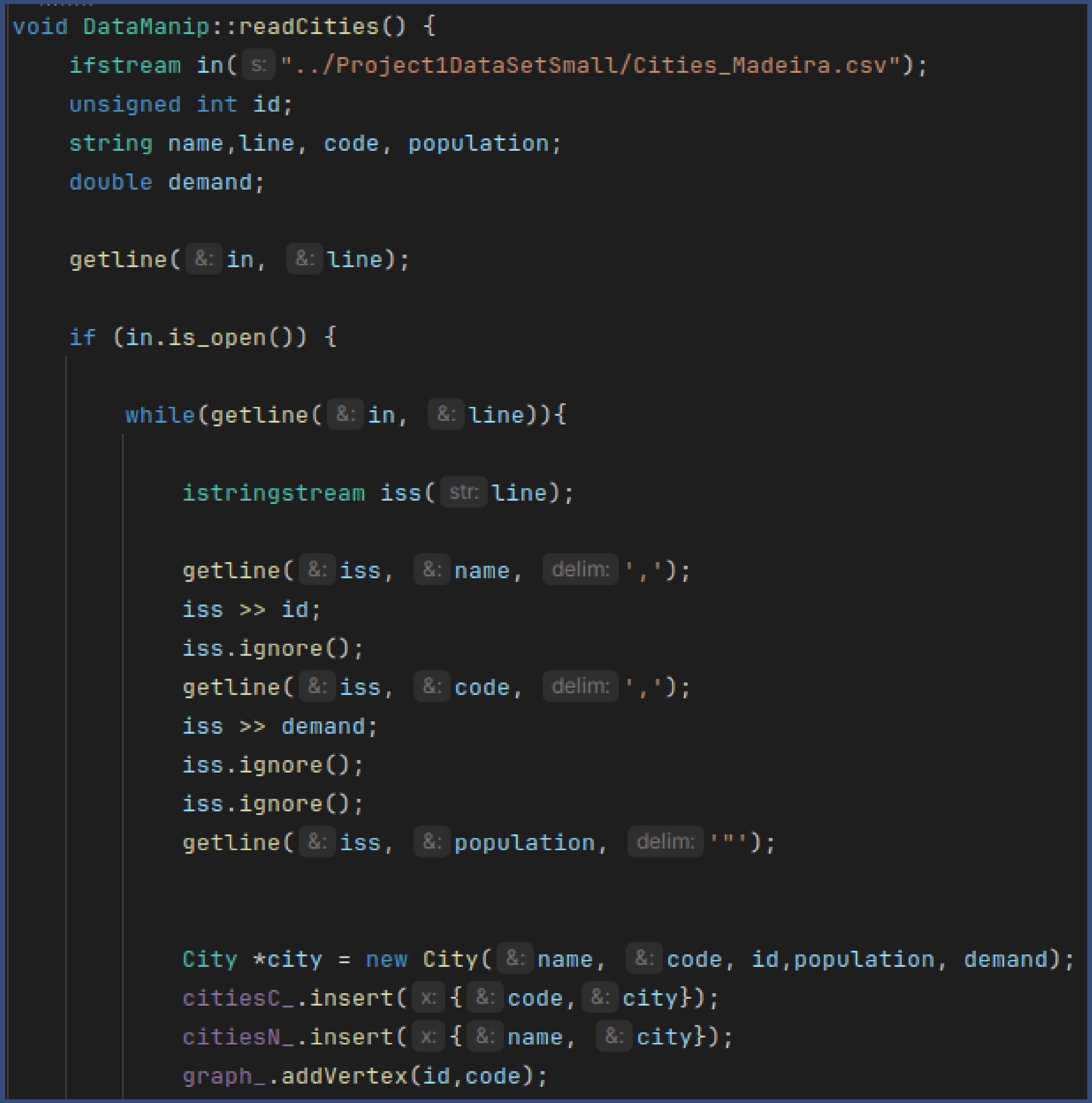


```
void DataManip::readPipes() {
    ifstream in( s: "../Project1DataSetSmall/Pipes_Madeira.csv");
    string service_point_a, service_point_b, line;
    unsigned int capacity, direction;

    getline( &: in,  &: line);

    if(in.is_open()) {
        while(getline( &: in,  &: line)){
            istringstream iss( str: line);

            getline( &: iss,  &: service_point_a,  delim: ',');
            getline( &: iss,  &: service_point_b,  delim: ',');
            iss>>capacity;
            iss.ignore();
            iss>>direction;

            if (direction == 1){
                graph_.addEdge( sourceCode: service_point_a,  destCode: service_point_b, capacity);
            }
            else{
                graph_.addEdge( sourceCode: service_point_a,  destCode: service_point_b, capacity);
                graph_.addEdge( sourceCode: service_point_b,  destCode: service_point_a, capacity);
            }
        }

    } else cout << "Could not open the file\n";
}
```

LEITURA DATASET



```
void DataManip::readStations() {
    ifstream in( s: "../Project1DataSetSmall/Stations_Madeira.csv");
    unsigned int id;
    string code,line;

    getline( &: in,  &: line);

    if (in.is_open()) {

        while(getline( &: in,  &: line)){

            if (line.front() == ',')
                continue;

            istringstream iss( str: line);
            iss>>id;
            iss.ignore();
            getline( &: iss,  &: code,  delim: ',');



            Station *station = new Station(id, &: code);
            stations_.insert( x: { &: code,  &: station});
            graph_.addVertex(id,code);
        }

    } else
        cout << "Could not open the file\n";
```

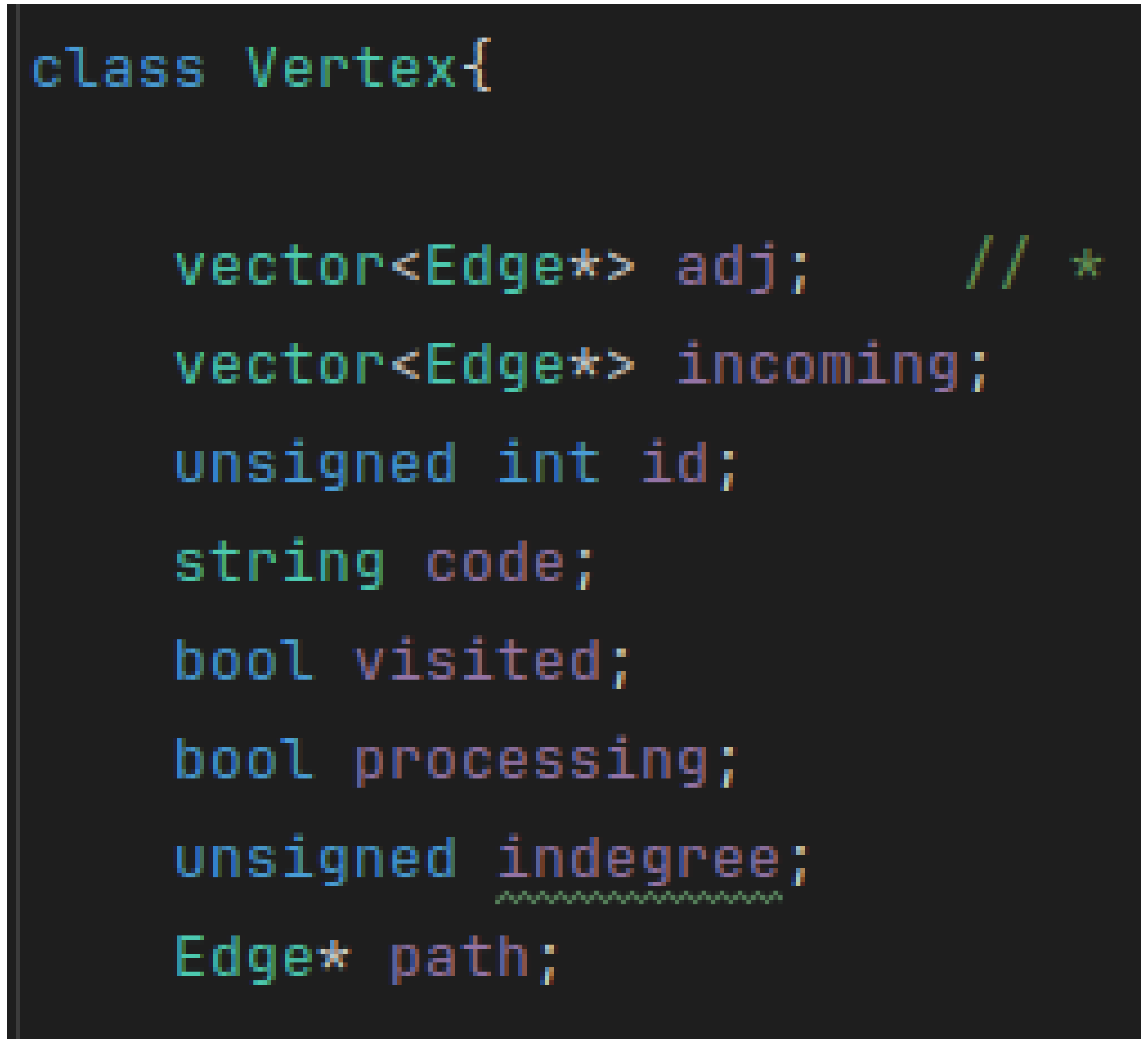


```
void DataManip::readCities() {
    ifstream in( s: "../Project1DataSetSmall/Cities_Madeira.csv");
    unsigned int id;
    string name,line, code, population;
    double demand;

    getline( &: in,  &: line);

    if (in.is_open()) {

        while(getline( &: in,  &: line)){

            istringstream iss( str: line);

            getline( &: iss,  &: name,  delim: ',');
            iss >> id;
            iss.ignore();
            getline( &: iss,  &: code,  delim: ',');
            iss >> demand;
            iss.ignore();
            iss.ignore();
            getline( &: iss,  &: population,  delim: '"');


            City *city = new City( &: name,  &: code, id,population, demand);
            citiesC_.insert( x: { &: code,  &: city});
            citiesN_.insert( x: { &: name,  &: city});
            graph_.addVertex(id,code);
```

GRAFO

Vertex:

- Vetor de Edges
- Vetor de Edges que entram
- Unsigned int id
- Códigos
- Booleana visited: para ver se o vértice já foi visitado
- Booleana processing
- Unsigned indegree
- Edge path

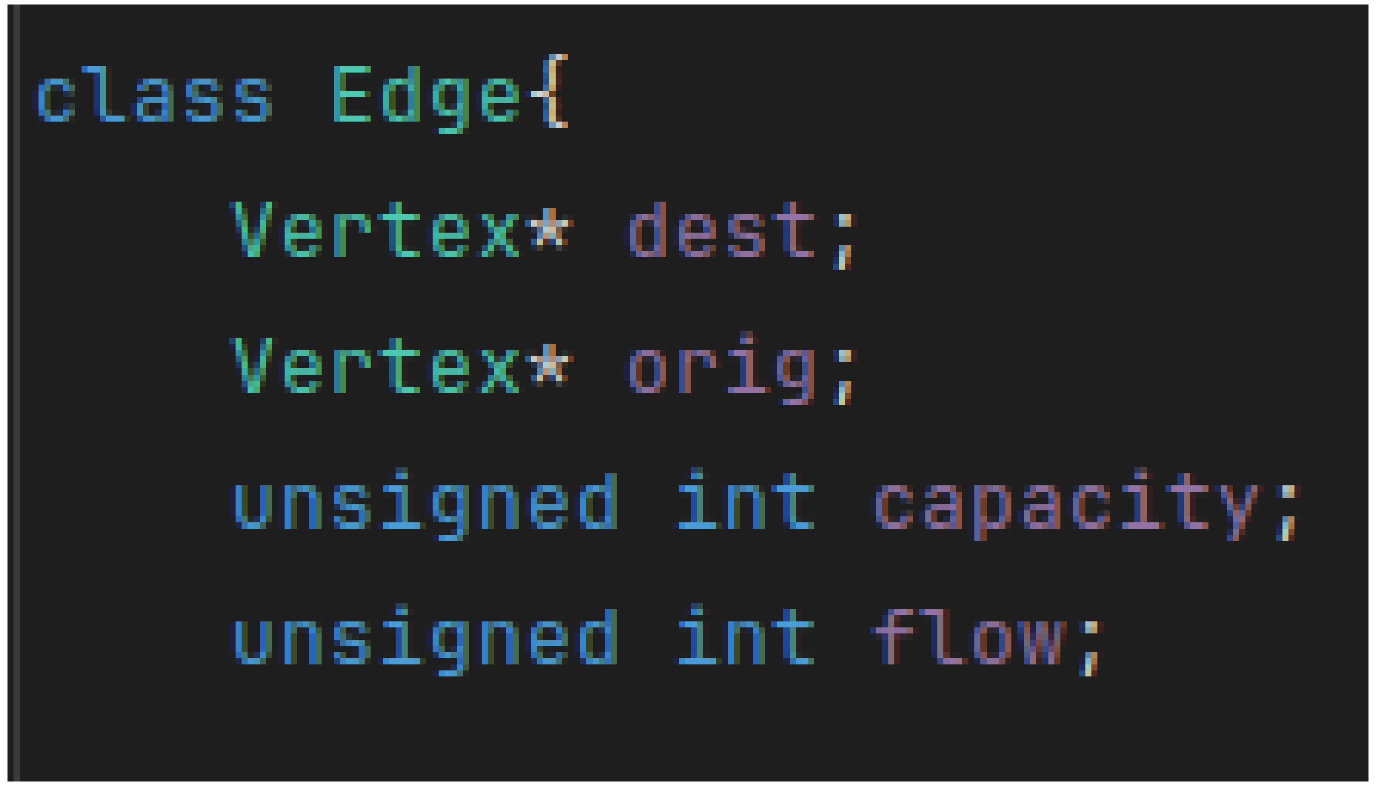


```
class Vertex{

    vector<Edge*> adj;    // *
    vector<Edge*> incoming;
    unsigned int id;
    string code;
    bool visited;
    bool processing;
    unsigned indegree;
    Edge* path;
```

GRAFO

Edge:

- Vértice Destino
- Vértice Origem
- Capacidade
- Fluxo

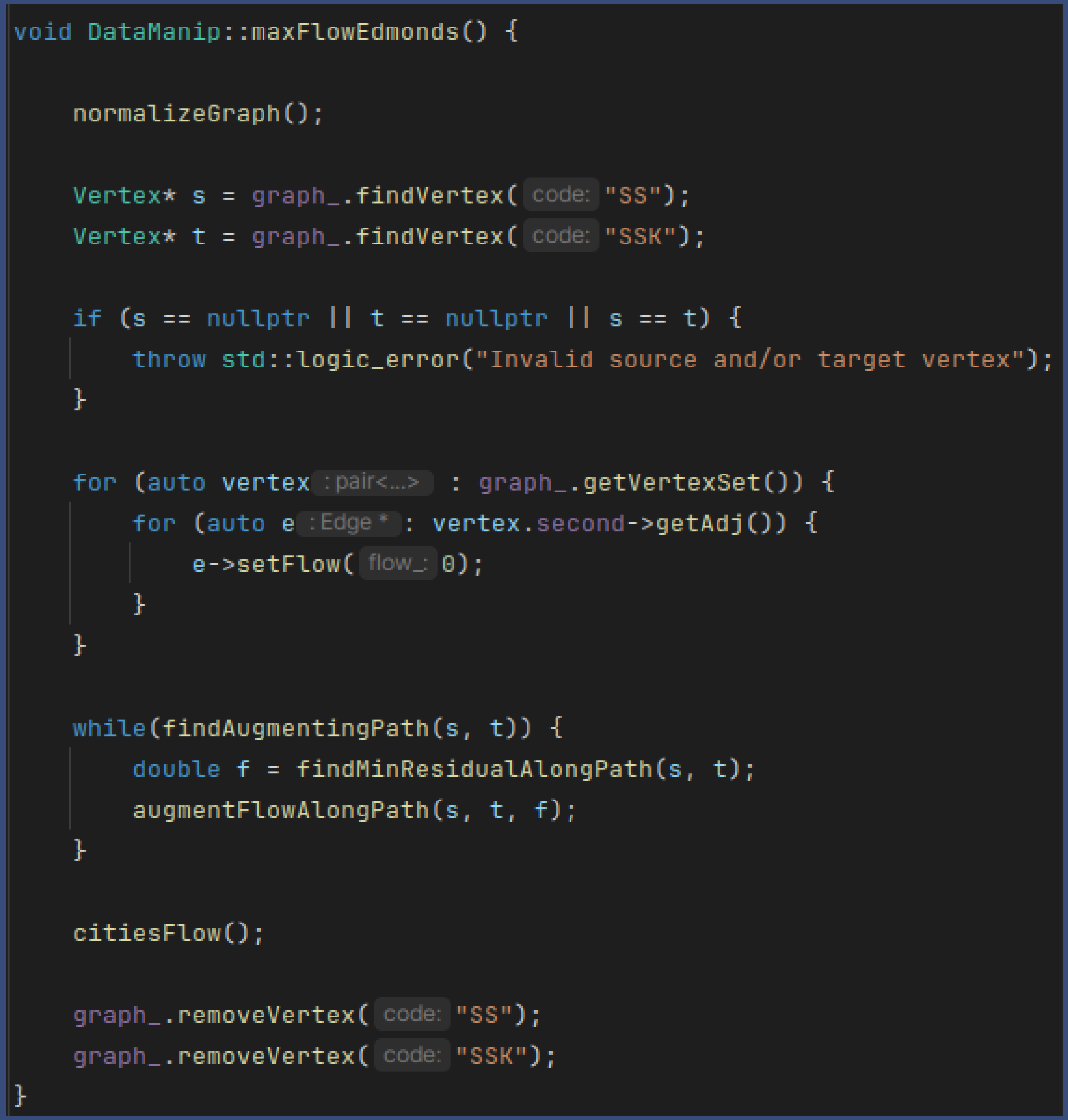


```
class Edge{
    Vertex* dest;
    Vertex* orig;
    unsigned int capacity;
    unsigned int flow;
```

FUNÇÕES IMPLEMENTADAS

- Fluxo máximo:
  - para uma dada “sink”
  - de uma “source” para uma “sink”
  - O maior de toda a rede

- Algoritmo de Edmonds-Karp
- Time Complexity: O (V * E * E)

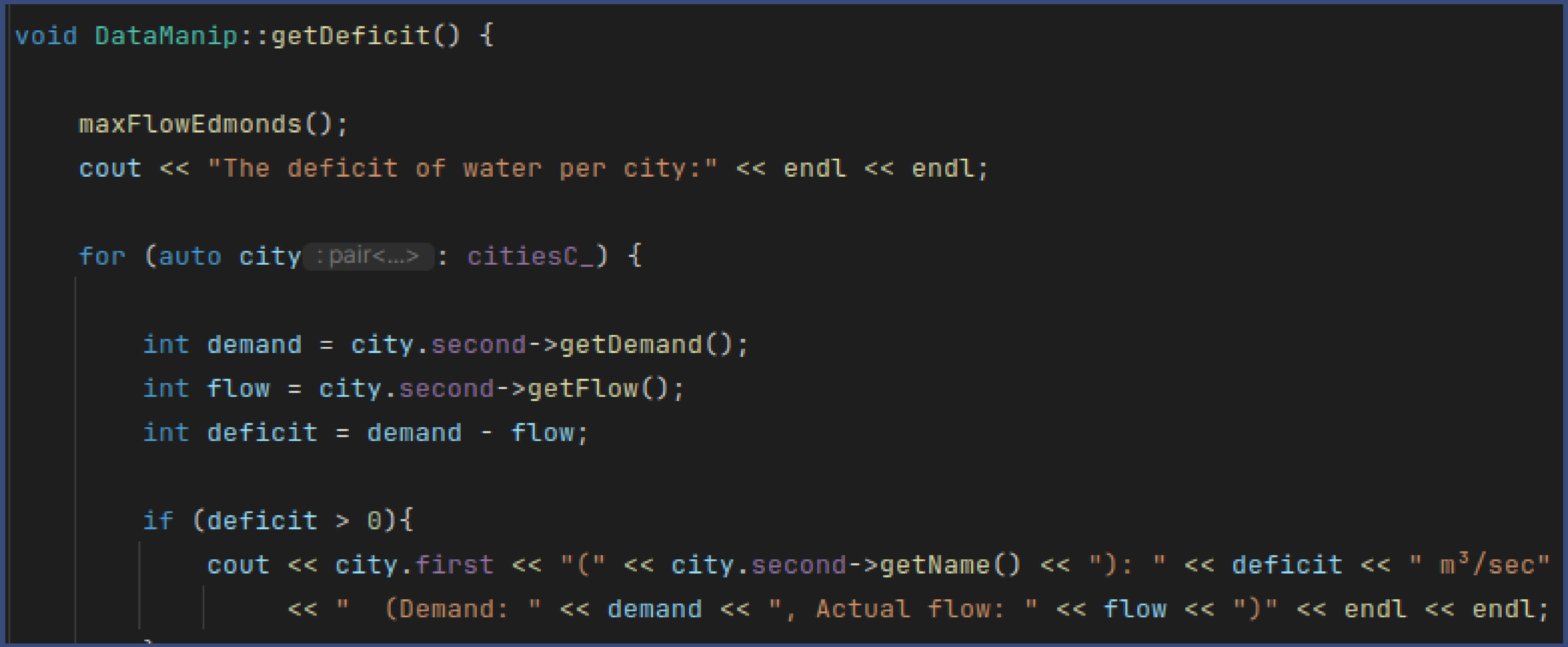


```
void DataManip::maxFlowEdmonds() {

    normalizeGraph();

    Vertex* s = graph_.findVertex( code: "SS");
    Vertex* t = graph_.findVertex( code: "SSK");

    if (s == nullptr || t == nullptr || s == t) {
        throw std::logic_error("Invalid source and/or target vertex");
    }

    for (auto vertex : pair<...> : graph_.getVertexSet()) {
        for (auto e : Edge * : vertex.second->getAdj()) {
            e->setFlow( flow_: 0);
        }
    }

    while(findAugmentingPath(s, t)) {
        double f = findMinResidualAlongPath(s, t);
        augmentFlowAlongPath(s, t, f);
    }

    citiesFlow();

    graph_.removeVertex( code: "SS");
    graph_.removeVertex( code: "SSK");
}
```

FUNÇÕES IMPLEMENTADAS

- Défice das cidades:
    - obter demanda e fluxo das cidades, permitindo perceber as diferenças entre oferta e procura

- Algoritmo de Edmonds-Karp
- Time Complexity: O (n)

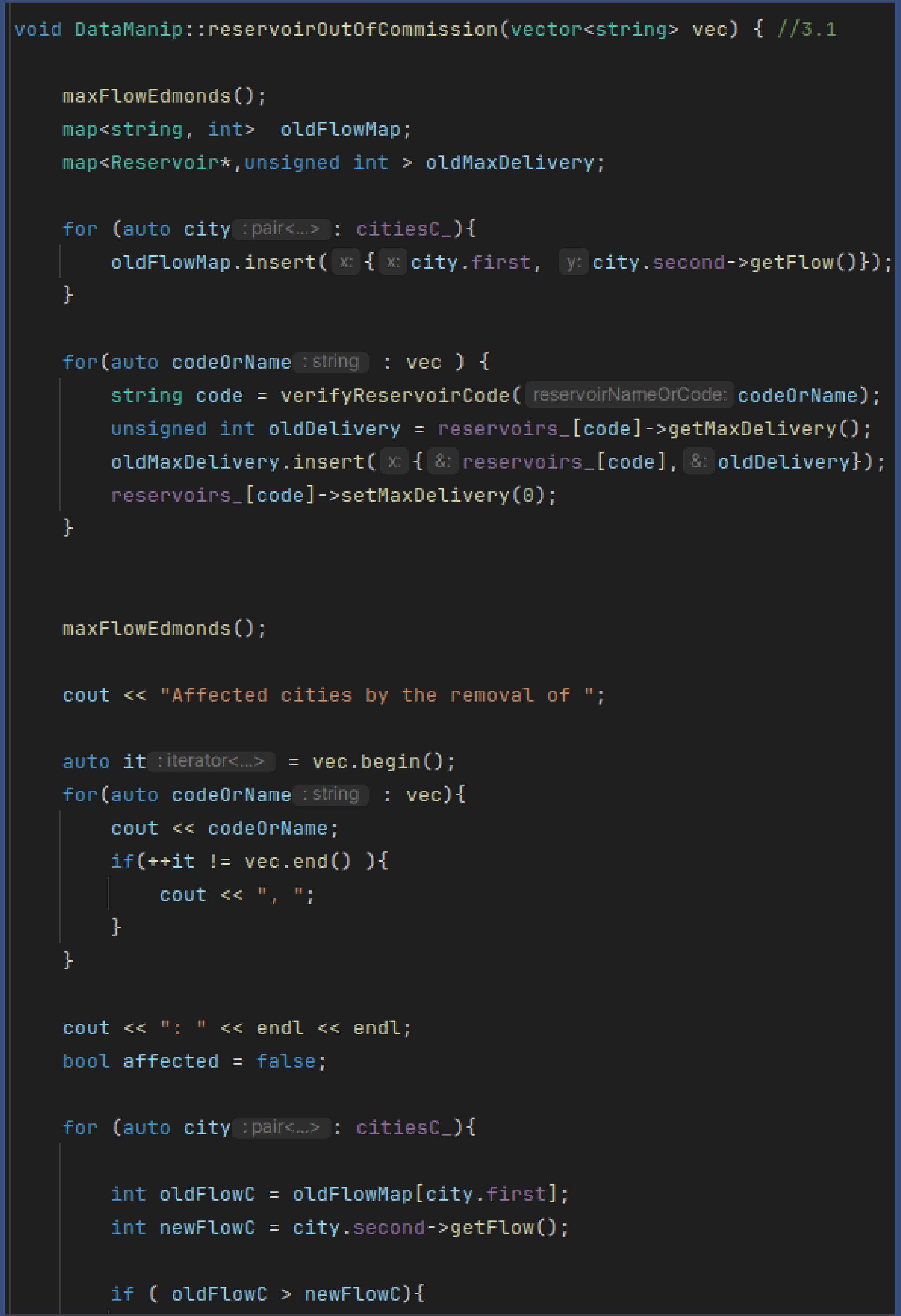


```
void DataManip::getDeficit() {

    maxFlowEdmonds();
    cout << "The deficit of water per city:" << endl << endl;

    for (auto city :pair<...> : citiesC_) {

        int demand = city.second->getDemand();
        int flow = city.second->getFlow();
        int deficit = demand - flow;

        if (deficit > 0){
            cout << city.first << "(" << city.second->getName() << "): " << deficit << " m³/sec"
                 << "  (Demand: " << demand << ", Actual flow: " << flow << ")" << endl << endl;
```

FUNÇÕES IMPLEMENTADAS

- Remover um Reservatório:
    - Perceber que cidades são afetadas após essa remoção
    - Verificar o que era recebido por cada cidade e, após a execução do algoritmo, ver o equilíbrio adotado para o novo fluxo

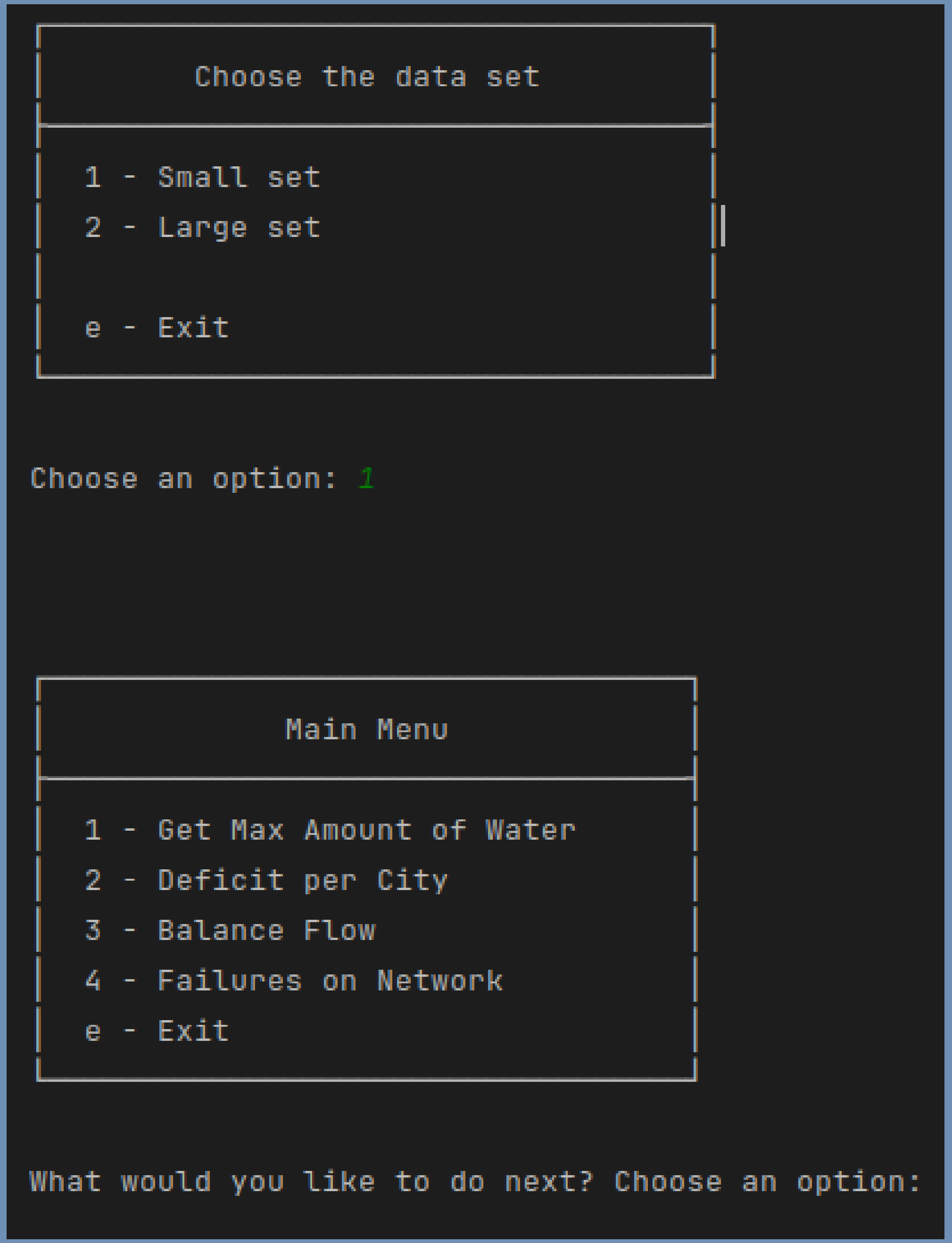
- Algoritmo de Edmonds-Karp
- Time Complexity: O (V *E*E)



```
void DataManip::reservoirOutOfCommission(vector<string> vec) { //3.1

    maxFlowEdmonds();
    map<string, int>  oldFlowMap;
    map<Reservoir*,unsigned int > oldMaxDelivery;

    for (auto city :pair<...> : citiesC_){
        oldFlowMap.insert( x: { x: city.first,  y: city.second->getFlow()});
    }

    for(auto codeOrName :string  : vec ) {
        string code = verifyReservoirCode( reservoirNameOrCode: codeOrName);
        unsigned int oldDelivery = reservoirs_[code]->getMaxDelivery();
        oldMaxDelivery.insert( x: { &: reservoirs_[code], &: oldDelivery});
        reservoirs_[code]->setMaxDelivery(0);
    }


    maxFlowEdmonds();

    cout << "Affected cities by the removal of ";

    auto it :iterator<...>  = vec.begin();
    for(auto codeOrName :string  : vec){
        cout << codeOrName;
        if(++it != vec.end() ){
            cout << ", ";
        }
    }

    cout << ": " << endl << endl;
    bool affected = false;

    for (auto city :pair<...> : citiesC_){

        int oldFlowC = oldFlowMap[city.first];
        int newFlowC = city.second->getFlow();

        if ( oldFlowC > newFlowC){
```

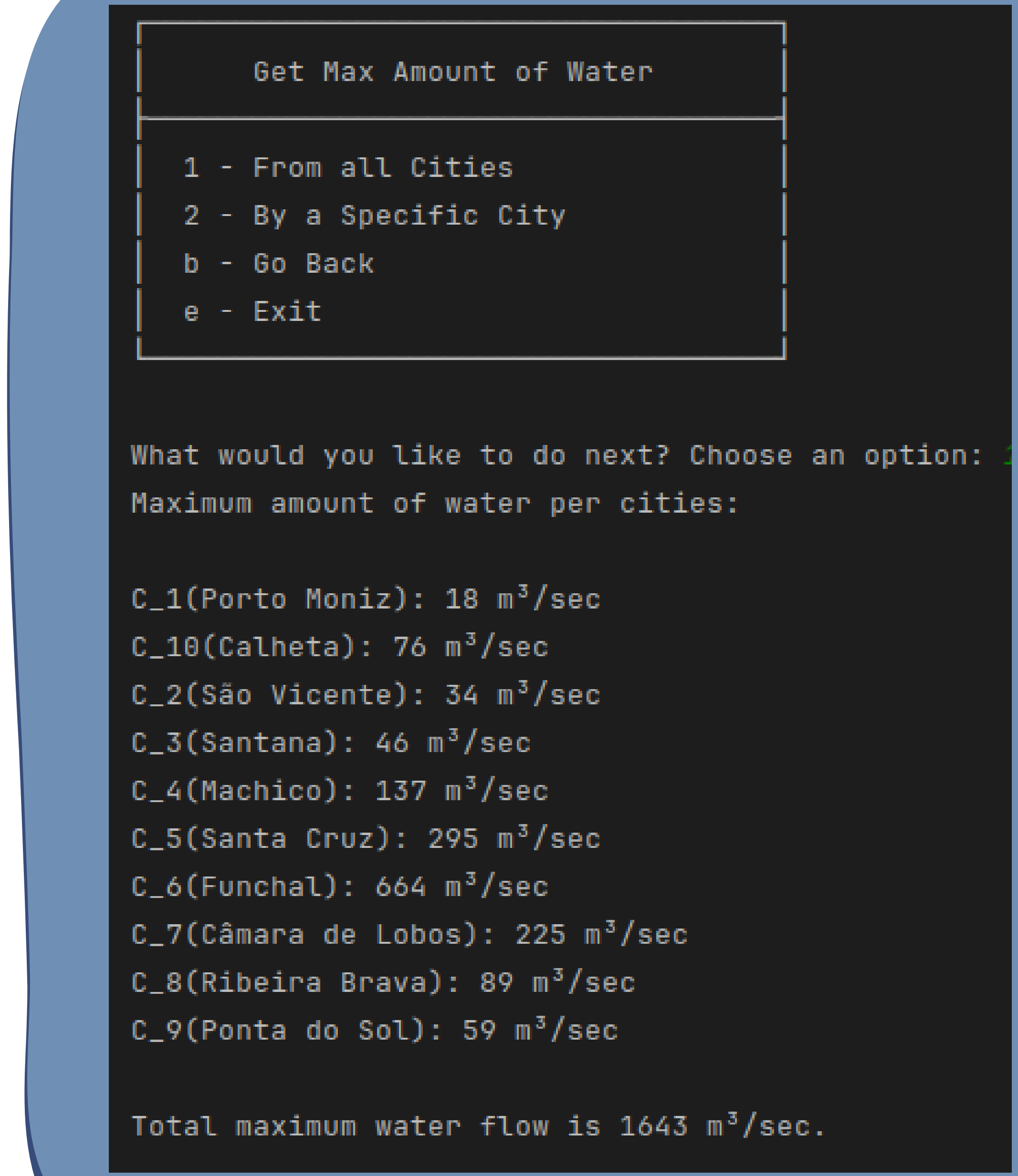
INTERFACE

- A partir do nosso menu principal, é possível consultar:
  - Máxima quantidade de água
  - Défice por cidade
  - Fluxo Equilibrado
  - Falhas na Rede de Água

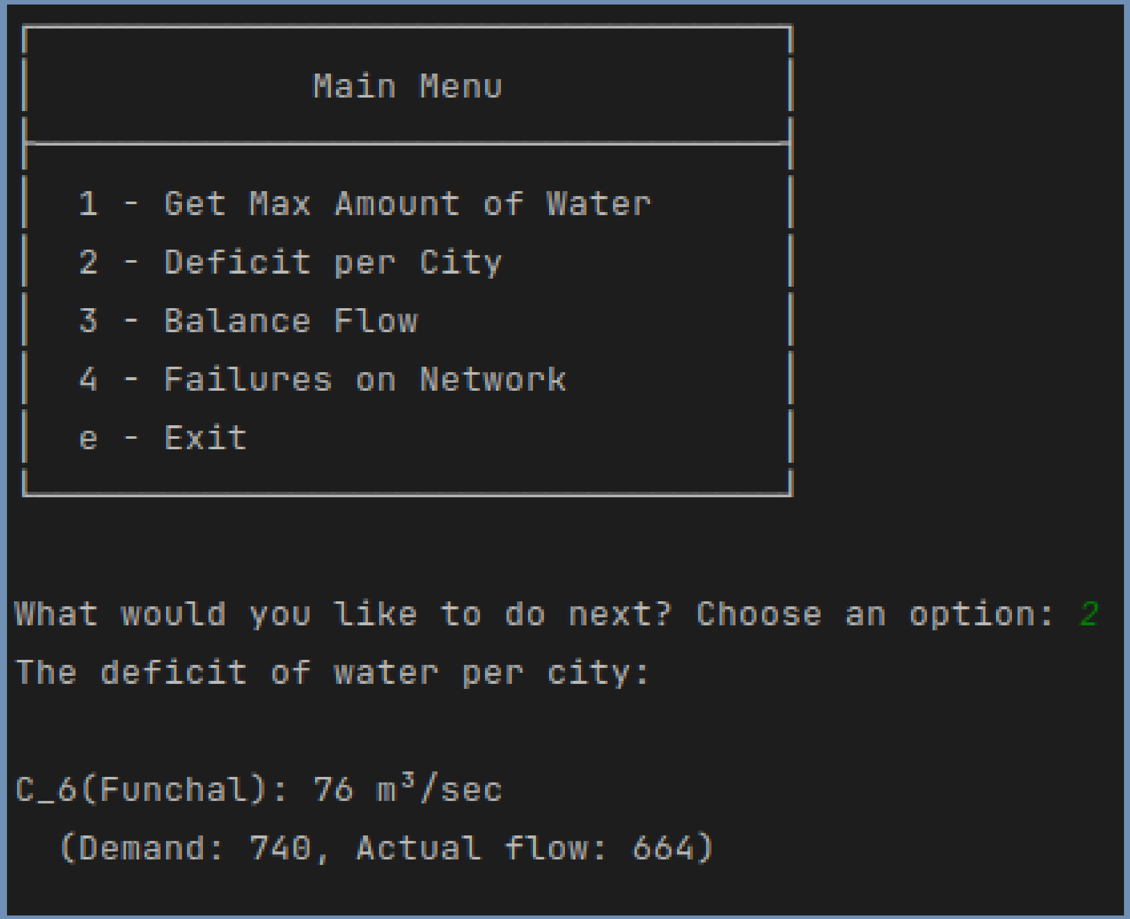


```
 ______________________________________
|                                      |
|          Choose the data set         |
|______________________________________|
|                                      |
|  1 - Small set                       |
|  2 - Large set                       |
|                                      |
|  e - Exit                            |
|______________________________________|


Choose an option: 1



 ____________________________________
|                                    |
|              Main Menu             |
|____________________________________|
|                                    |
|  1 - Get Max Amount of Water       |
|  2 - Deficit per City              |
|  3 - Balance Flow                  |
|  4 - Failures on Network           |
|  e - Exit                          |
|____________________________________|


What would you like to do next? Choose an option:
```

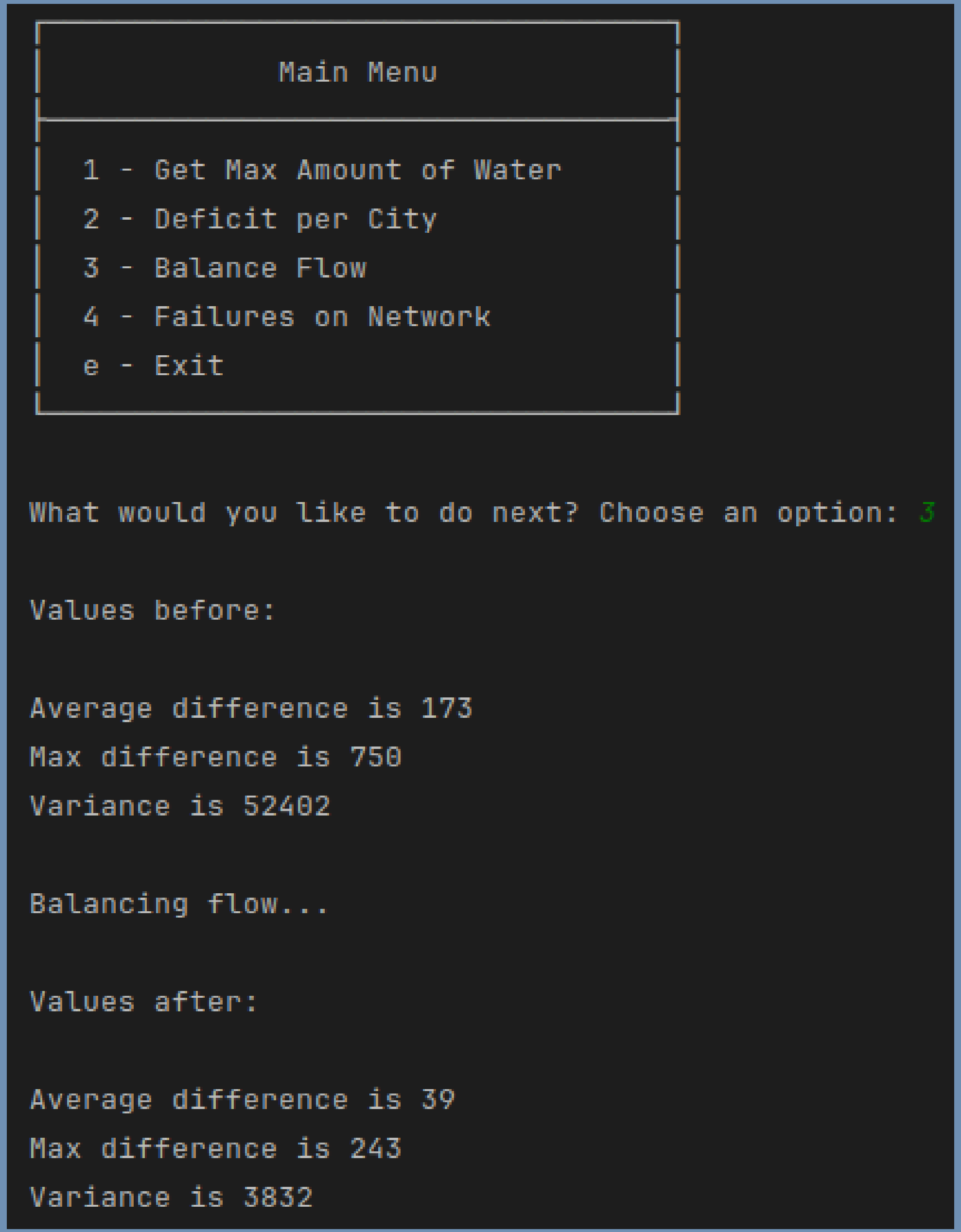
INTERFACE



```
┌─────────────────────────────────────┐
│     Get Max Amount of Water         │
├─────────────────────────────────────┤
│  1 - From all Cities                │
│  2 - By a Specific City             │
│  b - Go Back                        │
│  e - Exit                           │
└─────────────────────────────────────┘

What would you like to do next? Choose an option: 1
Maximum amount of water per cities:

C_1(Porto Moniz): 18 m³/sec
C_10(Calheta): 76 m³/sec
C_2(São Vicente): 34 m³/sec
C_3(Santana): 46 m³/sec
C_4(Machico): 137 m³/sec
C_5(Santa Cruz): 295 m³/sec
C_6(Funchal): 664 m³/sec
C_7(Câmara de Lobos): 225 m³/sec
C_8(Ribeira Brava): 89 m³/sec
C_9(Ponta do Sol): 59 m³/sec

Total maximum water flow is 1643 m³/sec.
```



```
┌─────────────────────────────────────┐
│              Main Menu              │
├─────────────────────────────────────┤
│  1 - Get Max Amount of Water        │
│  2 - Deficit per City               │
│  3 - Balance Flow                   │
│  4 - Failures on Network            │
│  e - Exit                           │
└─────────────────────────────────────┘

What would you like to do next? Choose an option: 2
The deficit of water per city:

C_6(Funchal): 76 m³/sec
  (Demand: 740, Actual flow: 664)
```



```
┌─────────────────────────────────────┐
│              Main Menu              │
├─────────────────────────────────────┤
│  1 - Get Max Amount of Water        │
│  2 - Deficit per City               │
│  3 - Balance Flow                   │
│  4 - Failures on Network            │
│  e - Exit                           │
└─────────────────────────────────────┘

What would you like to do next? Choose an option: 3

Values before:

Average difference is 173
Max difference is 750
Variance is 52402

Balancing flow...

Values after:

Average difference is 39
Max difference is 243
Variance is 3832
```

DIFICULDADES

Ao longo deste projeto, fomos por várias vezes desafiados. Um dos maiores desafios foi implementar a função que equilibrava o fluxo de água, pois não incidia sobre nenhum algoritmo em concreto, antes estudado. No entanto, em relação à utilização de grafos já estávamos bastante confortáveis.

Participação:

- Ana Sofia Baptista - 33.3
- Eduardo Santos - 33.3
- Pedro Pedro - 33.3

Obrigado