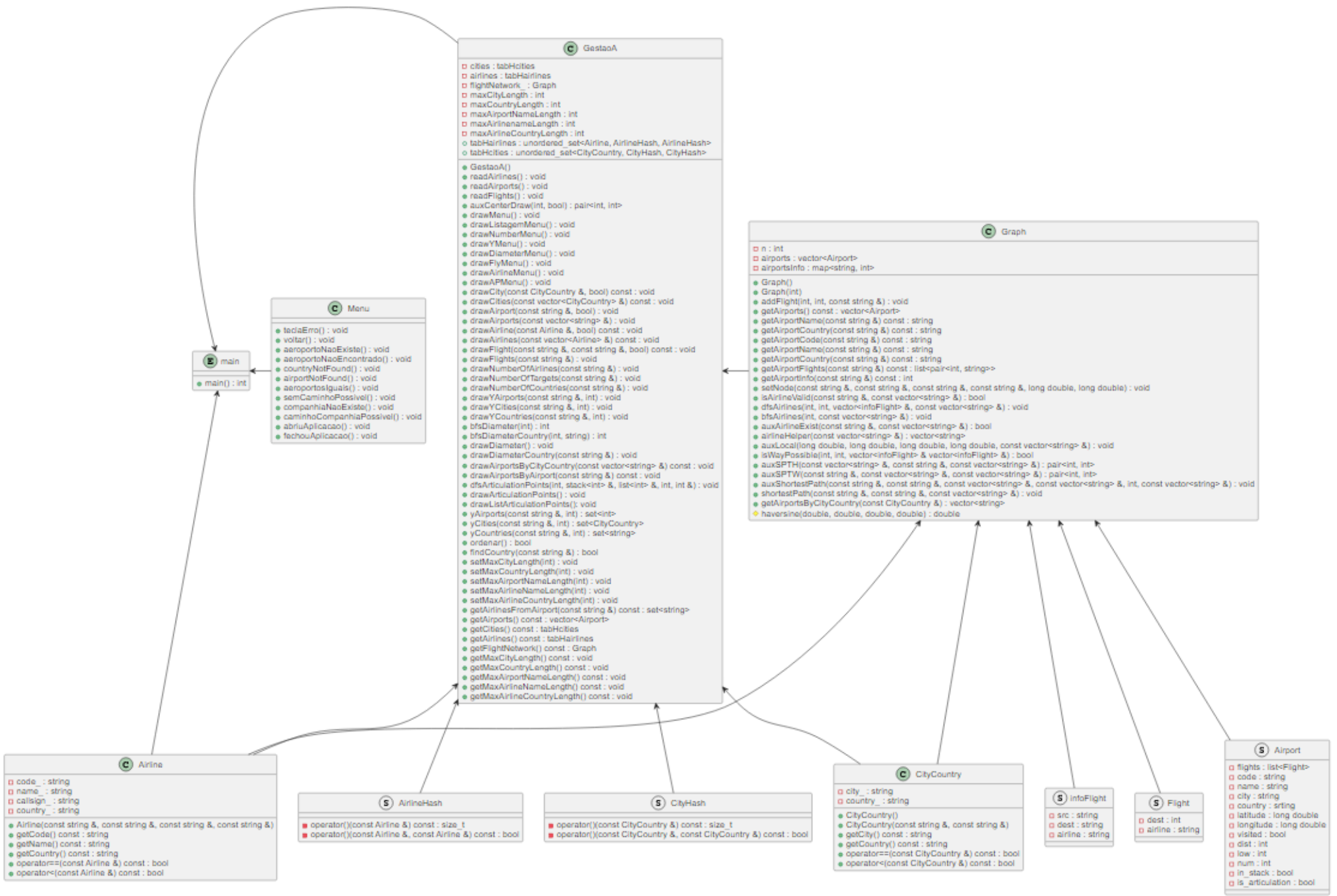
Trabalho 2 AED

Gonçalo Martins | up202108707
Leonor Filipe | up202204354
Luís Alves | up202108727

FEUP | 2022-2023
2LEIC15 + 2LEIC06

Diagrama de classes



Leitura do dataset a partir dos ficheiros dados

- De forma a armazenar as informações fornecidas, foi implementado um método por cada ficheiro na classe GestaoA, que cria uma variável ifstream para abertura e leitura das respetivas linhas:
  - **Método readAirlines**: leitura do ficheiro airlines.csv e armazenamento das companhias aéreas num unordered_set
  - **Método readAirports**: leitura do ficheiro airports.csv, armazenamento das cidades e respetivos países num unordered_set e criação de um nó no grafo por cada aeroporto
  - **Método readFlights**: leitura do ficheiro flights.csv e criação de uma aresta no grafo por cada voo, que conecta o aeroporto de destino ao aeroporto de origem desse mesmo voo

Grafo utilizado para representação do dataset

- De forma a representar o dataset da melhor forma possível, foi utilizado um **grafo dirigido e não pesado**
- Os **nós** representam os **aeroportos** (cuja posição identifica internamente cada aeroporto da classe Graph)
- As **arestas** correspondem aos **voos** realizados entre cada par de aeroportos

```
struct infoFlight {
    string src;
    string dest;
    string airline;
};

struct Flight {
    int dest;       // node de destino
    string airline; // airline
};

struct Airport {
    list<Flight> flights;  // lista de flights/edges deste aeroporto (adjacências)
    string code;           // código
    string name;           // nome
    string city;           // cidade
    string country;        // país
    long double latitude;  // latitude
    long double longitude; // longitude
    bool visited;          // para futuras pesquisas
    int dist;              // dist to source node
    int low, num;
    bool in_stack;         // aeroporto está ou não na stack
    bool is_articulation;  // aeroporto é ou não ponto de articulação
};

Graph();                   // default constructor
explicit Graph(int nodes); // Constructor: n.º nodes
```

Funcionalidades implementadas e algoritmos

1. **Menor número de voos dados dois locais – opção "Viajar entre dois locais"**

   "Local" e "Rede" de voos:

   - **Diretamente de um aeroporto para outro**
     - Usar qualquer companhia
     - Usar apenas uma companhia
     - Usar um conjunto de companhias
   - **Entre duas cidades**
     - Usar qualquer companhia
     - Usar apenas uma companhia
     - Usar um conjunto de companhias
   - **Entre duas localizações (coordenadas)**
     - Usar qualquer companhia
     - Usar apenas uma companhia
     - Usar um conjunto de companhias

   - **Algoritmo**: adaptação de **pesquisa em largura (BFS)**; utilização da **fórmula de Haversine**
   - **Implementação**:
     - **Algoritmo BFS**: utilizado para determinar a distância (nº de nós) a que cada aeroporto se encontra do ponto de partida
     - **Fórmula de Haversine**: implementada no cálculo da distância entre dois pontos, através das suas latitudes e longitudes
   - **Complexidade temporal**: O(|V|+|E|) na primeira opção e O(n) nas restantes

Funcionalidades implementadas e algoritmos

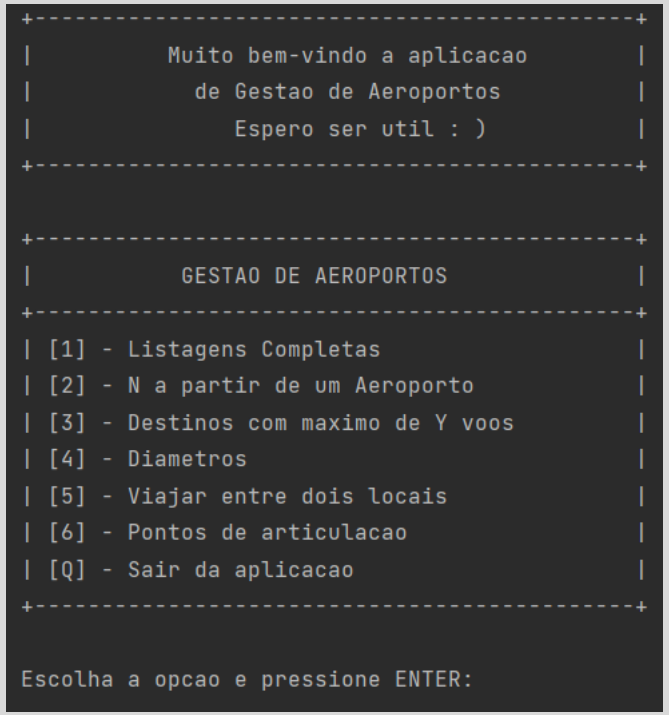
2. Informações sobre um aeroporto

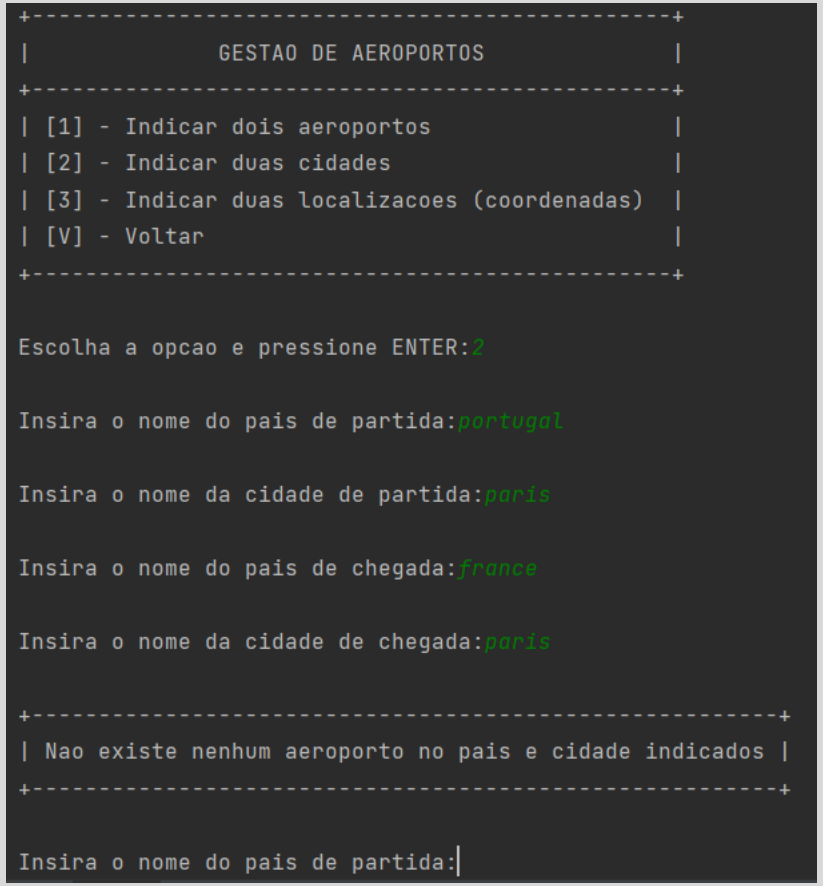
- **Listagens completas**
  - Listar cidades
  - Listar aeroportos
  - Listar companhias aéreas
  - Listar voos
  - Listar aeroportos numa cidade
  - Listar companhias aéreas num aeroporto
- **N a partir de um aeroporto**
  - Nº companhias aéreas num aeroporto
  - Nº destinos diferentes a partir de um aeroporto
  - Nº países diferentes a partir de um aeroporto
- **Destinos com máximo de Y voos**
  - Aeroportos atingíveis num máximo de Y voos
  - Cidades atingíveis num máximo de Y voos
  - Países atingíveis num máximo de Y voos

- **Algoritmo**: adaptação de **pesquisa em largura (BFS)** na opção N a partir de um aeroporto
- **Implementação**:
  - **Algoritmo BFS**: utilizado para determinar a distância (nº de nós) a que cada aeroporto se encontra do ponto de partida
- **Complexidade temporal**: $O(n^2)$ nas listagens e nos destinos com máximo de Y voos, e $O(n)$ em N a partir de um aeroporto

Interface com o utilizador

- Este projeto dispõe de um **menu interativo, intuitivo e amigável**, com **múltiplas opções encadeadas** dentro de cada funcionalidade diferente
- Todos os **inputs** do utilizador são **validados**, apresentando **mensagens de erro claras e sucintas** quando estes estão incorretos





Destaque de funcionalidades

- **Menor número de voos dados dois locais**
- **Destinos com máximo de Y voos**

Dificuldades encontradas

- Complexidade elevada do projeto
- Dificuldade em otimizar a complexidade dos algoritmos
- Dificuldade em resolver alguns bugs

Esforço de cada elemento do grupo

- Gonçalo: 33.3%
- Leonor: 33.3%
- Luís: 33.3%

Tarefas de valorização

- **Diâmetros**
  - Cálculo do diâmetro (grafo geral)
  - Cálculo do diâmetro para um país específico
- **Pontos de articulação**
  - Nº pontos de articulação
  - Listagem dos pontos de articulação

- **Algoritmo**: adaptação de **pesquisa em profundidade (DFS)** e **pesquisa em largura (BFS)**
- **Implementação**:
  - **Algoritmo BFS**: utilizado para determinar a distância (nº de nós) a que cada aeroporto se encontra do ponto de partida e, assim, possibilitar encontrar a maior distância (diâmetro)
  - **Algoritmo DFS**: utilizado para procurar os pontos de articulação do grafo
- **Complexidade temporal**: O(|V|*(|V|+|E|))

E: número total de arestas do grafo
N: tamanho/número total de nós do grafo