Trabalho prático 1

Horários de estudantes L.EIC

A elaboração dos horários da L.EIC é uma tarefa complexa. O propósito deste trabalho não é a elaboração de horários, mas sim o desenvolvimento de um sistema capaz de ajudar na gestão de horários após a sua elaboração (alteração, visualização, …).

O ficheiro **schedule.zip** contém informação real sobre os horários da L.EIC este semestre (os dados dos estudantes estão anonimizados). A seguir, é fornecida a indicação detalhada dos dados aí presentes.

Note que o horário de um estudante pode apresentar sobreposição de aulas, se estas não forem ambas do tipo TP ou PL. Isto é, pode verificar-se sobreposição de aulas T com TP, T com T, T com PL.

2022 - A 1S 1T 2T SP

Horas	Segunda	Terça	Quarta	Quinta	Sexta	Sábado
08:00 - 08:30					MD (T) 1LEIC01 B001	
08:30 - 09:00	AM I (T) 1LEIC01 B001	AM I (TP) 1LEIC01 B329				
09:00 - 09:30				FP (TP) 1LEIC01 B301		
09:30 - 10:00			FP (T) 1LEIC01 B003			
10:00 - 10:30					FP (T) 1LEIC01 B001	
10:30 - 11:00	ALGA (TP) 1LEIC01 B339	ALGA (T) 1LEIC01 B001	FSC (TP) 1LEIC01 B335			
11:00 - 11:30				FSC (T) 1LEIC01 B002	MD (TP) 1LEIC01 B327	
11:30 - 12:00						
12:00 - 12:30						
12:30 - 13:00						
13:00 - 13:30						
13:30 - 14:00						
14:00 - 14:30			PUP (TP) 1LEIC01 B102			
14:30 - 15:00						
15:00 - 15:30						
15:30 - 16:00						
16:00 - 16:30						
16:30 - 17:00						
17:00 - 17:30						
17:30 - 18:00						
18:00 - 18:30						
18:30 - 19:00						
19:00 - 19:30						
19:30 - 20:00						

Dados fornecidos

Os dados estão disponíveis em **schedule.zip** sob a forma de ficheiros csv (*comma separated values*).

- Ficheiro **classes_per_uc**

  Este ficheiro contém as turmas existentes em cada UC.

  A primeira linha inclui os cabeçalhos*: UcCode* (código da unidade curricular), *ClassCode* (código da turma)

```
UcCode,ClassCode
L.EIC001,1LEIC01
L.EIC001,1LEIC02
```

- Ficheiro **classes**

  Este ficheiro contém o horário para cada aula de todas as turmas das UCs..

  A primeira linha inclui os cabeçalhos*: ClassCode* (código da turma), *UcCode* (código da unidade curricular), *Weekday* (dia da semana), *StartHour* (hora de início da aula), *Duration* (duração da aula em horas), *Type* (tipo de aula: T, TP, PL)

```
ClassCode,UcCode,Weekday,StartHour,Duration,Type
1LEIC01,L.EIC001,Monday,10.5,1.5,TP
1LEIC02,L.EIC001,Thursday,9.5,1.5,TP
1LEIC03,L.EIC001,Tuesday,9,1.5,TP
```

- Ficheiro **students_classes**

  Este ficheiro contém as turmas dos estudantes em cada UC.

  A primeira linha inclui os cabeçalhos*: StudentCode* (código do estudante), *StudentName* (nome do estudante), *UcCode* (código da unidade curricular), *ClassCode* (código da turma)

```
StudentCode,StudentName,UcCode,ClassCode
202025232,Iara,L.EIC002,1LEIC05
202031607,Gisela,L.EIC004,1LEIC08
202031607,Gisela,L.EIC005,1LEIC08
202079037,Jose Jesualdo,L.EIC023,3LEIC08
```

Conceptualmente, uma aula é identificada por um código da UC, dia da semana, hora de início, duração e tipo (T/TP/PL). Um horário é um conjunto de aulas. Um estudante é identificado por um código e nome, e tem um horário associado. Uma turma é identificada por um código e tem um horário associado

Tarefas a implementar

Na implementação do trabalho, deve usar as estruturas de dados lista, vetor, fila e árvore binária de pesquisa (pode usar ainda outras estruturas, se considerar necessário). Essas estruturas servirão de base para as seguintes tarefas e nos seguintes contextos:

1. Leitura dos dados fornecidos e criação das estruturas de dados que representam o cenário referido. Deve usar as estruturas de dados lineares mais apropriadas em cada contexto, sendo, no entanto, sugerido e valorizado o uso da estrutura árvore binária de pesquisa[1] (estrutura de dados hierárquica) no armazenamento dos estudantes.
2. Listagens várias, totais e parciais, com critérios a definir pelo utilizador. Podem também ter ordenações distintas a definir pelo utilizador. Exemplos:
   - Ocupação de turmas/ano/UC (ordenação por UC, ordenação crescente, ordenação decrescente, …)
   - Horário de determinado estudante
   - Estudantes em determinada turma/UC/ano
   - Estudantes com mais de *n* UCs (implica uso de BST) (ordem alfabética, ordem nºs UCs,…)
   - …
3. Os pedidos de alteração de horários são guardados numa fila, sendo processados no final do dia. Os pedidos que não puderem ser satisfeitos, são guardados para arquivo em outra estrutura de dados à sua escolha. Os pedidos a tratar incluem:
   i. Remover estudante de turma/UC.
   ii. Adicionar estudante a uma turma/UC:
      a) Só é possível se o horário é compatível e a turma possui vaga. Considerar a capacidade de uma turma igual a um valor máximo **Cap**.
      b) Só é possível se o horário é compatível e não provoca desequilíbrio nas turmas dessa UC. Considerar que existe desequilíbrio nas turmas de uma UC se a diferença entre o nº de estudantes em duas quaisquer turmas dessa UC é >=4.
   iii. Alterar a turma/UC de um estudante. Considerar as alternativas a), b) do caso ii.
   iv. Alterar um conjunto de turmas/UCs de um estudante. Considerar as alternativas a) e b) do caso ii.

Implemente também outras funcionalidades que considere relevantes, para além dos requisitos globais enunciados.

[1] A estrutura árvore binária de pesquisa (BST) será lecionada mais adiante. O estudante pode e deve realizar o trabalho desde já, incluindo a BST mais tarde

Outros requisitos:

- O sistema deverá ter um menu que dispõe, de forma amigável, as funcionalidades implementadas.
- Deve incluir documentação do código implementado, usando Doxygen, e indicando a respetiva complexidade.

Resultados esperados

A aplicação a desenvolver deve permitir registar e gerir entidades e relações entre elas, fazendo uso das estruturas lineares vetor, lista e fila e da estrutura hierárquica árvore binária de pesquisa.

A aplicação a desenvolver deve:

- Utilizar **classes adequadas** para representação das entidades envolvidas
- Utilizar estruturas lineares (**vetores**, **listas** e **filas**) e a estrutura hierárquica **árvore binária de pesquisa** (esta última é um fator de valorização)
- Guardar informação em **ficheiros** para uso futuro
- Incluir documentação das funções implementadas, usando Doxygen, e indicando a sua complexidade.

A aplicação deve também permitir listagens várias:

- As listagens podem ser **totais** ou **parciais** com critérios a definir pelo utilizador
- Podem também ter **ordenações** distintas definidas pelo utilizador
- Devem ser usados algoritmos de **pesquisa** e **ordenação** para este efeito

Sobre a demonstração do trabalho:

- Preparação adequada para demonstrar as funcionalidades do trabalho
- Elaboração de uma apresentação (diapositivos) para suporte à demonstração