

Trabalho aed

HORÁRIOS ESTUDANTES L.EIC

Lara Bastos, up202108740

Lia Sobral, up202108741

Miguel Barros, up202108678

Problema

O trabalho tinha como objetivo a elaboração de um sistema capaz de ajudar a gestão do horário após a sua elaboração

IMPLEMENTAR:

- leitura dos dados nos ficheiros fornecidos
- criação de estruturas de dados que armazenem a informação

- fila com pedidos de troca de horários
- processamento de pedidos, no final do dia, que incluem:
 - remover estudantes da UC/turma
 - adicionar a nova UC/turma (se houver compatibilidade horários, vagas na turma e não causar discrepância entre lotação de turmas)
- fila com pedido de troca de horário recusados
- apresentar um menu
- incluir documentação do código, usando Doxygen



IMPLEMENTADA

criação de uma classe principal

"GestaoHorario" e um objeto desta de modo a chamar os seus métodos em "main.cpp"

```
class GestaoHorario {  
private:  
    set<Student> estudantes;  
    vector<TurmaH> turmas;  
    queue<Pedido> listaespera;  
    queue<Pedido> listaceites;  
    queue<Pedido> listarecusados;  
    list<UcTurma> ucturmas;  
public:  
    GestaoHorario() = default;  
    set<Student> getEstudantes() const;  
    vector<TurmaH> getTurmas() const;  
    list<UcTurma> getUcturmas() const;  
    queue<Pedido> getListaespera() const;  
    queue<Pedido> getListarecusados() const;  
    queue<Pedido> getListaceites() const;  
    void lerFicheiros(string path);  
    void guardarPedidos(string up, string uc, string classcode);  
    void processarPedidos(bool forçado);  
    Student findStudent(string up) const;  
    void listagem() const;  
};
```



lerFicheiros

lê os ficheiros com a informação de horários



guardarPedidos

recebe um pedido de mudança de turma e adiciona-o a uma lista de pedidos em espera



processarPedidos

verifica se os pedidos em espera são executáveis, se sim executa-os, se não adiciona-os a uma lista de pedidos recusados



IMPLEMENTADA

em "main.cpp" implementamos um switch/case que, de acordo com o input do utilizador:

1.mostra o perfil do aluno

2.1-recebe um pedido de mudança de turma

2-mostra o estado atual dos pedidos feitos

3-mostra o tempo restante até processar todos os pedido

4-forçar a processar pedidos

3.Ver turmas (horário e estudantes)

4.Mudar de estudante

```
1 - Ver perfil do aluno
2 - Pedidos
3 - Ver Turmas
4 - Mudar de estudante
5 - Terminar execucao do programa
```

```
Escolha uma opcao : |
```

```
1 - Criar novo pedido
2 - Ver estado atual dos meus pedidos
3 - Tempo restante ate processar os pedidos
4 - Forcar o processo de pedidos
```

```
Escolha uma opcao : |
```

CLASSES CRIADAS:

UC TURMA

- CÓDIGO UC
- TURMA
- LOTAÇÃO

BLOCO

- DIA DA SEMANA
- HORA INÍCIO
- DURAÇÃO
- TIPO

STUDENT

- UP
- NOME
- LIST<UCTURMAS>

TURMA H

- UCTURMA
- LIST<BLOCO>

PEDIDO

- STUDENT
- UCTURMA
- ESTADO

GESTÃO HORÁRIO

- SET<STUDENT>
- VECTOR<TURMAH>
- 3 QUEUE<PEDIDO>
- LIST<UCTURMA>

Funcionalidades

LISTAGENS COMPLETAS

```
set<Student> estudantes;  
vector<TurmaH> turmas;  
queue<Pedido> listaespera;  
queue<Pedido> listaceites;  
queue<Pedido> listarecusados;  
list<UcTurma> ucturmas;
```

em "GestaoHorario" temos armazenada toda a informação retirada dos ficheiros- **listagens completas**

SET<STUDENT>

set com todos os estudantes

VECTOR<TURMAH>

vetor com todas as turmas
H- cada UcTurma e os seus repetivos blocos

QUEUE<PEDIDO>

- pedidos em espera
- pedidos aceites
- pedidos recusados

LIST<UTURMA>

lista com todas as UcTurmas

Funcionalidades PESQUISA

FIND

no set de todos estudantes, para encontrar o pretendido, usamos o find-binary search tree

```
/// Binary Search Tree Implement

Student estudante( c: uc);

auto it :const_iterator<...> = estudantes.find( k: estudante);

if (it != estudantes.end()){

    estudante = *it;
}
else{
    cout << "UP Not Found" << endl;
    ver = false;
}
```

LISTAGENS PARCIAIS

na classe Student

LIST<UCTURMAS>

lista com as UcTurmas frequentadas por cada estudantes

na classe TurmaH

LIST<BLOCOS>

lista com os blocos existentes em cada UcTurma

Destaque DE FUNCIONALIDADE

armazenamento de variáveis com as horas atuais de modo a :

armazenar os pedidos ao longo do dia numa **listadeespera**



processá-los à meia noite

em qualquer momento, um aluno pode verificar o estado dos seus pedidos:

- em espera
- aceite
- ou recusado, e a razão

(antes de processar pedidos)

```
int seconds, minutes, hours;

if (!forcado) {
    time_t total_seconds = time(0);
    //getting values of seconds, minutes and hours
    struct tm *ct = localtime(&total_seconds);

    seconds = ct->tm_sec;
    minutes = ct->tm_min;
    hours = ct->tm_hour;
}

else { minutes = 0; hours = 0;}

if (hours == 0 && minutes == 0) {
```


ESFORÇO DOS ELEMENTOS:

- todos trabalhamos na elaboração da solução, criação das classes e os seus respetivos métodos e atributos
- Lara fez o powerpoint
- Lia fez a documentação
- Miguel fez a maior parte do menu

PRINCIPAIS DIFICULDADES:



Trabalhar com referências

ao alterar os atributos de alguns objeto
ex: atualizar lista das UcTurmas de um aluno quando alterávamos os horários



Usar find com objetos

foi necessário dar overload de alguns operadores(==;<)para conseguir usar o find em objetos



FIM

Obrigada pela atenção

