U.PORTO

FEUP FACULDADE DE ENGENHARIA
UNIVERSIDADE DO PORTO

Faculdade de Engenharia da Universidade do Porto

GESTÃO
DE FACULDADE
(TEMA 7)

Algoritmos e
Estruturas de Dados
2ºANO - **MIEIC**

Estudantes e Autores
Diogo Silva, up201706892
up201706892@fe.up.pt
Ana Mafalda Santos, up201706791
up201706791@fe.up.pt
Alexandre Carqueja, up201705049
up201705049@fe.up.pt

1 de **janeiro** de **2019**
2MIEIC**01_6**

Índice

Descrição do Tema

O tema deste trabalho é essencialmente criar um programa que facilite a gestão de uma **faculdade** dos seus **departamentos**, **cursos**, **disciplinas**, **estudantes**, **docentes** e **funcionários**.

Ou seja, a faculdade é tratada como uma pirâmide invertida: esta tem departamentos, estes por sua vez têm cursos, que por sua vez têm unidades curriculares.

A principal funcionalidade é permitir que os diversos membros da faculdade acedam e editem toda a informação envolvida nesta abstração mais simples e eficazmente.

De acordo com a função que desempenham do ponto de vista da faculdade – **visitante**, **professor**, **aluno** ou **administrador** – são capazes de realizar diferentes tarefas.

Solução Implementada

Como já tínhamos todas as classes implementadas (**College** (Faculdade), **Department** (Departamento), **Course** (Curso) e **UC** (Unidade Curricular), **People** (Pessoa), **Employee** (Empregado), **Student** (Estudante), **Teacher** (Docente) e **Staff** (Não Docente)) passámos para a segunda fase do trabalho, que se baseia maioritariamente no uso de novas estruturas de dados.

As informações relativas á faculdade estão então armazenadas de diferentes modos: em vetores, numa BST, numa Hash Table e numa Priority Queue.

A BST foi então usada para substituir o vetor de Students anteriormente presente na classe College. Com esta podemos editar toda a informação de um estudante, eliminá-lo e adicionar novos, caso necessário. O *header* utilizado para a implementação da BST foi o ficheiro BST.h fornecido pelos docentes da disciplina.

Para além disso foi-nos também pedido para implementar uma *priority queue* (fila de prioridade) que guardasse os alunos com o fim de lhes atribuir bolsas. Para isto foi necessário criarmos o novo membro dado "bolsas" e colocar o aluno com melhor média à frente da fila. Após a implementação desta estrutura de dados decidimos acrescentar um novo menu ao programa com mais algumas funcionalidades relativas à gestão destas bolsas, mais precisamente no Menu *Scholarships*, discutido mais a frente.

Caso a faculdade necessite de efetuar uma contratação urgente de algum funcionário (docente ou não docente), é dada preferência a pessoas que já tenham anteriormente trabalhado na instituição. Para tal, foi necessário manter um registo de todos os colaboradores – atuais e antigos – numa tabela de dispersão. Deste modo, foi preciso que todos estes trabalhadores (Employees) passassem a ter associado a si a respetiva situação de emprego.

Na eventualidade de ser necessário contratar alguém, pode-se agora facilmente consultar todos os antigos funcionários. A partir daqui é possível consultar todas as informações associadas a cada um destes e, após tomar a decisão de quem se quer contratar de volta, apenas é preciso indicar o seu nome e manter o salário anterior, ou então atribuir-lhe um novo, para que este seja efetivamente acrescentado à equipa. O recém-contratado funcionário fica agora disponível como empregado atual e, no caso dos professores, ficam ainda disponíveis para, tal como se fazia já na implementação anterior, associar a qualquer Departamento, Curso ou Unidade Curricular que se pretenda.

Classes utilizadas

College

Esta é a nossa classe principal. Nela guardamos um vetor com os departamentos, um vetor com os professores e um para os não docentes da faculdade e, portanto, contém as respetivas funções que permitem adicionar ou remover elementos desses mesmos vetores.

Para além destes vetores temos ainda uma Binary Search Tree (BST) com todos os alunos da faculdade, com a qual permitimos adicionar, remover, editar e apresentar todas as informações dos alunos da faculdade.

Para a implementação da BST criamos uma classe Student_Ptr, que contem um pointer pra um student como membro-dado, um construtor e uma função de comparação (<) para utilizar com os algoritmos de inserção, remoção e pesquisa da BST.

Já na implementação da Priority_Queue foi criada a classe Compara_St_Queue cujo o único elemento é uma função-membro que serve para ordenar os alunos dentro da *priority queue*.

Aqui encontra-se também uma Hash Table que facilita a gestão de todos os funcionários (docentes e não docente, atuais e antigos), sendo possível guardar as informações de todos os colaboradores que já trabalharam na faculdade. Estes são colocados na tabela a partir da *hash function* EmployeePtrHash que implementa o algoritmo dado nas aulas e apresenta uma taxa mínima de colisões.

Para além disto é nesta classe que temos o código de acesso do administrador e o nome da faculdade.

Esta classe foi usada como ponto central do nosso projeto, permitindo que através dela se possa aceder a toda a informação de forma mais eficaz.

People

A classe People é uma classe abstrata que tem como classes derivadas as classes Employee, Student, Teacher e Staff. Nesta classe temos todos os elementos que são comuns a todas essas pessoas da faculdade, tais como: nome, morada, número de telemóvel, código e data de nascimento.

Students

A classe Students possui os elementos que distinguem os alunos dos outros tipos de pessoas aqui tratadas, como por exemplo: o ano em que está inscrito, o seu curso e ainda um *map* que contém as unidades curriculares em que está inscrito e as notas em cada uma delas (se já avaliado).

Temos ainda uma *string* com o nome do curso que foi criada para facilitar o processo de leitura do ficheiro, permitindo assim guardar inicialmente apenas o seu nome, isto porque só após a criação dos cursos é possível fazer a correta correspondência entre esse objeto e essa string. Toda a informação destes são guardads na Binary Search Tree do College (mencionada anteriormente).

StudentsPtr

Esta nova classe constitui uma ligação entre a classe Student e o apontador usado na BST para a mesma.

O seu único membro dado é um *pointer* para a classe Student e tem também um *operator overloading* que permite organizar a mesma de acordo com os cursos, ou caso este seja o mesmo, por ordem alfabética de nome.

Employee

A classe Employee possui argumentos comuns a quem trabalha na faculdade, assim, tanto a classe Staff como a classe Teacher são suas derivadas.

Esta classe tem como parâmetros o salário do trabalhador, o seu NIF e, agora, um novo – *working* – para ser conhecida a sua situação de emprego na faculdade.

Teacher

Tal como na classe anterior, a classe Teacher também só possui parâmetros que o distinguem de uma outra pessoa e de um trabalhador, tais como a sua categoria e um vetor com as unidades curriculares que leciona.

Staff

Temos também a classe Staff cujo único parâmetro é a área de trabalho desse funcionário.

EmployeePtr

Por último, foi criada uma nova classe para fazer a gestão de todos os funcionários (atuais e antigos) na Hash Table employeeTable. Esta é *friend class* tanto da classe Teacher como da Staff, de forma a poder aceder a todos os seus métodos *private* e *protected*. Como membro-dado pode ter um apontador para Professor (Teacher*) ou um apontador para Funcionário Não-docente (Staff*) e, de forma a saber de qual dos dois se trata, um tipo (*type*) (que indica se é *teacher* ou *staff*). A sua colocação na tabela é feita com a *hash function* referida na classe College.

Lista de Casos de Utilização

Implementação do UI

O sistema de navegação de menus foi concebido com o objetivo de uma utilização fácil e intuitiva, realizando-se sempre por três partes:

Cabeçalho

Esta secção é delimitada por uma "caixa" onde estão presentes todas as informações do menu atual, desde a **página principal**, onde apenas o nome da **Faculdade** é mostrado, um **Departamento**, onde informações como morada e número telefónico também são acrescentadas, e até aos perfis de **Pessoas.**

Seleção de Menus

Seguidamente é nos mostrada uma lista numerada de opções possíveis, desde **listagens**, **procura** por nome, **edição** de atributos, etc. No final é nos sempre apresentada a opção de voltar ao menu anterior, e nalguns casos, de cancelar a funcionalidade atual.

User Input

Uma linha após a listagem de opções é deixado espaço para a escolha do utilizador, sempre verificada pela função criada para este fim.

Menus a salientar

Como é habitual, nem todas as funcionalidades têm a mesma complexidade e importância, portanto segue-se uma listagem e explicação de alguns dos menus e funções mais importantes no nosso programa, acrescentados nesta segunda parte do trabalho:

Menu Scholarships

Neste menu é-nos possível atribuir um número variável de bolsas **x** com um valor variável **n** (escolhido pelo utilizador). Ou seja, os **x** melhores alunos receberão uma bolsa com **n** valor. Para este efeito adicionamos um parâmetro a *priority queue*, o valor da bolsa de cada aluno, que faz com que os alunos com menos bolsa recebida fiquem a frente (para que as bolsas não sejam todas entregues sempre ao aluno com melhor média, já que a bolsa não altera o valor

da média).

Para além disso também nos é possível ver quem é o aluno que se encontra na frente da fila (aluno que receberá a próxima bolsa) e pagar propinas (retirar bolsas a alunos).

Menu Employees

Aqui é possível consultar todos os funcionários (tanto docentes como não docentes) separados apenas pela situação atual de emprego na faculdade.

- **Current Employees** (funcionários atuais) pode-se consultar as informações de cada um destes através da listagem apresentada ou procurando o seu nome.
- **Former Employees** (funcionários antigos) tal como na anterior, pode-se consultar as informações de cada um destes através da listagem apresentada ou procurando o seu nome. Para além disto, é possível contratar (**Hire Employee**) estes funcionários e atribuir (ou não) um novo salário.

Obstáculos

Tal como na primeira parte do trabalho deparamo-nos com algumas dificuldades, sendo que todas foram resolvidas e estamos muito satisfeitos com o resultado final.

Entre as quais podemos salientar:

- **Destrutores**

  Novamente, um dos problemas persistentes consistiu na implementação dos destrutores das classes, uma vez que, devido aos largo número de estruturas de dados com apontadores para objetos de outras classes, foi trabalhoso perceber se toda a memória já não utilizada estava a ser libertada e, por sua vez, corretamente.

- **Função de leitura (*load* do ficheiro)**

  Numa situação um pouco semelhante com a anterior, foi necessário perceber se tudo estava a ser colocado corretamente e no sítio pretendido.

- **BST desenvolvida com pointers**

  Pela primeira vez, decidimos criar uma Binary Search Tree que guarda a sua informação com pointers. Foi então mais complicado que o habitual perceber como a implementar e se a informação nela contida era alterada como pretendido.

- **Delete dos Employees**

  Para adicionar na Hash Table deixámos de poder libertar a memória contida nos pointers quando eliminávamos um dos professores. Para resolver esta questão, foi necessário implementar um novo algoritmo que verificava todos os Departamentos, Cursos e UCs para manualmente fazer a sua remoção completa da faculdade.

Distribuição de Tarefas

Tal como na Parte I deste projeto, conseguimos que as tarefas fossem distribuídas uniformemente por cada membro de acordo com as respetivas disponibilidades e pontos fortes.

Sendo um grupo de amigos, mantivemos um alto nível de comunicação entre todos, o que facilitou todo este processo. Cada membro foi responsável por uma das estruturas de dados, mas sempre que necessário havia alguma rotação, o que garantiu a familiarização de todos com a implementação das classes e funções principais.

A continuação da utilização de um repositório privado no **GitHub** permitiu uma mais rápida colaboração.

Conclusão

Como já referido, conseguimos atingir todos os nossos objetivos, com alguns obstáculos, mas que através de algum trabalho e muita cooperação foram ultrapassados.

Esta parte do projeto permitiu-nos aplicar as estruturas de dados lecionadas nas aulas teóricas e práticas, mas para além disso possibilitou-nos ter um maior à-vontade com conceitos já aplicados anteriormente, como por exemplo, *operator overloading* e tratamento de exceções.