Relatório Projeto Princípios de Programação Procedimental

Gestão da Base de Dados de Uma Universidade

2021/2022

UNIVERSIDADE D
COIMBRA

Carlos Miguel Matos Soares, Nº 2020230124

Docente: Noé Paulo Lopes Godinho

Regente: Maria José Patrício Marcelino

Introdução

Feito com base na cadeira de Princípios de Programação Procedimental (PPP), este projeto tinha o intuito de gerir as contas de um conjunto de alunos de uma Universidade. Deveria permitir a introdução de alunos, com um certo número de dados e, com base nas operações dos mesmos, introduzir despesas às contas dos mesmos.

Para começar o projeto da melhor maneira, começou-se por escolher as estruturas de dados mais adequadas.

Manual de Programador / Utilizador

Explicação do código implementado e escolhas feitas ao longo do programa.

No início do desenvolvimento da aplicação foram escolhidas as estruturas de dados a usar no programa. Depois de uma longa análise, em que a dúvida precedia sobre listas ligadas ou arvores binárias, foi escolhida a estrutura lista ligada, uma vez que para ordenamento de estudantes e inserção dos mesmos, a implementação era realizada de forma mais rápida.

Criaram-se duas estruturas, uma para o estudante e outra para a despesa, em que dentro do estudante era chamada a estrutura despesa, para a cada despesa estar associada uma lista de despesas.

De seguida foi criada uma base de dados em forma .txt, e, de seguida, realizada a função de leitura de dados. Em cada linha do ficheiro de dados estava a informação de cada estudante, com um argumento extra que é o número de despesas de cada aluno. Este argumento foi criado para tornar mais fácil a leitura de despesas, pois, com base nesse numero seria lido um certo numero de linhas e inserido nas despesas.

O ficheiro era aberto por leitura e registado na base de dados.

```
Gabriel,07-01-2000,1,2,3,380.00,1
10.00,Pear,20-10-2021
Pedro,12-12-2002,1,1,1,497.00,1
1.00,Apple,12-12-2022
Pinokio,10-03-2002,1,1,4,500.00,0
```

Realizando o menu, a cada nova funcionalidade criada era também criada a função que realizaria o pedido.

```
1. Add a new student.
2. Delete a student.
3. List all students alphabetically.
4. List students with balance below a certain amount (in descending order of balances).
5. Present all the information of a particular student.
6. Make an expense for a particular student.
7. Upload a student's account with a value
8. Exit
Your Choice:1
```

1. **Add a new student.**

Foi criada a função insertStudent() que adicionaria à lista um novo estudante, verificando antes se o mesmo já existe ou não na lista.

Era atualizado o número de estudantes, para mais tarde, na função de ordenar, o ciclo for percorrer o número correto de estudantes e não andar a percorrer zonas vazias para aumentar o tempo de execução.

2. **Delete a student.**

Passado como argumento os detalhes do aluno, esta função irá verificar se o aluno existe ou não, e, no caso de existir, daria free ao node em que estava inserido o estudante e era decrementado o número de estudantes.

3. **List all students alphabetically.**

Quando pressionada esta opção, é chamada a função printStudent() que dá print

a um estudante, porém, estando dentro de um ciclo for que percorreria o número de estudantes, daria print a todos.

Na mesma função era chamada uma outra função que ordenava, de forma alfabética, a lista de estudantes.

4. **List students with balance below a certain amount (in descending order of balances).**

Assim que pressionada a tecla número 4, o utilizador colocaria um limite em que, cm base no mesmo, eram listados todos os estudantes com o saldo abaixo desse limite, porém, de forma decrescente.

5. **Present all the information of a particular student.**

Quando recebida a escolha número 5, significa que o utilizador quer ver a informação de um estudante em especifico e então, envia, como argumento, os dados do aluno. Após a verificação da existência do mesmo, será listada a informação do mesmo, com a função printStudent().

6. Make an expense for a particular student.

Quando o utilizador deseja fazer uma compra para um estudante, coloca a informação do mesmo para a verificação da existência, e caso exista, é pedido ao utilizador que insira os detalhes da despesa, bem como valor, descrição e data da mesma.

Após a inserção dos dados é decrementado ao saldo do aluno o valor da compra.

7. Upload a student's account with a value.

Se o utilizador desejar depositar dinheiro na conta de um determinado aluno, irá inserir os dados do mesmo e, após verificação, o valor escolhido pelo utilizador será depositado na conta do aluno em questão.

Após o término do programa, será aberto para escrita o ficheiro de dados e inserido no mesmo o conteúdo das listas.

Controlos para inputs do utilizador.

No menu, local onde o utilizador irá inserir tudo que irá fazer o programa correr, e, consequentemente, é necessário proteger os valores inseridos pelo utilizador.

Quando uma das escolhas no menu não é válida, é pedido ao utilizador que insira uma outra, e, enquanto a escolha não for igual a 8, o menu continuará a ser mostrado.

Existe uma função que validará a data inserida, tendo em conta os dias, meses do ano e o ano em questão, verificando se é bissexto ou não, se é mês de 30 ou 31, fevereiro ou não, etc.

Os restantes valores inseridos para cada parâmetro do aluno apenas têm verificação de serem ou não dígitos, à exceção do saldo, em que é verificado também se o mesmo é negativo.

Quando inserido o valor de limite para o ordenamento de alunos com base no saldo é verificado se o limite é um valor positivo, uma vez que o saldo não pode ser negativo, como referido em cima.

No momento em que é realizada uma despesa a um aluno, é verificado se o mesmo possui, ou não, saldo suficiente para a realização da mesma.

Comentários sobre o projeto

Foi realizada uma função que verificaria se as datas de despesas eram possíveis, ou seja, era atribuída uma idade mínima para puder realizar uma despesa, porém, como a mesma estava a dar erro, foi apenas colocada em comentário.

A função insertStudent estava a dar erro quando era apenas verificado o número do aluno, sendo esse número único, porém, como a mesma dava erro, acabou por ser decidido passar todos os argumentos do estudante.

Conclusão

Apesar de um trabalho acessível, este era um projeto que ocupava um certo tempo e certos conhecimentos.

Devido à quantidade de projetos por fazer, tornou-se difícil conciliar tudo e, devido a esse facto, pode não ter sido implementado da melhor forma.