1 2 9 0

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

Manual do Utilizador

Amanda Oliveira de Menezes 2017124788

Princípios da Programação Procedimental
Licenciatura em Engenharia Informática
Junho, 2022

1. Introdução

Esta aplicação foi desenvolvida com o objetivo de administrar uma base de dados local contendo informações de estudantes registados. Um utilizador pode consultar informações e manipular dados como o saldo e a turma em que determinado estudantes está inscrito. A interface apresenta todas as ações possíveis ao conteúdo da base de dados que é organizada em um ficheiro de texto. Cada uma das funcionalidades estão descritas a seguir.

2. Descrição das funcionalidades

```
MENU
1- Introduzir dados de um(a) alunx
2- Eliminar um(a) alunx existente
3- Listar todxs xs alunxs
4- Listar xs alunxs com saldo abaixo de um determinado valor
5- Apresentar toda a informação de um(a) determinadx alunx
6- Efetuar uma despesa por um(a) determinadx alunx
7- Carregar a conta de um(a) alunx com um valor
8- Alteração de turma
9- Sair
```

Figura 1. Representação visual da interface da menu de administração.

1 – Introduzir dados de um novo estudante

Esta ação possibilita registar as informações de um aluno que não está presente na base de dados atual. Para isso, é necessário fornecer informações quanto ao nome, data de nascimento, turma em que está inscrito, número de estudante, ano em que está matriculado e saldo atual disponível.

Os dados são recolhidos com a função fgets() uma vez que utiliza o tamanho do buffer definido a evitar uso indevido de memória e lê o whitespace ao contrário da função scanf no caso de input de strings.

A função check_num() é responsável por verificar se um estudante já se encontra registado na base de dados da aplicação, e em caso positivo, envia uma mensagem de erro ao utilizador a dizer operação anulada.

Com todas as informações definidas é colocado um novo nó à lista ligada com a função append() que o acrescenta ao final da lista. A seguir as novas informações são acrescentadas à base de dados.

2 – Eliminar um estudante existente

Para a eliminação de um estudante registado o utilizador precisa inserir o número de respetivo estudante. Desta forma a função para deletar o nó percorre a lista ligada a

verificar se de facto este estudante está registado e em caso positivo é então apagada a informação, libertando a memória deste nó. A base de dados é então rescrita com a nova informação da lista ligada atualizada.

3 – Listar toda a informação de todos os estudantes registados

A listagem da lista ligada é feita em ordem alfabética e apresenta toda a informação contida na base de dados. Para a comparação de nomes com a presença de caracteres com acento ou sem ser símbolo na codificação ASCII recorre-se à biblioteca "lib-utf8" para a retirada destes caracteres e assim comparação das strings já modificadas.

4 – Listar os estudantes registados com um saldo inferior a um determinado valor

Para esta funcionalidade o utilizador precisa inserir o valor limite a ser consultado e desta forma são listadas todas as informações dos estudantes que possuem saldo inferior a este valor obtido de forma decrescente.

5 – Apresentar toda a informação de um determinado estudante

Nesta ação o utilizador pode consultar toda a informação registada de um estudante a partir do fornecimento de seu número de estudante . Caso não seja encontrado é mostrada na interface uma mensagem de erro a dizer que não foi possível localizar o estudante na base de dados.

6 – Efetuar uma despesa por um determinado estudante

Esta funcionalidade é responsável por gerar uma despesa associada a um estudante registado. Para isto é preciso que o utilizador forneça a informação quanto ao valor da despesa e o número do estudante. Assim, o estudante é procurado na lista ligada e caso não seja encontrado aparece uma mensagem de erro e também verifica-se se o saldo disponível é superior a despesa. Caso não seja também é gerada uma mensagem de erro. A base de dados é então atualizada com esta nova informação.

7 – Carregar a conta de um estudante com um determinado valor

Para carregar a conta de um estudante o utilizador precisa fornecer o valor a ser carregado e o número do estudante. Assim este valor será acrescido ao saldo atual e a nova informação será atualizada na base de dados.

8 – Alterar a inscrição de turma de um determinado estudante

No caso de alteração da turma em que um estudante está inscrito é preciso que o utilizador forneça o número do estudante e a turma em que será transferido. Assim a informação deste estudante é atualizada na base de dados.

9 – Sair da aplicação

Esta ação permite o utilizador sair do menu e finalizar o programa sem erros. Como a atualização da base de dados é garantida durante a alteração nas demais funcionalidades, não é preciso dar *refresh* das informações modificadas ou adicionadas.

3. Arquitetura da aplicação

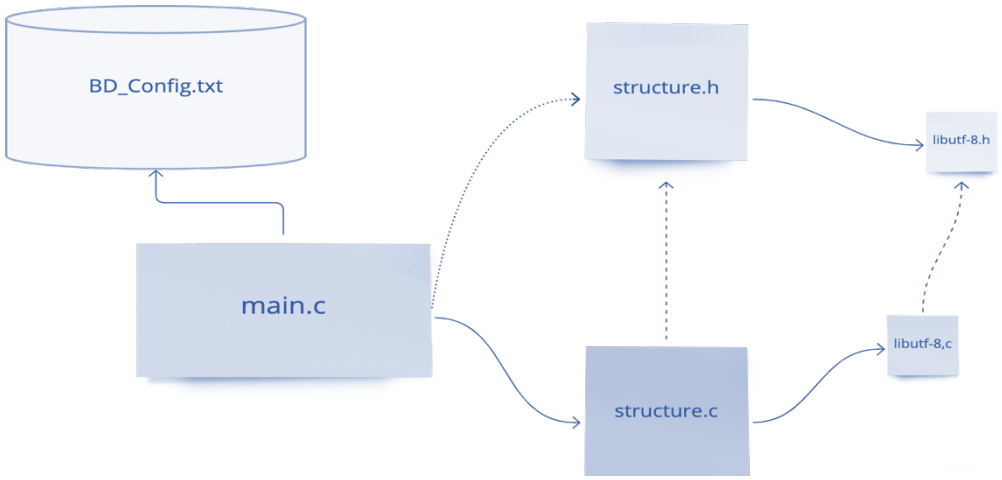

Figura 2. Arquitetura implementada à aplicação.

4. Estrutura de dados

A estrutura de dados utilizada foi de lista ligada que conecta nós através de uma estrutura de nó que contém associado o próximo nó da fila. A inserção é feita em forma de append e o novo nó criado é adicionado ao final da fila. A informação para adição de nós à lista é feita por referência e não por cópia, otimizando o custo de memória do programa.

Para o acesso aos nós contidos na fila é necessário passar como parâmetro das funções a referência da raíz (head) para assim percorrer enquanto um nó a null não é encontrado, atualizando o nó atual com o valor do seu próximo nó.

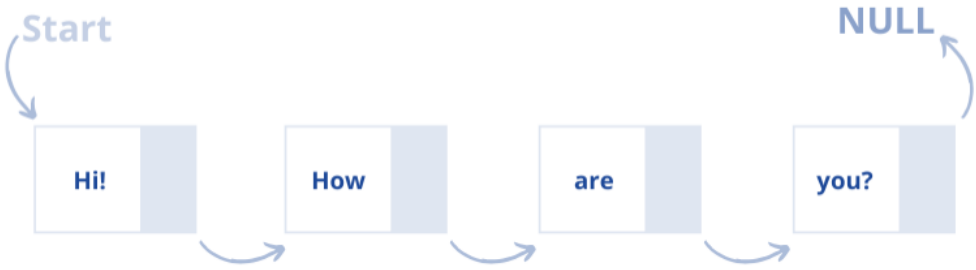

Figura 3. Representação ilustrativa do funcionamento de uma lista ligada.

5. Leitura e escrita do ficheiro “BD_Config.txt”

Em princípio toda a informação contida na lista ligada foi lida no ficheiro de texto da base de dados da aplicação. Para a leitura do ficheiro de texto é necessário recorrer à função fscanf() a ter em conta o caracter de quebra de linha para extrair o valor correto para cada instância da estrutura dos nós.

Para a escrita do ficheiro, no caso particular de se inserir um novo estudante antes não registado, é realizado o processo de append com a função implementada append() que insere este novo nó ao final da fila, sem ter em conta alguma ordenação particular. Desta forma, o novo estudante é então acrescentado ao ficheiro com o modo de escreta ‘a’ através da função writeToFileScreen () que no driver code (main) passa como parâmetro o ponteiro para a estrutura deste estudante.

Depois deste passo toda a fila é reescrita no ficheiro devido ao modo ‘w’, responsável por apagar a informação que já lá estava escrita e escrever tudo novamente. Esta ação é realizada pela função refreshBD() e devolve uma mensagem de erro caso não seja possível abrir o ficheiro.

6. Ordenação da lista ligada

A ordenação requerida foi alfabética para a apresentação dos estudantes registados. Para isso ´foi necessário recorrer ao algoritmo de ordenação Bubble Sort [2].

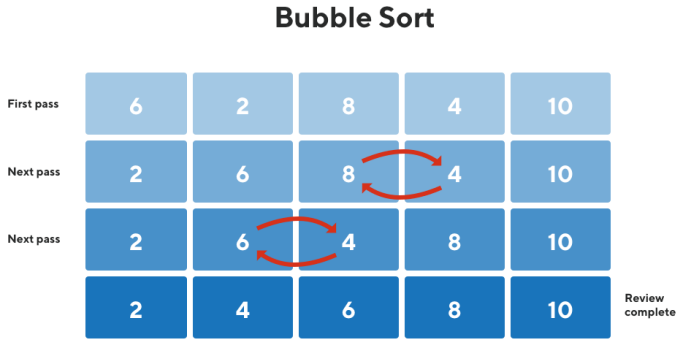

Figura 4. Ilustração do algoritmo Bubble Sort.

A partir da comparação das strings com a função strcmp() entre o nó atual o nó imediatamente a seguir foi possível identificar a ordem da lista. Para os casos em que as strings contém caracteres que não fazem parte dos caracteres ASCII como letras com acentos foi preciso utilizar a biblioteca “libutf-8” [1]. A função strtobase_u8() retira todos os acentos e cedilhas e passa tudo para minúsculas para permitir comparações e ordenações. Primeiramente coloca na string destino a string origem transformada, sabendo que a string destino tem de ter pelo menos o mesmo espaço que a string

origem e espera-se que a string origem esteja codificada em UTF-8 e cria a string destino também.

Após esta comparação o Bubble Sort altera a ordem dos nós com um nó auxiliar temporário (temp). O temp toma então os valores do nó atual, o nó atual toma os valores do próximo nó e por fim este nó seguinte toma os valores do nó temporário. A transferência de valores é realizada a partir da cópia dos mesmos e no caso de strings se utiliza a função strcpy().

Todo este passo é feito até se chegue ao final da lista, ou seja, até que o nó seguinte ao atual seja null. Para a ordenação dos saldos é feita de forma semelhante com a função bubbleSort_saldo() que também leva como parâmetro a referência do nó raíz da lista só que a comparação é feita em valores do tipo double entre saldos para os colocar em ordem decrescente.

7. Compilação do código fonte da aplicação

A compilação foi realizada utilizando as ferramentas -Wall, -Wextra e -Werror para assegurar que não houvesse nenhuma mensagem de erro ou aviso do compilador. O comando utilizado pode ser visualizado a seguir:

```
gcc -Wall -Wextra -Werror -o main main.c structure.c lib-utf8.c
```

8. Referências

[1] Silva, João Gabriel. “Biblioteca de funções para processar multibyte characters em C v2.1” (2021-04-25).

[2] BubbleSort Algorithm, disponível em: https://www.geeksforgeeks.org/bubble-sort/