

**PROFESSOR: Carlos Barrico, PHD | Frutuoso Silva, PHD | José Morgado, PHD  
|**

Bruno Monteiro, a43994,

Duarte Arribas, a44585.

Algoritmos e Estruturas de Dados

**Segundo semestre**

**ANO LETIVO: 2020-2021**

Última Modificação: 2021-05-22

**Engenharia Informática**

**Librorithm** (AED Trabalho Prático)

**Índice**

[Introdução 2](#_Toc72582984)

[Como compilar e correr? 3](#_Toc72582985)

[Explicação do programa 4](#_Toc72582986)

[Inicio 4](#_Toc72582987)

[Main Menu 4](#_Toc72582988)

[Sinais, saida do programa e correcao de memory leaks 4](#_Toc72582989)

[Prompt 5](#_Toc72582990)

[Separacao por ficheiros e documentacao do codigo 5](#_Toc72582991)

[Modularidade e globalizacao do programa 5](#_Toc72582992)

[Recuperabilidade 5](#_Toc72582993)

[File Menu 5](#_Toc72582994)

[Books Menu 7](#_Toc72582995)

[Clients Menu 8](#_Toc72582996)

[Orders Menu 9](#_Toc72582997)

[Operations Menu 9](#_Toc72582998)

[Algoritmos 11](#_Toc72582999)

[Merge Sort em linked lists 11](#_Toc72583000)

[Insertion Sort em Arrays 12](#_Toc72583001)

[Conclusão 14](#_Toc72583002)

[Bibliografia 15](#_Toc72583003)

# Introdução

Neste trabalho pretendemos construir uma aplicacao de gestao de uma biblioteca digital. Para tal, implementaremos diversas estruturas de dados e os seus algoritmos, de forma a tornar a aplicacao mais otimizada e rapida.

Objetivamos o conhecimento de conseguir aplicar estruturas de dados a um projeto real, de aplicacao real.

Este projeto armazena dados permantentemente em ficheiros e aplica diversos algoritmos para operacoes segundo clientes, livros e pedidos de uma biblioteca.

# Como compilar e correr?

Para compilar o programa, existe um makefile dentro da pasta principal **librorithm**. Numa shell de linux, correr o seguinte comando:

Em caso de corrupção, correr antes o comando:

e de seguida o primeiro.

Para correr o programa, simplesmente correr:

# Explicação do programa

## Inicio

### Main Menu

Inicialmente, o programa começa com o menu principal e uma prompt:

Text

Description automatically generated

O menu inicial contem todas as opcoes descritas no enunciado, numeradas de , e o para sair do programa.

### Sinais, saida do programa e correcao de memory leaks

Fizemos uma detecao de sinais de sistema, nomeadamente o **interrupt signal** (quando o ctrl-c e` premido) e o **quit signal** (quando o ctrl-q e` premido); fizemo-lo de modo a que o programa abortasse normalmente, para que conseguisse colocar um **clean handler** aquando da saida do programa.

Neste clean handler, a memoria dinamicamente alocada e` libertada e os buffers resetados a **NULL**, e uma mensagem com os nossos dados de autoria e` mostrada:

Text

Description automatically generated

Tivemos o cuidado de limpar quaisquer *memory leaks* que houvessem no programa, utilizando as flags de compilacao no compilador **gcc**:

Limpamos ainda quaisquer buffers necessarios, e fechamos todos os ficheiros previamente abertos.

### Prompt

A **prompt** e` dada por , onde user e` o nome de utilizador da shell utilizada (no meu caso YozNacks).

### Separacao por ficheiros e documentacao do codigo

Num aparte, utilizamos ficheiros .c e .h para organizacao do nosso projeto. Os ficheiros .h apenas conte^m prototipos de funcoes, definicoes e estruturas. E` de sublinhar que todos os ficheiros .h estao protegidos por *include guards*, para que nao haja conflitos de compilacao, aquando da chamada recursiva de ficheiros .h. Os ficheiros .c apenas conte`m *includes* de ficheiros .h de modo a seguir o desenvolvimento moderno de aplicacoes em C.

Realcamos que todo o nosso codigo esta estruturadamente comentado, isto e`, todas as funcoes encontram-se comentadas de acordo com as docs de programas de geracao de documentacao automatica, como o **doxygen**[1]; alem disso, temos pequenos comentarios de organizacao de codigo, assim como de realce de pequenos detalhes.

### Modularidade e globalizacao do programa

Tentamos ao maximo, dividir o nosso codigo em funcoes, de modo a seguir o principio de “cada funcao faz uma e uma so coisa”[2], e, portanto, temos inumeras funcoes, que, em conjunto, constitue`m um software inteiro.

A linguagem das variaveis/comentarios esta` maioritariamente em Ingles, visto que e` uma linguagem global, e achamos no nosso interesse tornar o programa mais acessivel.

Tentamos ainda, seguir alguns dos principios de IHC[3], para poder tornar o programa mais *user friendly*.

Realcamos ainda que, otimizamos o nosso codigo de modo a tornar o codigo mais rapido.

### Recuperabilidade

Queremos ainda sublinhar que escrevemos intensivamente codigos de detecao de erros para tornar o programa **recuperavel**.

## File Menu

Neste menu, o utilizador podera criar/abrir/guardar os dados da biblioteca:

Text

Description automatically generated

A primeira opcao e` utilizada para a criacao de novas estruturas de dados dentro do programa (ao, por exemplo, destruir a lista dos clientes).

A segunda opcao e` utilizada para apagar todos os dados dentro dos ficheiros que foram, previamente, guardados, e que estao a espera de serem carregados para o programa.

A terceira opcao e` utilizada para ler todos os dados dos ficheiros de cada estrutura de dados do programa (ignorando ficheiros vazios). Note-se, porem, que esta chama a primeira opcao, isto e`, apaga primeiro as estruturas de dados, e apenas depois le os dados dos ficheiros.

A quarta opcao e` utilizada para guardar todos os dados das estruturas de dados em memoria (do programa) para os ficheiros (de modo a torna-los permantentes). Note-se, porem, que esta chama a segunda opcao, isto e`, apaga primeiro os dados dos ficheiros, e apenas depois guarda dos dados.

A opcao volta para o menu principal.

Text

Description automatically generated

## Books Menu

Neste menu, o utilizador podera inserir/remove/alterar/consultar os livros da biblioteca (isto acontece ao nivel de memoria do programa, isto e`, nas estruturas de dados do programa):  
Text

Description automatically generated

Note-se que estes livros estao guardados numa arvore binaria equilibrada.

A primeira opcao pede os dados do livro ao utilizador, denotando-se que todos eles sao verificados de acordo com erros de input ao nivel de sintaxe e logica (e.g., inserir um ISBN com mais ou menos de digitos). Caso o input seja , a insercao do novo livro e` interrompida. Apos a insercao do livro caso ainda nao exista, verifica-se se a arvore continua equilibrada, equilibrando-a caso nao esteja.

A segunda opcao remove um livro da estrutura de dados, dado o seu ISBN. Mais uma vez, verifica-se se a arvore continua equilibrada, equilibrando-a caso nao esteja.

A terceira opcao abre um segundo menu para alterar os livros:

Text

Description automatically generated

Neste submenu, a primeira opcao e` utilizada para alterar as caracteristicas de um dado livro (pelo ISBN), enquanto a segunda e` utilizada para repor o stock para uma dada quantia, a um dado livro (pelo ISBN). A opcao volta para o menu dos livros.

A quarta opcao abre um terceiro menu para consultar os livros, segundo uma dada caracteristica:  
Text

Description automatically generated

Neste submenu, a primeira opcao mostra o livro com o dado ISBN, a segunda opcao mostra os livros que contenham a substring dada no seu titulo, a terceira opcao mostra os livros de acordo com o primeiro autor e ano, e a quarta opcao mostra todos os livros. A opcao volta para o menu dos livros.

A opcao do menu dos livros volta para o menu principal.

## Clients Menu

Neste menu, o utilizador podera inserir/remove/alterar/consultar os clientes da biblioteca (isto acontece ao nivel de memoria do programa, isto e`, nas estruturas de dados do programa):

Text

Description automatically generated with medium confidence

Note-se que estes clientes estao guardados numa lista ligada simples.

A primeira opcao pede os dados do cliente ao utilizador, denotando-se que todos eles sao verificados de acordo com erros de input ao nivel de sintaxe e logica (e.g., inserir um NIF com mais ou menos de digitos). Caso o input seja , a insercao do novo cliente e` interrompida. Note-se que o array das orders dos utilizadores e` inicializado a NULL, visto que ainda nenhum pedido foi feito para esse cliente.

A segunda opcao remove um cliente da estrutura de dados, dado o seu NIF.

A terceira opcao e` utilizada para alterar os dados de um cliente. Se o dado introduzido for , o dado fica igual ao que ja esta`.

A quarta opcao abre um terceiro menu para consultar os clientes, segundo uma dada caracteristica:

Text

Description automatically generated  
Neste submenu, a primeira opcao mostra o cliente com o dado NIF, a segunda opcao mostra os clientes com um dado nome, a terceira opcao mostra os clientes que tenham uma dada morada, e a quarta opcao mostra todos os clientes. A opcao volta para o menu dos clientes.

A opcao do menu dos clientes volta para o menu principal.

## Orders Menu

Neste menu, o utilizador podera inserir/remove pedidos (isto acontece ao nivel de memoria do programa, isto e`, nas estruturas de dados do programa):

Graphical user interface, text

Description automatically generated

Note-se que estes pedidos estao guardados numa fila.

A primeira opcao pede todos os dados requisitados por um pedido, verificando a sintaxe e logica dos inputs (incluindo verificacao de stock, e.g.).

A segunda opcao remove o pedido do inicio da fila, adicionando-o ao array das orders do respetivo cliente.

## Operations Menu

Neste menu, o utilizador possui uma escolha entre as operacoes necessarias do enunciado, assim como novas. A opcao sai para o menu principal:

Text

Description automatically generated

A primeira opcao mostra a quantidade de livros vendidos numa data especifica.

A segunda opcao mostra a data mais recente de um dado ISBN.

A terceira opcao mostra a quantidade de livros de um dado cliente (dado um NIF).

A quarta opcao mostra livros mais recentes duma dada area cientifica.

A quinta opcao mostra mostra os best-selling books.

A sexta opcao mostra a area cientifica com mais livros.

A setima opcao mostra o cliente com mais livros comprados.

A oitava opcao mostra os clientes por numero de compras por ordem decrescente.

A nona opção mostra o ano com mais publicações.

A décima opção mostra o cliente que gastou mais dinheiro.

A décima primeira opção verifica o desperdício de memória em todas as estruturas de dados.

A décima segunda opção mostra um cliente, no qual o seu nome começa com um dado char.

A décima terceira opção indica o número de pedidos que faltam ser satisfeitos.

A décima quarta opção indica qual é o livro mais dispendioso.

A décima quinta opção indica o número de livros existentes na árvore binária de pesquisa equilibrada e o valor total desse mesmo stock.

# Algoritmos

Tal como referido no enunciado, utilizamos dois algoritmos novos para alem de todos os dados nas aulas.

## Merge Sort em linked lists

Tentamos implementar o merge sort para a lista ligada dos clientes[4]:



Para esta implementacao recursiva, tivemos que utilizar duas outras funcoes de auxilio, uma para dividir a lista em duas e a outra para ajudar na juncao da parte esquerda com a direita da lista.





Foi utilizada na funcao de mostrar os clientes de forma decrescente do numero de compras (ordenar e printar recursivamente por ordem inversa).

Este e` um algoritmo muito eficiente, tendo uma ordem de complexidade maioritariamente melhor que a do **quicksort**[5].

## Insertion Sort em Arrays

Como foi pouco utilizado, utilizamos um algoritmo mais facil de implementar, mas mais eficiente dos mais usuais (e.g., bubble sort):



Foi utilizado na ordenacao do array dos livros mais vendidos de acordo com o seu ISBN.

# Conclusão

Em suma, este trabalho trouxe um vasto conhecimento sobre a aplicacao de estruturas de dados e seus algoritmos sob contexto de uma aplicacao/projeto real.

Aprendemos imenso sobre a organizacao de um projeto, na divisao de tarefas e trabalho colaborativo entre os membros do grupo.

Aprendemos ainda sobre a aplicacao de estruturas de dados em contexto de estruturas diferentes e com aplicacao de novos algoritmos, que ajudam a concretizar as operacoes requiridas e nao requiridas no enunciado.

Queremos, finalmente, destacar que estes conhecimentos nos servirao bastante no futuro para quaisquer outro tipos de projetos onde performance e` um fator importante, e qual a melhor maneira de os aplicar.

# Bibliografia

[1] “Doxygen.” https://www.doxygen.nl/index.html.

[2] R. C. Martin, *Clean Code*. .

[3] A. Dix, *Human-Computer Interaction*. .

[4] “Merge Sort for Linked Lists - GeeksforGeeks.” https://www.geeksforgeeks.org/merge-sort-for-linked-list/ (accessed Apr. 20, 2021).

[5] “QuickSort vs MergeSort - GeeksforGeeks.” https://www.geeksforgeeks.org/quick-sort-vs-merge-sort/ (accessed Apr. 20, 2021).