

Algorítmos e Estruturas de Dados

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

Bruno Monteiro, a43994,

Duarte Arribas, a44585.

**Segundo semestre**

**ANO LETIVO: 2020-2021**

Última Modificação: 2021-05-22

**Engenharia Informática**

**Librorithm** (AED Trabalho Prático)

**Índice**

[Introdução 2](#_Toc72584788)

[Como compilar e correr? 3](#_Toc72584789)

[Explicação do programa 4](#_Toc72584790)

[Inicio 4](#_Toc72584791)

[Main Menu 4](#_Toc72584792)

[Sinais, saida do programa e correcao de memory leaks 4](#_Toc72584793)

[Prompt 5](#_Toc72584794)

[Separacao por ficheiros e documentacao do codigo 5](#_Toc72584795)

[Modularidade e globalizacao do programa 5](#_Toc72584796)

[Recuperabilidade 5](#_Toc72584797)

[File Menu 5](#_Toc72584798)

[Books Menu 7](#_Toc72584799)

[Clients Menu 8](#_Toc72584800)

[Orders Menu 9](#_Toc72584801)

[Operations Menu 9](#_Toc72584802)

[Algoritmos 11](#_Toc72584803)

[Merge Sort em linked lists 11](#_Toc72584804)

[Insertion Sort em Arrays 12](#_Toc72584805)

[Conclusão 14](#_Toc72584806)

[Bibliografia 15](#_Toc72584807)

# Introdução

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

Objetivamos o conhecimento de conseguir aplicar estruturas de dados a um projeto real, de aplicação 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 contém todas as opções descritas no enunciado, numeradas de , e o para sair do programa.

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

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

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

Text

Description automatically generated

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

Limpámos ainda quaisquer buffers necessários, e fechamos todos os ficheiros previamente abertos.

### Prompt

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

### Separação por ficheiros e documentação do código

Num aparte, utilizamos ficheiros .c e .h para organização do nosso projeto. Os ficheiros .h apenas contêm protótipos de funções, definições e estruturas. É de sublinhar que todos os ficheiros .h estão protegidos por *include guards*, para que não haja conflitos de compilação, aquando da chamada recursiva de ficheiros .h. Os ficheiros .c apenas contém *includes* de ficheiros .h de modo a seguir o desenvolvimento moderno de aplicações em C.

Realçámos que todo o nosso código esta estruturadamente comentado, isto é, todas as funções encontram-se comentadas de acordo com as docs de programas de geração de documentação automática, como o **doxygen**[1]; além disso, temos pequenos comentários de organização de código, assim como de realce de pequenos detalhes.

Gerámos, pois, a documentação automatica com o doxygen, que pode ser encontrada na pasta **docs** do projeto.

### Modularidade e globalizacao do programa

Tentámos ao máximo, dividir o nosso código em funções, de modo a seguir o princípio de “cada função faz uma e uma só coisa”[2], e, portanto, temos inúmeras funções, que, em conjunto, constituem um software inteiro.

A linguagem das variáveis/comentários está maioritariamente em Ingles, visto que é uma linguagem global, e achámos no nosso interesse tornar o programa mais acessível.

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

Realçámos ainda que, otimizamos o nosso código de modo a torná-lo mais rápido.

### Recuperabilidade

Queremos ainda sublinhar que escrevemos intensivamente códigos de deteção de erros para tornar o programa **recuperável**.

## File Menu

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

Text

Description automatically generated

A primeira opção é utilizada para a criação de novas estruturas de dados dentro do programa (ao, por exemplo, destruir a lista dos clientes).

A segunda opção é utilizada para apagar todos os dados dentro dos ficheiros que foram, previamente, guardados, e que estão à espera de serem carregados para o programa.

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

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

A opçãoo volta para o menu principal.

Text

Description automatically generated

## Books Menu

Neste menu, o utilizador poderá inserir/remove/alterar/consultar os livros da biblioteca (isto acontece ao nível de memória do programa, isto é, nas estruturas de dados do programa):  
Text

Description automatically generated

Note-se que estes livros estão guardados numa árvore binaria equilibrada.

A primeira opção pede os dados do livro ao utilizador, denotando-se que todos eles são verificados de acordo com erros de input ao nivel de sintaxe e lógica (e.g., inserir um ISBN com mais ou menos de digitos). Caso o input seja , a inserção do novo livro é interrompida. Após a inserção do livro caso ainda não exista, verifica se a arvore continua equilibrada, equilibrando-a caso não esteja.

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

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

Text

Description automatically generated

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

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

Description automatically generated

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

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

## Clients Menu

Neste menu, o utilizador poderá inserir/remove/alterar/consultar os clientes da biblioteca (isto acontece ao nível de memória do programa, isto é, nas estruturas de dados do programa):

Text

Description automatically generated with medium confidence

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

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

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

A terceira opção é utilizada para alterar os dados de um cliente. Se o dado introduzido for , o dado fica igual ao que já está.

A quarta opção abre um terceiro menu para consultar os clientes, segundo uma dada característica:

Text

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

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

## Orders Menu

Neste menu, o utilizador poderá inserir/remover pedidos (isto acontece ao nível de memória do programa, isto é, nas estruturas de dados do programa):

Graphical user interface, text

Description automatically generated

Note-se que estes pedidos estão guardados numa fila.

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

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

## Operations Menu

Neste menu, o utilizador possui uma escolha entre as operações necessárias do enunciado, assim como novas. A opção sai para o menu principal:

Text

Description automatically generated

A primeira opção mostra a quantidade de livros vendidos numa data específica.

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

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

A quarta opção mostra livros mais recentes duma dada área científica.

A quinta opção os best-selling books.

A sexta opção mostra a área científica com mais livros.

A setima opção mostra o cliente com mais livros comprados.

A oitava opção mostra os clientes por número 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 além 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 implementação recursiva, tivemos que utilizar duas outras funções de auxílio, uma para dividir a lista em duas e a outra para ajudar na junção da parte esquerda com a direita da lista.





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

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

## Insertion Sort em Arrays

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



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

# Conclusão

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

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

Aprendemos ainda sobre a aplicação de estruturas de dados em contexto de estruturas diferentes e com aplicação de novos algoritmos, que ajudam a concretizar as operações requiridas e não requiridas no enunciado.

Queremos, finalmente, destacar que estes conhecimentos nos servirão bastante no futuro para quaisquer outro tipos de projetos onde performance é 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).