

Universidade do Minho
Escola de Engenharia
Mestrado Integrado em Engenharia de Telecomunicações e Informática
Métodos de Programação II

Trabalho Prático 2: Apontadores e Listas Ligadas

Gestão de um armazém

Bruno Xavier Brás dos Santos A72122

Ana Catarina Gonçalves da Cunha A85492

Guimarães
26 de Maio de 2018

Resumo

“There is no reason for any individual to have a computer at home”¹

O presente relatório divide-se em vários capítulos, cada capítulo está repartido em secções, e por cada secção existe algumas subsecções. A organização dos capítulos está baseada da seguinte forma:

No primeiro capítulo colocamos a introdução.

No segundo capítulo colocamos uma descrição informal do projeto, os dados mais relevantes e os inputs/outputs do nosso programa.

O terceiro capítulo descreve de forma sucinta como implementamos a solução, bem como as estruturas de dados usada e como efetuamos algumas das funções mais complexas do nosso projeto.

No quarto capítulo, colocamos alguns testes e resultados do programa.

No quinto capítulo colocamos a conclusão e concluimos acerca dos objetivos traçados na introdução.

No último capítulo colocamos as referências bibliográficas dos conteúdos que nos ajudaram a realizar este projeto.

A última parte do relatório são os apêndices. No apêndice A, e único, colocamos a implementação do código em linguagem C.

¹<https://www.theguardian.com/technology/2011/feb/09/ken-olsen-obituary>

Conteúdo

| | | |
|----------|---|-----------|
| 1 | Introdução | 2 |
| 2 | Análise e Especificação | 3 |
| 2.1 | Descrição informal do problema | 3 |
| 2.2 | Especificação dos Requisitos | 3 |
| 2.2.1 | Dados | 3 |
| 2.2.2 | Inputs/Outputs do programa | 4 |
| 3 | Concepção/desenho da Resolução | 5 |
| 3.1 | Estruturas de Dados | 6 |
| 3.2 | Algoritmos e Funções utilizadas | 6 |
| 4 | Codificação e Testes | 9 |
| 4.1 | Alternativas, Decisões e Problemas de Implementação | 9 |
| 4.2 | Testes realizados e Resultados | 9 |
| 4.3 | Estrutura do Código e ficheiros utilizados | 12 |
| 5 | Conclusão | 14 |
| 6 | Bibliografia | 15 |
| A | Código do Programa | 16 |

Capítulo 1

Introdução

O presente relatório foi realizado para um projeto da unidade curricular de métodos de programação II do 1º ano do curso de MIETI da Universidade do Minho, sendo este o segundo projeto realizado. O primeiro projeto foi realizado com base em estruturas e arrays. Neste projeto o objetivo é a utilização de estruturas, listas ligadas, memória dinâmica e apontadores.

Os nossos objetivos para este projeto são: aplicar os conceitos abordados na sala de aula, adquirir metodologias de trabalho e elaborar um programa que seja eficiente, que cumpra os requisitos propostos e que possua uma interação com o utilizador dinâmica e realística.

O projeto consiste na gestão de um armazém. Da leitura do enunciado são várias as operações que o nosso programa tem de realizar, desde a inserção de um artigo ao efetuar uma encomenda. Todas as questões retratadas no enunciado encontram-se em detalhado na secção "Descrição informal do problema".

Para o projeto, tivemos que implementar um programa escrito na linguagem de programação C e elaborar um relatório referente ao mesmo.

Para a elaboração do presente relatório, optamos por utilizar o software \LaTeX , tal como feito no primeiro projeto.

Capítulo 2

Análise e Especificação

2.1 Descrição informal do problema

Para a resolução deste trabalho prático, provimos de um enunciado de um projeto que o tivemos que resolver. Este trabalho prático tem como objetivo a gestão de um armazém. O ramo de negócio e a caracterização de cada produto em stock ficou ao critério de cada grupo. O nosso grupo optou por escolher um ramo de negócio que fosse abrangente, no sentido que os produtos em stock no armazém fossem variados. Para tal, o foco do nosso negócio são moveis e artigos de decoração. Os produtos em stock podem ser qualquer tipo de móveis, desde mesas e cadeiras a colchoes e espelhos, e qualquer artigo de decoração, desde jarros a quadros. Do enunciado do problemas retiramos que devem ser várias as operações a realizar sobre os produtos em stock. Elas são: (i) carregar (ler os dados relativos a cada produto no armazém, os quais serão inseridos direta e manualmente no terminal pelo operador); (ii) apagar (remover um produto do armazém); (iii) ver (visualizar os dados dum certo produto do armazém ou uma listagem global de todos) e (iv) manter (corrigir um ou mais dados); (v) ler encomendas, uma de cada vez, inseridas manualmente pelo operador no terminal; (vi) verificar se a encomenda é exequível, ou seja, se há em stock a quantidade suficiente de cada item; (vii) atualizar o stock; (viii) calcular o valor total da encomenda. Para não perder os dados dos produtos em armazém entre as sessões de trabalho, as informações contidas na lista ligada (sobre os produtos em stock), devem ser guardadas num ficheiro de texto sequencial. Para tal, devem escrever os dados no ficheiro com a função `fprintf`. No início de uma nova sessão esses dados devem ser lidos com `fscanf` e `fgets` para reconstruir a lista ligada. Neste trabalho prático, e como era o segundo trabalho prático da unidade curricular, para a sua resolução, foi privilegiado a utilização de estruturas, lista ligadas, ficheiros sequencias, apontadores e memória dinâmica. Pois, foram estes os conceitos que abordamos na segunda parte do semestre nas aulas.

2.2 Especificação dos Requisitos

2.2.1 Dados

Os dados do nosso projeto são os produtos em stock no armazém. Cada artigo é caracterizado por um nome, uma referência única, um preço e a quantidade existente em stock. Estes quatros parâmetros formam o ID de um artigo. O nome de um artigo corresponde a uma string que nos dá três informações acerca do artigo: gama do artigo, breve descrição (se é uma cadeira ou mesa, por exemplo) e a sua cor. Por exemplo, para o seguinte nome de um artigo "GERESTA clch mol 160x200 firme branco": "GERESTA" é a gama do artigo; "colch mol 160x200 firme" é uma breve descrição; "branco" é a cor. Sendo o artigo em questão um colchão(clch) de molas, de medidas 160x200 branco e cuja gama é denominada por GERESTA. A Referencia de um artigo é um número superior a 99. Esse número é único e permite identificar um artigo. Assim, como uma referencia só identifica apenas um artigo as operações no nosso armazém serão realizadas pela referencia do artigo. Para o exemplo acima com o colchão GERESTA, a sua referência é 145. O preço corresponde ao valor de um determinado artigo. É representado por um número de vírgula flutuante(número real) e a unidade monetária

utilizada foi o euro. No exemplo acima, o preço de cada colchão GERESTA é de 320,00 euros. A quantidade corresponde ao número de artigos que existem em stock no armazém. É um numero inteiro. No exemplo acima, a quantidade de colchões de referência 145 é de 11 unidades. Sendo assim, e de forma , a referencia 145 identifica um colchão de molas branco cujo preço de cada colchão é de 320,00 euros. Em stock no armazém existem 11 unidades.

2.2.2 Inputs/Outputs do programa

Os inputs do nosso programa podem ser cada um dos quatro parâmetros que identifica um artigo e os números que identifica cada operação que pretende que seja efetuada sobre dos artigos. Os outputs (saídas) correspondem ao resultado de cada operação sobre os artigos. Por exemplo, podem ser: a visualização de um artigo, o resultado de fazer uma encomenda ou a visualização da lista de todos os artigos armazenados no armazém.

Capítulo 3

Concepção/desenho da Resolução

O nosso objetivo para este projeto foi implementar um programa que cumpria os requisitos propostos, adicionamos mais algumas operações para que o nosso armazém fosse o mais completo possível e que possuísse uma boa interface . Assim, qualquer utilizador pode fazer usufruto do programa sem qualquer ajuda.

No início da implementação, definimos como cada artigo era representado. Como vimos anteriormente, um artigo é definido por uma referencia, um nome, o preço e a quantidade. Para agrupar estes quatro parâmetros na mesma variável, definimos uma estrutura e denominamos por ID ARTIGO. Na secção seguinte é explicado cada uma das estruturas concebidas no projeto.

Definimos dois novos tipos de dados para representar as listas ligadas, LLista e LEncomendas. LLista é a lista ligada para onde todos os artigos do armazém serão carregados de um ficheiro ou inseridos pelo utilizador. LEncomendas representa a lista ligada com as encomendas já realizadas.

A lista ligada com os artigos, LLista, é ordenada por ordem crescente do número de referencia de cada artigo. Assim, é facilitado o processo de inserção, remoção e procura. A lista com as encomendas feitas é ordenada por ordem crescente do número da encomenda. Para fazer uma encomenda, adicionamos os artigos que pretendemos na encomenda e a quantidade. Depois de adicionar todos os artigos que pretendemos, é calculado o número de artigos que fazem parte da encomenda, o número de referencias e o valor total. De seguida é gerado aleatoriamente um número que representará a encomenda. O número da encomenda tem como objetivo facilitar a procura da encomenda mais tarde. Na lista ligada das encomendas, cada membro da lista (cada encomenda) contém o seu numero, o número total de artigos, o número total de referencias e o valor total da encomenda.

Implementamos um menu inicial, onde o utilizador pode interagir com o programa e usufruir de todas as operações que são permitidas. É permitido: (i) inserir um artigo no armazém; (ii) remover um artigo do armazém; (iii) alterar o seu nome, referencia, preço e quantidade; (iv) ver informações de uma dada referencia; (iv) ver a lista de todos os artigos que existem no armazém; (v) fazer uma encomenda; (vi) ver número total de artigos e de referencias que existem numa encomenda; (vii) ver valor total de uma encomenda; (viii) ver lista com as últimas encomendas efetuadas; (iv) ver número de artigos e referências que existem no armazém.

Quando iniciamos o programa, é pedido se pretende inicializar o programa(instalar), ou se pretende recuperar os dados de uma secção de trabalho anterior. Se for esse o caso, é carregado nas duas listas ligadas os dados que estão em dois ficheiros, um ficheiro contém os dados de todos os artigos do armazém, no outro é armazenado todas as encomendas efetuadas. Assim é possível utilizar os dados de uma secção anterior. Se a opção escolhida for a de inicializar o programa, então é criado dois novos ficheiros que têm como função guardar os dados contidos nas duas listas ligadas.

Depois de cada interação no menu inicial, é guardado automaticamente nos ficheiros as informações que estão contidas nas duas listas ligadas.

3.1 Estruturas de Dados

Para este projeto implementados três novos tipos de dados. Na figura 3.1 podemos ver o código das respectivas definições dos novos tipos de dados.

```
typedef struct IDartigo{
    char nomeArtigo[N_ARTIGO];
    int referencia;
    float preco;
    int quantidade;
} ID_ARTIGO;

typedef struct lligada{
    ID_ARTIGO artigo;
    struct lligada *prox;
}*LLista;

typedef struct encomendas{
    int numeroEncomenda;
    float precoTotal;
    int numArtigos;
    int numReferencias;
    struct encomendas *prox;
}*LEncomendas;
```

Figura 3.1: estruturas de dados usado no projeto

O primeiro tipo de dados definido, "ID Artigo", é definido através de uma estrutura com 4 membros. Esta estrutura representa a identificação de um artigo. O primeiro membro é uma string com o nome do artigo, o segundo membro é um inteiro que representa a referência do artigo, o terceiro membro é um float que corresponde ao preço e o último membro é representa a quantidade de artigo que existe em stock. Através deste novo tipo de dados, conseguimos definir a representação de uma artigo.

O segundo tipo de dados definidos, "LLista", é definido como um apontador para a estrutura "struct lligada". Este novo tipo representa a lista ligada que irá conter todos os artigos armazenados no armazém. Como membros desta estrutura temos uma variável do tipo "ID ARTIGO", explicado no ponto anterior, que é uma estrutura, e uma variável do tipo apontador para uma variável do tipo "struct lligada". Ou seja, como membros temos uma estrutura que contém todas as informações de um dado artigo, e um apontador que tem como objetivo armazenar o endereço do elemento seguinte da lista ligada, que é um elemento do tipo "struct lligada".

O terceiro tipo definido, "LEncomendas", é definido como um apontador para a estrutura "struct encomendas", que representa a lista ligada com as encomendas efetuadas. Este novo tipo é definido por uma estrutura que tem, como membros, um inteiro que corresponde ao número da encomenda, uma variável do tipo float que representa o valor total da encomenda, um inteiro que corresponde ao numero total de artigos que existem na encomenda, um inteiro que corresponde ao número total de referencias da encomenda e uma variável do tipo apontador para uma variável do tipo "struct encomendas", que tem como objetivo armazenar o endereço do seguinte elemento da lista ligada, ou "NULL" caso este elemento é o último da lista ligada.

3.2 Algoritmos e Funções utilizadas

Esta secção tem como objetivo demonstrar o funcionamento das funções mais importantes no nosso projeto. A declaração (protótipo) das funções definidas encontram se no ficheiro "TipoDadosE-DeclaracaoFuncoes.h". Este ficheiro encontra-se na última secção deste projeto.

```
int inserirInformacaoArtigo(LLista l,int *r, char *n, float *p, int *q)
```

Esta função recebe uma cópia do endereço do primeiro elemento da lista ligada l que contém todos os artigos do armazém, recebe o endereço da variável que representa a referência de uma artigo, uma string, o endereço da variável que contém o preço de um artigo e o endereço da variável que contém a quantidade do artigo. Esta função tem como objetivo inserir as informações de um artigo(nome,

referência, preço e quantidade). A função recebe a lista de todos os artigos para, se o artigo que inserirmos já existir no armazém, não inserir um artigo repetido. Se a referência já existir, mostra a respetiva mensagem e pergunta se pretende adicionar mais artigos a essa referência. A instrução "fflush(stdin)" temo como objetivo limpar o buffer do teclado. Quando implementamos esta função, quando escrevíamos a referência do artigo, o output do programa saltava para a parte de escrever o preço do artigo, pois, interpretava a tecla "enter" como um carácter e não conseguíamos escrever o nome de um artigo. Esta função tem dois valores de retorno, 1 ou 2. Retorna 1 se o artigo introduzido já existir no armazém, 2 se tal artigo não existir. Estes valores de retorno têm como função ajudar a controlar o fluxo do programa e na resolução de possíveis erros que podem aparecer.

```
int procurarNomePrecoQuantidade_porReferencia(LLista,int,char *,float *,int *)
```

O objetivo desta função é, ao inserirmos um dada referência, o utilizador sem precisar inserir o nome, a quantidade e o preço, ser mostrado no ecrã essas informações do respetivo artigo, dando assim, um pouco de "inteligência" ao programa. Tem dois valores de retorno, 0 se o artigo existir no armazém, 1 caso contrário.

```
int inserirArtigo(LLista *l, int r, char *n, float preco, int q)
```

Esta função insere um determinado artigo passado como argumento e o endereço da cabeça de lista da lista ligada com todos os artigos do armazém. O endereço da cabeça de lista é colocado num apontador. Assim, o parâmetro l é um apontador para apontador. Enviamos o endereço da cabeça da lista porque, na função se inserirmos um artigo no início, a cabeça da lista é modificada. Esta função insere os artigos ordenadamente pelo número de referência dos artigos. Se o artigo a inserir já existe no armazém é enviado para o output a respetiva mensagem e pergunta se pretende efetuar a operação, retornando o valor 0. Se não existir adiciona-o à lista e retorna o valor 1. Se tais operações não são efetuadas, retorna o valor -1.

```
int removerArtigo(LLista *l, int refe)
```

Esta função funciona de forma idêntica à anterior. Neste caso, recebe uma cópia da referência do artigo e o endereço da cabeça de lista. Se encontrar a referência no armazém remove o artigo libertando a memória alocado pelo artigo através da função "free()". Os valores de retorno 1 e 0, correspondem, respetivamente, a se o artigo foi removido com sucesso ou se tal facto não acontecer.

```
int atualizarStock(LLista *l, int refe, int quantiEncomendar)
```

Quando efetuamos uma encomenda com sucesso, o stock do armazém tem que ser atualizado. Esta função foi definida para esse fim. Recebe o endereço da cabeça de lista, a referencia do artigo e a quantidade a encomendar. Se a quantidade encomendada é igual ao stock que existe em armazém, então o artigo é removido da lista ligada através da função "free()". Se a quantidade a encomendar é inferior, então o stock é apenas atualizado através da instrução "(p->artigo).quantidade -= quantiEncomendar".

```
int fazerEncomendas(LLista,int *,char *,float *,int *,int *)
```

Esta função tem como objetivo verificar se a encomenda de um determinado artigo é exequível, ou seja, se a quantidade em stock é maior ou igual do que a quantidade a encomendar. Se não for exequível, é enviado para o standard input/output a respetiva mensagem e a função retorna o valor 0. Se existir quantidade suficiente em stock, retorna o valor 1.

```
int lerFicheiro(LLista *l, char *nomeFich)
```

A função "lerFicheiro()" é aplicada quando se pretende carregar os dados do ficheiro que contém todos os artigos do armazém para a lista ligada que representa a lista dos artigos em stock no armazém. Para tal, recebe como parâmetros o endereço do primeiro elemento da lista e o nome do ficheiro.

Para fazer a leitura de uma ficheiro sequencial usando as funções `fscanf()` e `fgets()`, o ficheiro tem de estar escrito com um determinado padrão. O padrão do nosso ficheiro é: a primeira linha corresponde a uma string com o nome do artigo e na segunda linha tem um número inteiro que corresponde à referência do artigo, um número real que corresponde ao preço do artigo e um número inteiro que corresponde à quantidade do artigo. As restantes linhas seguem sempre este padrão. Na figura 3.2 podemos ver como é representado uma linha no ficheiro.

| | |
|----------------------------------|--|
| HEMNES est 49x197 castanho claro | <code>while(fgets(frase,50,fp) != NULL){</code> |
| 235 34.00 6 | <code> comprimento=strlen(frase);</code> |
| BILLY EST 80X28X202 PRET CAST | <code> if(frase[comprimento-1]=='\n'){</code> |
| 236 78.00 7 | <code> frase[comprimento-1]=0;</code> |
| | <code> }</code> |
| | <code> fscanf(fp,"%d %f %d\n",&r, &p, &q);</code> |
| | <code> inserirArtigo(l,r,frase,p,q);</code> |
| | <code>}</code> |

Figura 3.2: padrão no ficheiro que contém todos os artigos e código em C que lê o respetivo ficheiro

Para ler uma frase utilizamos a função `"fgets(frase,50,fp) != NULL"` que lê uma frase de 50 bytes e coloca-a na variável `"frase"`. Esta função, para além de todos os caracteres da frase, também lê o carácter `"new line"`. Quando fazemos a leitura do ficheiro para a lista ligada, o carácter `"new line"` não pode ser lido. Para contornar este facto, eliminar esse carácter da string lida, percorremos todos os caracteres da frase e quando encontrar `"new line"` colocamos o carácter terminador de string. Assim, é efetuada uma leitura eficiente para a lista ligada do nome do artigo. De seguida uso a função `"fscanf()"` para fazer a leitura da linha seguinte, que contém a referência, o preço e a quantidade do artigo. A leitura termina quando a função `"fgets()"` devolver `NULL`, que significa que chegou ao fim do ficheiro.

Capítulo 4

Codificação e Testes

4.1 Alternativas, Decisões e Problemas de Implementação

No decorrer da implementação em linguagem C do projeto, foi necessário efetuar algumas alternativas e tomar algumas decisões.

Uma das decisões foi qual seria a melhor forma de processar um artigo, se pelo seu nome ou pela sua referência. Como o nome de um artigo tem vários caracteres e torna-se ambíguo com nomes de artigos parecidos, então escolhemos a referência do artigo, que é um número único de no mínimo três dígitos. Assim, quando é feito o processamento de algum artigo, é sempre pedido ao utilizador a sua referência e não o seu nome.

Decidimos que a lista ligada que irá conter todos os artigos do armazém fosse uma lista ordenada por ordem crescente do número da referência dos artigos. Assim, a procura, inserção e remoção de um artigo é uma tarefa mais rápida pois, só tem que se percorrer a lista enquanto o artigo é menor. Decidimos optar por este método, também pelo facto de que um armazém com um certo nível de organização, para além de mais eficiente, é mais estético.

Uma encomenda, como dito anteriormente, tem sempre um número associado que a identifica. Este número é um número gerado aleatoriamente, mas não pode ser repetido por nenhuma outra encomenda. Uma encomenda com um número é uma encomenda mais fácil de se encontrar. A lista ligada que irá conter todas as encomendas efetuadas, é uma lista ligada ordenada por ordem crescente do número da encomenda.

4.2 Testes realizados e Resultados

Quando é executado o programa, é enviado para o standard input/output uma mensagem inicial com uma pergunta, se pretende inicializar o programa ou inicia-lo como os dados da sessão anterior. Esta mensagem encontra-se na figura 4.1.

```
*****
                        GESTAO
                        DE UM ARMAZEM

Copyright c 2018 Todos os direitos reservados
realizado por : Bruno e Catarina
                universidade do minho

*****

Primeira vez que utiliza o programa, deve inicializa-lo (instalar).
Prima 1 para ir para o Menu Inicial ou 2 para inicializar o programa: _
```

Figura 4.1: Abertura do programa

De seguida temos o Menu Inicial do programa. É no menu que o utilizador pode escolher a operação que pretende efetuar. A imagem 4.2 mostra o menu inicial do nosso programa.

```
***** 1 MENU INICIAL *****

1. Inserir artigo
2. Ver informacoes de um artigo
3. Remover artigo
4. Alterar informacoes de um artigo
5. Ver lista de stock do armazem
6. Numero total de artigos e referencias no armazem

7. Fazer uma encomenda
8. Ver lista de encomendas realizadas
9. Numero Total de encomendas realizadas
10. Numero total de referencias e artigos encomendados
11. Valor total efetuado

11. Definicoes

0. Terminar o Programa

***** 1 MENU INICIAL *****

Insira a opcao: _
```

Figura 4.2: Abertura do programa

Se o utilizar decidir adicionar um artigo pode escolhe a opção número 1 do menu inicial. O output seguinte é apresentado na figura 4.3 na parte esquerda. Se o utilizador colocar o a referência 235, como essa referência já existe no armazém é enviado para o standard i/o a mensagem que se encontra na figura 4.3 na parte direita.

| | |
|--|---|
| <pre>Menu Inicial 1. Inserir artigo Insira a referencia: _</pre> | <pre>Menu Inicial 1. Inserir artigo Insira a referencia: 235 Referencia ja existe no armazem. Referencia: 235 Nome HEMNES est 49x197 castanho claro Preco: 34.00 euros Quantidade: 6 Insira quantidade a adicionar: _</pre> |
|--|---|

Figura 4.3: Opção 1 do menu inicial

Se escolher a opção 3 do menu inicial, remover um artigo, o utilizador de seguida digita a referência do artigo a remover. Se a referência não existir no armazém é enviado para o standard i/o a respetiva mensagem de erro, se o artigo existir, é removido. Este ultimo caso é retratado na figura 4.4.

```
Artigo a remover:
Nome: GERESTA clch mol 160x200 firme cinza
Referencia: 146
Preco: 399.00 euros
Quantidade: 12

ARTIGO REMOVIDO COM SUCESSO.

Prima <M> para o Menu Inicial. Prima <0> para sair do programa:
```

Figura 4.4: Opção 3 - remover um artigo

Para a visualização da lista de todos os artigos armazenados no armazém, o utilizador deve inserir a opção 5. A figura 4.5 mostra a parte inicial da lista de todos os artigos presentes no armazém.

```
Menu Inicial
  4. Ver lista de todos os artigos

LISTA DE TODOS OS ARTIGOS NO ARMAZEM

FRIHETEN ass castanho
102 22.44 euros 11

KIVIK ES SF 2
103 109.00 euros 9

KIVIK cp sf 3 azul escuro
104 18.00 euros 14

KIVIK CP SF 3 ORRSTA CZ CL
105 19.00 euros 13

KIVIK ES SF 3
106 249.00 euros 11

STOCKSUND cp sf cinz
110 5.88 euros 12

STOCKSUND cp sf 2 lug cinz
111 11.45 euros 5

GERESTA clch mol 140x200 firme branco
140 250.00 euros 22

HOVAG colch molas ensac 140x190 firme cinz esc
141 144.00 euros 22
```

Figura 4.5: lista dos artigos presentes no armazém

O nosso programa tem uma opção em que podemos ver, de forma atualizada, quantas referencias existem no armazém e o número total de artigos. A figura 4.6 retrata-nos isso.

```
Menu Inicial
  5. Numero total de artigos e referencias

Numero de referencias: 108
Numero de artigos: 3611

Prima <M> para o Menu Inicial. Prima <0> para sair do programa: _
```

Figura 4.6: lista dos artigos presentes no armazém

Para fazer uma encomenda, selecionamos a opção 7. A figura 4.7 mostra-nos um exemplo de um utilizador que pretende efetuar uma encomenda com a referência 234. O utilizador também pode adicionar mais referências e artigos à sua encomenda. A figura 4.8 mostra nos um exemplo de uma encomenda com 8 referências diferentes. Neste exemplo, o número que identifica a encomenda é o 51 cujo valor total é de 677,00 euros.

| | |
|---|--|
| <pre>Menu Inicial 6. Fazer uma encomenda Insira a referencia: 234 Artigo a encomendar: Nome: BILLY EST 80X28X202 BR Referencia: 234 Preco: 23.00 euros Stock disponivel: 222 Quantidade a encomendar: 10</pre> | <pre>ARTIGO REGISTADO Nome: BILLY EST 80X28X202 BR Referencia: 234 Quantidade do artigo a encomendar: 10 Preco Total dos artigos: 230.00 euros Prima 1 para inserir novo artigo na encomenda. 0 para Terminar encomenda</pre> |
|---|--|

Figura 4.7: Processos para efetuar uma encomenda

```

Encomenda Final:
    Numero da Encomenda: 51
    Numero de Referencias encomendadas: 8
    Numero de Artigos encomendados: 29
    Preco Total: 677.00 euros

STOCK NO ARMAZEM ATUALIZADO COM SUCESSO.

Prima <M> para o Menu Inicial. Prima <0> para sair do programa: _

```

Figura 4.8: exemplo de uma encomenda completa

Ao seleccionar a opção 8, podemos ver a lista de todas as encomendas efetuadas. Na figura 4.9 podemos ver algumas das encomendas já realizadas.

```

Menu Inicial
  7. Ver lista de encomendas realizadas

Numero Encomenda    Numero de Referencias  Numero de Artigos    Preco Total
-----
    1                33                5                112.00 euros
   51                 8                29                677.00 euros
    6                16                26               3669.00 euros
    3                25                 4                511.00 euros
    2                43                14                270.00 euros
    5                97                 5                435.00 euros
   85                 2                10                230.00 euros

Prima <M> para o Menu Inicial. Prima <0> para sair do programa: _

```

Figura 4.9: lista de encomendas efetuadas

Na opção 11 colocamos as definições. No sub-menu definições existem duas opções. Como usar o programa ou restaurar o programa. A opção como usar o programa tem como objetivo ajudar o utilizador com as suas dúvidas acerca do funcionamento do mesmo.

```

***** 1.9 DEFENICOES *****

    1. Como usar o programa
    2. Restaurar o programa - Inicializar o armazem

    5. Ir para o Menu Inicial

***** 1.9 DEFENICOES *****

Insira a opcao:

```

Figura 4.10: sub-menu definições

4.3 Estrutura do Código e ficheiros utilizados

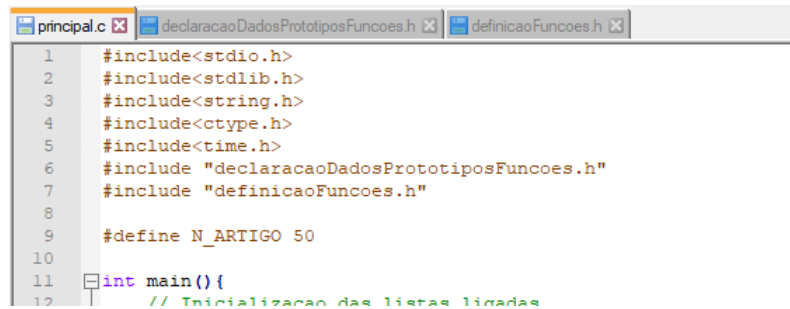
Para este projeto, ao contrário do anterior, decidimos dividir o ficheiro com o código em três ficheiros.

Um ficheiro designado por "principal.c" onde colocamos apenas a função "main()". Este ficheiro tem como objetivo o controlo da execução do programa.

Um ficheiro designado por "definicaoDadosPrototiposFuncoes.h", onde definimos os novos tipos de dados e colocamos as declarações (protótipos) das funções.

Um ficheiro designado por "definicaoFuncoes.h" onde definimos as funções que utilizamos no projeto.

A figura 4.10 mostra como fizemos o "include" dos dois ficheiros anteriores no ficheiro com a função "main()".



```
1  #include<stdio.h>
2  #include<stdlib.h>
3  #include<string.h>
4  #include<ctype.h>
5  #include<time.h>
6  #include "declaracaoDadosPrototiposFuncoes.h"
7  #include "definicaoFuncoes.h"
8
9  #define N_ARTIGO 50
10
11  int main() {
12      // Inicializacao das listas ligadas
```

Figura 4.11: lista de encomendas efetuadas

Para além dos ficheiros anteriores, para o funcionamento correto deste programa este projeto ainda engloba mais dois ficheiros, "listaArtigos.txt" e "listaEncomendas.txt", que serão usados para armazenar as listas ligadas com os artigos do armazém e as encomendas efetuadas, respetivamente.

Capítulo 5

Conclusão

Através da realização deste projeto proposto no âmbito da unidade curricular de métodos de programação II do curso de MIETI, conseguimos compreender e aplicar os conhecimentos adquiridos nas aulas. Cumprimos os objetivos propostos na introdução. Implementamos um programa eficaz e que cumpre os requisitos propostos.

Decidimos, para além de cumprir os requisitos pedidos no enunciado, elaborar um programa que possui uma excelente interação com o utilizador comum e que fosse mais amplo nas opções que podia oferecer.

As operações que se podem realizar sobre o armazém são: (i) inserir um artigo; (ii) ver informações de um artigo; (iii) remover um artigo; (iv) alterar informações de um artigo; (v) ver lista de todos os artigos em armazém; (vi) ver número total de artigos e referencias no armazém; (vii) fazer uma encomenda; (viii) ver listas de encomendas realizadas; (ix) ver número total de encomendas realizadas; (x) ver número total de referências; artigos e o preço das encomenda; (xi) ver valor faturado de todas as encomendas feitas;

Um dos aspetos negativos deste projeto é as limitações que um utilizador tem sobre as encomendas efetuadas. O utilizador não pode remover uma encomenda que já tenha sido efetuada.

Os principais conceitos da linguagem C abordados neste projeto passaram pela utilização de estruturas, apontadores, alocação de memória, listas ligadas e ficheiros de texto.

Para a elaboração deste relatório, utilizamos o software \LaTeX . Tal como dito no primeiro projeto, concluímos que este software é uma excelente forma de produzir trabalhos científicos, relatórios e textos académicos com excelente qualidade.

Capítulo 6

Bibliografia

1. Damas, Luís (2001) Linguagem C, Lisboa: FCA
2. Slides colocados na BlackBoard Learn da unidade curricular
3. <https://www.latex-project.org/> - site oficial L^AT_EX.

Apêndice A

Código do Programa

Lista-se a seguir o código C do programa que foi desenvolvido.

Este apêndice engloba três ficheiros:

1. `declaracaoDadosPrototiposFuncoes.h`
2. `definicaoFuncoes.h`
3. `principal.c`

```

1  // FICHEIRO COM A DEFINICAO DO NOVOS TIPOS DE DADOS E DECLARACOES (PROTOTIPOS) DAS FUNCOES UTILIZADAS
2
3  #define N_ARTIGO 50
4
5  typedef struct IDartigo{
6      char nomeArtigo[N_ARTIGO];
7      int referencia;
8      float preco;
9      int quantidade;
10 } ID_ARTIGO;
11
12
13 typedef struct lligada{
14     ID_ARTIGO artigo;
15     struct lligada *prox;
16 }*LLista;
17
18 typedef struct encomendas{
19     int numeroEncomenda;
20     float precoTotal;
21     int numArtigos;
22     int numReferencias;
23     struct encomendas *prox;
24 }*LEncomendas;
25
26 // DECLARACAO DAS FUNCOES. PROTOTIPO DAS FUNCOES
27 void inic_random();
28 int inserirReferencia();
29 float inserirPreco();
30 void inserirNome(char *);
31 int inserirQuantidade();
32 int verDadosArtigo(LLista); // 191
33 int inserirInformacaoArtigo(LLista,int *,char *,float *,int *); //11 313
34 void imprimeLista(LLista); //400
35 void imprimeEncomendas(LEncomendas); //354
36 int numeroTotalReferencias(LLista);
37 int numeroTotalArtigos(LLista);
38 int procurarNomePrecoQuantidade_porReferencia(LLista,int,char *,float *,int *); // 118
39 LLista procurarArtigo(LLista,int);
40 LLista criarArtigo(int,char *,float,int);
41 int inserirArtigo(LLista *,int,char *,float,int);
42 int removerArtigo(LLista *,int); // 131
43 int existeNumeroEncomenda(LEncomendas,int);
44 LEncomendas alocarEncomenda(int,int,int,float);
45 int inserirListaEncomendas(LEncomendas *,int,int,int,int);
46 int fazerEncomendas(LLista,int *,char *,float *,int *,int *); //260
47 int atualizarStock(LLista *,int,int);
48 int escreverFicheiro(LLista,char *);
49 int lerFicheiro(LLista *,char *); //281
50 int escreverFicheiroEncomendas(LEncomendas,char *); //224
51 int lerFicheiroEncomendas(LEncomendas *,char *); //301
52 int criarFicheiro(char *);
53 int menu();
54 int esperar();
55 void creditos();
56 int subMenuDefenicoes();
57

```

```

1 // FICHEIRO COM AS DEFENICOES DAS FUNCOES
2
3 #define N_ARTIGO 50
4
5 void inic_random(){
6     long ultime;
7     time(&ultime);
8     srand((unsigned) ultime);
9 }
10 // Insere uma referencia de uma artigo. Devolve como resultado da funcao o inteiro inserido
11 int inserirReferencia(){
12     int referencia;
13     printf("Insira a referencia: ");
14     scanf("%d",&referencia);
15     return referencia;
16 }
17
18 // Insere o preco de um artigo
19 // Devolve como resultado da funcao o preco introduzido
20 float inserirPreco(){
21     float preco;
22     printf("Insira o preco: ");
23     scanf("%f",&preco);
24     return preco;
25 }
26
27
28 // Insere o nome de um artigo
29 // Recebe uma string
30 void inserirNome(char *nomeArtigo){
31     printf("Insira o nome do artigo: ");
32     gets(nomeArtigo);
33 }
34
35 // Insere a quantidade
36 int inserirQuantidade(){
37     int quantidade;
38     printf("Insira a quantidade do artigo: ");
39     scanf("%d",&quantidade);
40     return quantidade;
41 }
42
43 // criar um novo elemento da lista ligada, que no contexto do problema e um artigo
44 // Devolve o endereco do elemento criado em caso de sucesso, em caso de insucesso, devolve NULL
45 LLista criarArtigo(int refe, char *nome, float p, int q){
46     LLista novo = (LLista) malloc(sizeof(struct lligada));
47     if(novo!=NULL){
48         (novo->artigo).referencia = refe;
49         strcpy((novo->artigo).nomeArtigo,nome);
50         (novo->artigo).preco=p;
51         (novo->artigo).quantidade = q;
52         novo->prox = NULL;
53     }
54     return novo;
55 }
56
57 // Procura se existe um artigo pela sua referencia na lista
58 //A funcao rece a cabeca da lista e a referencia a procurar
59 // Devolve o endereco do artigo se encontrar a sua referencia. Devolve NULL caso a referencia nao existe na
60 lista
61 LLista procurarArtigo(LLista l, int refe){
62     while(l!=NULL && (l->artigo).referencia != refe)
63         l=l->prox;
64     return l;
65 }

```

```

66
67 // Insere um artigo na lista.
68 // A lista esta ordenada por ordem crescente do numero da referencia do artigo
69 // Tem como parametro um apontador para apontador. Recebe o endereco da cabeca da lista
70 // Devolve, -1 se nao conseguir criar o artigo. 0 se o artigo ja existir em stock(atualiza o stock). 1 se
for criado um novo artigo.
71 int inserirArtigo(LLista *l, int r, char *n, float preco, int q){
72     int nOpcao;
73     LLista p = NULL, novo=criarArtigo(r,n,preco,q);
74     if(novo==NULL){
75         printf("Erro ao criar o artigo. Tente novamente.\n");
76         return -1;
77     }
78     if((p=procurarArtigo((l),(novo->artigo).referencia))!=NULL){
79         printf("A referencia introduzida existe no armazem. Prima 1 para atualizar o stock ou 0 para sair:
");
80         scanf("%d",&nOpcao);
81         if(nOpcao==0)
82             return 0;
83         (p->artigo).quantidade = (p->artigo).quantidade + (novo->artigo).quantidade;
84         printf("Stock Atualizado Com Sucesso.\n");
85         return 1;
86     }
87     if((l)==NULL){
88         (*l)=novo;
89         return 1;
90     }
91     if((novo->artigo).referencia < ((*l)->artigo).referencia){
92         novo->prox=(*l);
93         (*l)=novo;
94         return 1;
95     }
96     p=(*l)->prox;
97     while(p!=NULL && (novo->artigo).referencia > (p->artigo).referencia){
98         l=&((*l)->prox);
99         p=p->prox;
100     }
101     novo->prox=p;
102     (*l)->prox=novo;
103     return 1;
104 }
105
106 // Funcao recursiva que verifica se existe uma encomenda com o mesmo numero
107 int existeNumeroEncomenda(LEncomendas l, int n){
108     if(l==NULL)
109         return 0;
110     if(l->numeroEncomenda == n)
111         return 1;
112     existeNumeroEncomenda(l->prox,n);
113 }
114
115
116 // Procura o nome e o preco de uma artigo no armazem pela referencia
117 // devolve 0 caso nao o encontra. 1 se o artigo existe.
118 int procurarNomePrecoQuantidade_porReferencia(LLista l, int refe, char *nome, float *pre, int *quanti){
119     LLista p=NULL;
120     if((p=procurarArtigo(l,refe))!=NULL)
121         return 0;
122     strcpy(nome,(p->artigo).nomeArtigo);
123     (*pre)=(p->artigo).preco;
124     (*quanti)=(p->artigo).quantidade;
125     return 1;
126 }
127
128 // Funcao que remova um artigo da lista ligada
129 // Recebe o o endereco da cabeca da lista e e colocada numa variavel do tipo apontador

```

```

130 // Devolve 0 se o artigo nao existir na lista. 1 se a remocao foi feita com sucesso
131 int removerArtigo(LLista *l, int refe){
132     LLista p=(*l), q=NULL;
133     if(*l==NULL)
134         return 0;
135     if((*l->artigo).referencia == refe){
136         (*l)=(*l)->prox;
137         free(p);
138         return 1;
139     }
140     q=(*l)->prox;
141     while(q!=NULL && refe > (q->artigo).referencia){
142         p=q;
143         q=q->prox;
144     }
145     if(q!=NULL){
146         p->prox=q->prox;
147         free(q);
148         return 1;
149     }
150     return 0;
151 }
152
153 LEncomendas alocarEncomenda(int numeroEncomenda, int numReferenciasEncomenda, int numArtigosEncomenda,
float precoTotalEncomenda){
154     LEncomendas novo = (LEncomendas) malloc (sizeof(struct encomendas));
155     if(novo!=NULL){
156         novo->numeroEncomenda = numeroEncomenda;
157         novo->numReferencias = numReferenciasEncomenda;
158         novo->numArtigos = numArtigosEncomenda;
159         novo->precoTotal = precoTotalEncomenda;
160         novo->prox = NULL;
161     }
162     return novo;
163 }
164
165 int inserirListaEncomendas(LEncomendas *en, int numeroEncomenda, int numReferenciasEncomenda, int
numArtigosEncomenda, int precoTotalEncomenda){
166     LEncomendas novo = alocarEncomenda(numReferenciasEncomenda, numeroEncomenda, numArtigosEncomenda,
precoTotalEncomenda);
167     LEncomendas q = NULL;
168     if(novo==NULL)
169         return 0;
170     if(*en==NULL){
171         (*en)=novo;
172         return 1;
173     }
174     if((novo->numeroEncomenda) < (*en->numeroEncomenda){
175         novo->prox = (*en);
176         (*en)=novo;
177         return 1;
178     }
179     q=(*en)->prox;
180     while(q!=NULL && (novo->numeroEncomenda) < (q->numeroEncomenda)){
181         en=&((*en)->prox);
182         q=q->prox;
183     }
184     novo->prox = q;
185     (*en)->prox = novo;
186     return 1;
187 }
188
189
190 int verDadosArtigo(LLista l){
191     int refe;
192     LLista p=NULL;

```

```

193     printf("Insira a referencia do artigo: ");
194     scanf("%d",&refe);
195     p=procurarArtigo(l,refe);
196     if(p==NULL){
197         printf("\nNao existe nenhum artigo com essa referencia.\n");
198         return 0;
199     }
200     printf("\nInformacao do artigo:\n\n");
201     printf("\t Referencia:   %d\n", (p->artigo).referencia);
202     printf("\t Nome:         %s\n", (p->artigo).nomeArtigo);
203     printf("\t Preco:         %.2f euros\n", (p->artigo).preco);
204     printf("\t Quantidade:   %d\n", (p->artigo).quantidade);
205 }
206
207
208
209 int escreverFicheiro(LLista l, char *nomeFich){
210     FILE *fp = NULL;
211     if((fp=fopen(nomeFich,"w+"))==NULL){
212         printf("Erro ao abrir ficheiro\n");
213         return 0;
214     }
215     while(l!=NULL){
216         fprintf(fp,"%s\n", (l->artigo).nomeArtigo);
217         fprintf(fp,"\t %d \t%.2f \t%d\n", (l->artigo).referencia, (l->artigo).preco, (l->artigo).quantidade);
218         l=l->prox;
219     }
220     fclose(fp);
221     return 1;
222 }
223
224 int escreverFicheiroEncomendas(LEncomendas l, char *nomeFichE){
225     FILE *fp = NULL;
226     if((fp=fopen(nomeFichE,"w"))==NULL){
227         printf("Erro ao abrir ficheiro\n");
228         return 0;
229     }
230     while(l!=NULL){
231         fprintf(fp,"\t%d\t\t%d\t\t%d \t\t%.2f\n", l->numeroEncomenda, l->numReferencias, l->numArtigos, l->precoTotal);
232         l=l->prox;
233     }
234     fclose(fp);
235 }
236
237 int atualizarStock(LLista *l, int refe, int quantiEncomendar){
238     LLista p = procurarArtigo((*l),refe);
239     LLista q = NULL;
240     if((p->artigo).quantidade > quantiEncomendar){
241         (p->artigo).quantidade -= quantiEncomendar;
242         return 1;
243     }
244     if(*l==p){
245         (*l)=(*l)->prox;
246         free(p);
247         return 1;
248     }
249     p=(*l);
250     q=p->prox;
251     while(q!=NULL && (q->artigo).referencia != refe){
252         p=q;
253         q=q->prox;
254     }
255     p->prox=q->prox;
256     free(q);

```

```

257     return 1;
258 }
259
260 int fazerEncomendas(LLista l, int *refe, char *n, float *pre, int *quanti, int *quantiEncomendar){
261     printf("\n");
262     (*refe)=inserirReferencia();
263     if((procurarNomePrecoQuantidade_porReferencia(l,*refe,n,pre,quanti))==0){
264         printf("Nao e possivel encomendar o artigo. Artigo nao existe no armazem.\n\n");
265         return 0;
266     }
267     printf("\nArtigo a encomendar:\n");
268     printf("\tNome: %s\n",n);
269     printf("\tReferencia: %d\n",(*refe));
270     printf("\tPreco: %.2f euros\n",(*pre));
271     printf("\tStock disponivel: %d\n\n",(*quanti));
272     printf("Quantidade a encomendar: ");
273     scanf("%d",quantiEncomendar);
274     if((*quantiEncomendar) > (*quanti)){
275         printf("Nao e possivel efetuar a encomenda. Quantidade pretendida superior ao stock em armazem\n\n"
);
276         return 0;
277     }
278     return 1;
279 }
280
281 int lerFicheiro(LLista *l, char *nomeFich){
282     FILE *fp=NULL;
283     int r,q,comprimento;
284     float p;
285     char frase[50];
286     if((fp=fopen(nomeFich,"r"))==NULL){
287         return 0;
288     }
289     while(fgets(frase,50,fp) != NULL){
290         comprimento=strlen(frase);
291         if(frase[comprimento-1]=='\n'){
292             frase[comprimento-1]=0;
293         }
294         fscanf(fp,"%d %f %d\n",&r, &p, &q);
295         inserirArtigo(l,r,frase,p,q);
296     }
297     fclose(fp);
298 }
299
300 int lerFicheiroEncomendas(LEncomendas *en, char *nomeFichE){
301     FILE *fp = NULL;
302     int numeroEncomenda, numeroReferencias, numeroArtigos;
303     float precoTotal;
304     if((fp=fopen(nomeFichE,"r"))==NULL)
305         return 0;
306     while((fscanf(fp,"%d %d %d %f",&numeroEncomenda, &numeroReferencias, &numeroArtigos, &precoTotal)) !=
EOF){
307         inserirListaEncomendas(en, numeroEncomenda, numeroReferencias, numeroArtigos, precoTotal);
308     }
309     fclose(fp);
310     return 1;
311 }
312
313 // Insere as informacoes de um artigo
314 // devolve 1 se o artigo ja existe. 2 caso contrario
315 int inserirInformacaoArtigo(LLista l,int *r, char *n, float *p, int *q){
316     LLista pt_l = NULL;
317     int d;
318     (*r)=inserirReferencia();
319     pt_l = procurarArtigo(l,(*r));
320     if(pt_l!=NULL){

```



```

321     printf("\nReferencia ja existe no armazem.\n");
322     strcpy(n,(pt_l->artigo).nomeArtigo);
323     (*p)=(pt_l->artigo).preco;
324     (*q)=(pt_l->artigo).quantidade;
325     printf("\tReferencia: %d\n",(*r));
326     printf("\tNome %s\n",n);
327     printf("\tPreco: %.2f euros\n",(*p));
328     printf("\tQuantidade: %d\n\n",(*q));
329     printf("Insira quantidade a adicionar: ");
330     scanf("%d",&d);
331     (l->artigo).quantidade += d;
332     return 1;
333 }
334 fflush(stdin);
335 inserirNome(n);
336 (*p)=inserirPreco();
337 (*q)=inserirQuantidade();
338 return 2;
339 }
340
341 // Funcao que cria um ficheiro com o nome enviado como argumento
342 // Devolve 1 em caso de sucesso. 0 se nao foi possivel criar
343 int criarFicheiro(char *nomeFich){
344     FILE *fp=NULL;
345     if((fp=fopen(nomeFich,"w"))==NULL)
346         return 0;
347     fclose(fp);
348     return 1;
349 }
350
351 // imprime a lista com todas as encomendas efetuadas
352 void imprimeEncomendas(LEncomendas l){
353     printf(" Numero Encomenda      Numero de Referencias   Numero de Artigos    Preco Total\n\n");
354     while(l!=NULL){
355         printf("\t%d\t\t\t%d\t\t%2d\t\t%.2f euros\n",l->numeroEncomenda, l->numReferencias, l->numArtigos,
356             l->precoTotal);
357         l=l->prox;
358     }
359 }
360
361 int menu(){
362     // mostra as opcoes do menu
363     int n;
364     puts("\t***** 1 MENU INICIAL *****");
365     putchar('\n');
366     printf("\n\n");
367     printf("\t\t\t 1. Inserir artigo\n");
368     printf("\t\t\t 2. Ver informacoes de um artigo\n");
369     printf("\t\t\t 3. Remover artigo\n");
370     printf("\t\t\t 4. Alterar informacoes de um artigo\n");
371     printf("\t\t\t 5. Ver lista de stock do armazem\n");
372     printf("\t\t\t 6. Numero total de artigos e referencias no armazem\n");
373     putchar('\n');
374     printf("\t\t\t 7. Fazer uma encomenda\n");
375     printf("\t\t\t 8. Ver lista de encomendas realizadas\n");
376     printf("\t\t\t 9. Numero Total de encomendas realizadas\n");
377     printf("\t\t\t 10. Numero total de referencias e artigos encomendados\n");
378     printf("\t\t\t 11. Valor total efetuado\n");
379     putchar('\n');
380     printf("\t\t\t 11. Definicoes\n\n");
381     printf("\t\t\t 0. Terminar o Programa\n");
382     puts("\t***** 1 MENU INICIAL *****");
383     putchar('\n');
384     do{
385         printf("Insira a opcao: ");

```

```

386         scanf(" %d",&n);
387     }
388     while((n<1 || n>12) && n!=0);
389     return n;
390 }
391
392 int esperar(){
393     char c='a';
394     do{
395         printf("Prima <M> para o Menu Inicial. Prima <0> para sair do programa: ");
396         scanf(" %c",&c);
397     }
398     while(toupper(c)!='M' && c!='0');
399     if(c=='0')
400         return 0;
401     else
402         return 1;
403 }
404
405
406 // imprime no standard input/output a lista total de todos os artigos existentes no armazem
407 // Recebe o endereco do primeiro membro da lista
408 void imprimeLista(LLista l){
409     printf("\n\t\t\t\tLISTA DE TODOS OS ARTIGOS NO ARMAZEM\n\n");
410     while(l!=NULL){
411         printf("%s\n",l->artigo).nomeArtigo);
412         printf("%4d\t %.2f euros \t %d\n\n",l->artigo).referencia, (l->artigo).preco, (l->artigo).
quantidade);
413         l=l->prox;
414     }
415 }
416
417 // Funcao recursiva que calcula o numero total de referencias existentes no armazem
418 int numeroTotalReferencias(LLista l){
419     if(l==NULL)
420         return 0;
421     return 1+numeroTotalReferencias(l->prox);
422 }
423
424 // Funcao recursiva que calcula o numero total de artigos
425 int numeroTotalArtigos(LLista l){
426     if(l==NULL)
427         return 0;
428     return (l->artigo).quantidade + numeroTotalArtigos(l->prox);
429 }
430
431 void credits(){
432     printf("\t*****\n");
433     printf("\n");
434     printf("\t\t\t\t\tGESTAO \n");
435     printf("\t\t\t\t\tDE UM ARMAZEM\n");
436     printf("\n");
437     printf("\n");
438     printf("\n");
439     printf("\t Copyright c 2018 Todos os direitos reservados\n");
440     printf("\t\t\t\t\trealizado por : Bruno e Catarina\n");
441     printf("\t\t\t\t\tuniversidade do minho\n");
442     printf("\n");
443     printf("\n");
444     printf("\t*****\n");
445 }
446
447 int subMenuDefenicoes(){
448     int n;
449     puts("\t***** 1.9 DEFENICOES *****");
450     putchar('\n');

```

```
451     printf("\n\n");
452     printf("\t\t\t 1. Como usar o programa\n");
453     printf("\t\t\t 2. Restaurar o programa - Inicializar o armazem\n");
454     putchar('\n');
455     printf("\t\t\t 5. Ir para o Menu Inicial\n");
456     printf("\n\n");
457     puts("\t***** 1.9 DEFENICOES *****");
458     putchar('\n');
459     do{
460         printf("Insira a opcao: ");
461         scanf(" %d",&n);
462     }
463     while((n<1 || n>2) && n!=5);
464     return n;
465 }
```

```

1  // FICHEIRO COM A FUNCAO MAIN
2
3  #include<stdio.h>
4  #include<stdlib.h>
5  #include<string.h>
6  #include<ctype.h>
7  #include<time.h>
8  #include "declaracaoDadosPrototiposFuncoes.h"
9  #include "definicaoFuncoes.h"
10
11 #define N_ARTIGO 50
12
13 int main(){
14     // Inicializacao das listas ligadas
15     LLista l = NULL;
16     LEncomendas en = NULL;
17
18     int r,q, quantiEncomendar, nOpcao, x;
19     int numArtigosEncomenda, numReferenciasEncomenda, numeroEncomenda;
20     float p;
21     float precoTotalEncomenda;
22     char n[100];
23     char nomeFich[20]="listaArtigos.txt";
24     char nomeFichE[20]="listaEncomendas.txt";
25
26
27     // Se for a primeira vez a utilizar o programa
28     // tem que se criar os ficheiros que nos quais se armazen a lista de encomentas e a lista dos artigos
29     creditos();
30     printf("\nPrimeira vez que utiliza o programa, deve inicializa-lo (instalar).\n");
31     printf("Prima 1 para ir para o Menu Inicial ou 2 para inicializar o programa: ");
32     scanf("%d",&nOpcao);
33     if(nOpcao==2){
34         x = criarFicheiro(nomeFich);
35         if(x=0){
36             printf("Erro ao criar o ficheiro dos artigos do armazem.\n");
37             nOpcao=esperar();
38             exit(1);
39         }
40         x = criarFicheiro(nomeFichE);
41         if(x=0){
42             printf("Erro ao criar o ficheiro da lista de encomendas ja efetuadas.\n");
43             nOpcao=esperar();
44             exit(2);
45         }
46     }
47     else{
48         // carregar os dados para as listas ligadas
49         lerFicheiro(&l, nomeFich);
50         lerFicheiroEncomendas(&en, nomeFichE);
51     }
52     inic_random();
53     while(nOpcao!=0){
54         system("cls");
55         nOpcao=menu();
56         //Inserir um artigo
57         if(nOpcao==1){
58             system("cls");
59             printf("Menu Inicial\n");
60             printf("\tl. Inserir artigo\n\n");
61             if(inserirInformacaoArtigo(l,&r,n,&p,&q) == 1) // ll 313
62                 ;
63             else
64                 inserirArtigo(&l,r,n,p,q);
65             printf("\nARTIGO ADICIONADO COM SUCESSO.\n\n");
66         }

```

```

67     if(nOpcao==2){
68         system("cls");
69         printf("Menu Inicial\n");
70         printf("\t2. Ver informacoes de um artigo\n\n");
71         verDadosArtigo(l);           // 191
72         printf("\n\n");
73     }
74     if(nOpcao==3){
75         system("cls");
76         printf("Menu Inicial\n");
77         printf("\t3. Remover um artigo\n\n");
78         r = inserirReferencia();
79         if(procurarNomePrecoQuantidade_porReferencia(l,r,n,&p,&q) == 0){
80             printf("\n\nArtigo nao existe no armazem.\n\n\n");
81         }
82         else{
83             system("cls");
84             printf("Artigo a remover:\n");           //131
85             printf("\tNome: %s\n",n);
86             printf("\tReferencia: %d\n",r);
87             printf("\tPreco: %.2f euros\n",p);
88             printf("\tQuantidade: %d\n",q);
89             removerArtigo(&l,r);
90             printf("\nARTIGO REMOVIDO COM SUCESSO.\n\n\n");
91         }
92     }
93     if(nOpcao==5){
94         system("cls");
95         printf("Menu Inicial\n");
96         printf("\t4. Ver lista de todos os artigos\n\n");
97         imprimeLista(l);           //400
98     }
99     if(nOpcao==6){
100         system("cls");
101         printf("Menu Inicial\n");
102         printf("\t5. Numero total de artigos e referencias\n\n");
103         printf("\n\t Numero de referencias: %d\n",numeroTotalReferencias(l));
104         printf("\t Numero de artigos: %d\n\n\n",numeroTotalArtigos(l));
105     }
106     if(nOpcao==7){
107         system("cls");
108         printf("Menu Inicial\n");
109         printf("\t6. Fazer uma encomenda\n\n");
110         precoTotalEncomenda=0;
111         numArtigosEncomenda=0;
112         numReferenciasEncomenda=0;
113         do{
114             numeroEncomenda = 1 + (rand()%100);
115         }
116         while(existeNumeroEncomenda(en,numeroEncomenda));
117         while(nOpcao!=0){
118             if(fazerEncomendas(l,&r,n,&p,&q,&quantiEncomendar) != 0){           // 260
119                 system("cls");
120                 printf("ARTIGO REGISTADO\n");
121                 printf("\tNome: %s\n",n);
122                 printf("\tReferencia: %d\n",r);
123                 printf("\tQuantidade do artigo a encomendar: %d\n",quantiEncomendar);
124                 printf("\tPreco Total dos artigos: %.2f euros\n\n",p*quantiEncomendar);
125                 precoTotalEncomenda=precoTotalEncomenda+(p*quantiEncomendar);
126                 numArtigosEncomenda=numArtigosEncomenda+quantiEncomendar;
127                 numReferenciasEncomenda++;
128                 atualizarStock(&l,r,quantiEncomendar);
129             }
130             printf("Prima 1 para inserir novo artigo na encomenda. 0 para Terminar encomenda\n");
131             scanf("%d",&nOpcao);
132             system("cls");

```

```

133     }
134     printf("Encomenda Final:\n");
135     printf("\tNumero da Encomenda: %d\n", numeroEncomenda);
136     printf("\tNumero de Referencias encomendadas: %d\n", numReferenciasEncomenda);
137     printf("\tNumero de Artigos encomendados: %d\n", numArtigosEncomenda);
138     printf("\tPreco Total: %.2f euros\n\n", precoTotalEncomenda);
139     printf("\nSTOCK NO ARMAZEM ATUALIZADO COM SUCESSO.\n\n\n");
140     inserirListaEncomendas(&en, numeroEncomenda, numReferenciasEncomenda, numArtigosEncomenda,
precoTotalEncomenda);
141     getchar();
142     nOpcao=1;
143 }
144 if(nOpcao==8){
145     system("cls");
146     printf("Menu Inicial\n");
147     printf("\t7. Ver lista de encomendas realizadas\n\n\n");
148     imprimeEncomendas(en); // 354
149     printf("\n\n");
150 }
151 if(nOpcao==11){
152     system("cls");
153     nOpcao = subMenuDefenicoes();
154     if(nOpcao==2){
155         x = criarFicheiro(nomeFich);
156         if(x=0){
157             printf("Erro ao criar o ficheiro dos artigos do armazem.\n");
158             exit(3);
159         }
160         x = criarFicheiro(nomeFichE);
161         if(x=0){
162             printf("Erro ao criar o ficheiro da lista de encomendas ja efetuadas.\n");
163             exit(4);
164         }
165         printf("PROGRAMA RESTAURADO COM SUCESSO.\n\n");
166     }
167     if(nOpcao==5)
168         ;
169 }
170 // guardar os dados das listas ligadas nos respetivos ficheiros
171 escreverFicheiro(1,nomeFich);
172 escreverFicheiroEncomendas(en,nomeFichE);
173
174 if(nOpcao==0){
175     system("cls");
176     puts("Programa terminado com sucesso!");
177     puts("Obrigado\n");
178     credits();
179     exit(10);
180 }
181 nOpcao=esperar();
182 if(nOpcao==0){
183     system("cls");
184     puts("Programa terminado com sucesso!");
185     puts("Obrigado\n");
186     credits();
187     exit(10);
188 }
189 }
190 }

```