ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Projeto de Laboratório de Programação

Licenciatura em Engenharia Informática

2019/2020

Grupo 37

Gonçalo Nobre– 8190593

Jaques Resende – 8190214

José Miguel Silva – 8190237

• Introdução:

Neste projeto o pretendido é criar uma aplicação para uma empresa de transporte, utilizando conhecimentos abordados na cadeira de Laboratório de Programação, dos quais incluem: estruturas de dados, constantes, funções, arrays para guardar conjuntos de estruturas, operações básicas de gestão de utilizadores CRUD (Create, Read, Update, Delete) e memória dinâmica.

A aplicação foi desenvolvida com dois utilizadores principais o Diretor e o Cliente, sendo que o Diretor não necessita de se autenticar e tem todos os privilégios administrativos, no entanto o Cliente é obrigado a usar o seu NIF (Número de Identificação Fiscal) como identificador para aceder ao seu menu. O programa ainda permite guardar os dados em ficheiros para utilização futura.

A abordagem feita em relação a listas geradas a partir de memória dinâmica foi definir 2 variáveis para o número de elementos numa lista e para o tamanho máximo que uma lista pode tomar. Também foi decidido que sempre que houvesse uma função que utilizasse memória dinâmica a respetiva função iria ter um apontador duplo para a lista.

```
int opt = 0, nElementos = 0, tam_max_cli = 0;
int nEncomendas = 0, tam_max_enc = 0;
struct Cliente *ap_clientes;
struct Encomenda *ap_encomendas;
struct Preco precos;
ap_clientes = (struct Cliente*) malloc(TAM_INICIAL * sizeof(struct Cliente));
ap_encomendas = (struct Encomenda*) malloc(TAM_INICIAL * sizeof(struct Encomenda));
```

1. Excerto do código da main.c

• Funcionalidades requeridas

De modo cumprir as funcionalidades exigidas no trabalho foram criadas 5 estruturas, cada uma respetiva a uma lista ou subestrutura.

A abordagem para o tamanho das listas nas estruturas foi atribuir o mesmo tamanho através de uma constante de modo a não repetir o tamanho de cada lista. Para a morada optamos por definir uma estrutura à parte que pudesse ser utilizada como subestrutura nas Encomendas e Clientes.

Para dados que podiam assumir no mínimo 2 valores conhecidos utilizamos enumerações.

```
#define TAM_STR 100
#define TAM_INICIAL 1
#define TAM_INCREMENTO 1
#define TAM_DECREMENTO 1

enum Decricao {NFragil, Fragil};
enum Status {Desativo, Ativo};
enum Tipo {Regular, Urgente};
enum Expedicao {NExpedida, Expedida};
```

Aqui estão as estruturas de dados definidas no planeamento do trabalho:

```
struct Encomenda {
    struct Artigo *artigo;
    int nartigos;
    int nifcliente;
    struct tm hora_envio;
    enum Expedicao expedicao;
    enum Tipo tipo_transporte;
    float custo_total;
    float distancia_km;
    struct Morada endereco_origem;
    struct Morada endereco_destino;
};
```

```
struct Cliente {
    char nome[TAM_STR];
    int nif;
    int cc;
    enum Status status;
    struct Morada mpredef;
};
```

```
struct Morada{
    char rua[TAM_STR];
    int num;
    char localidade[TAM_STR];
    char codigopostal[TAM_STR];
    char pais [TAM_STR];
};
```

```
struct Artigo {
    float peso;
    float volume;
    enum Decricao descricao;
};
```

- ## Funcionalidades propostas

As funcionalidades propostas são as seguintes:

- Relatório anual/diário;
- Melhor e pior cliente;

- ## Estrutura analítica do projeto

O projeto tem um menu principal com 4 opções (Cliente, Diretor, Carregar dados, Guardar dados) que levam para os seguintes submenus (Gestão de Utilizadores, Gestão de

Encomendas, Gestão de Preços, Geração de Faturas). O Diretor a partir do menu de Gestão de Utilizadores é possível criar, editar, remover e reativar clientes enquanto que o cliente pode editar e remover o seu perfil. A partir do menu Gestão de Encomendas é exequível para o Diretor apenas atualizar o estado das encomendas enquanto que para o Cliente é possível adicionar, consultar, eliminar encomendas e ainda consultar o estado das mesmas. No menu Gestão de preços o Cliente pode consultar a tabela de preços que é previamente definida pelo Diretor no seu menu de gestão de preços para além de o mesmo a poder consultar e editar. Por último o Cliente e o Diretor podem ainda aceder ao menu de geração de faturas em que podem gerar faturas e reimprimi-las.

```
do {
    readInt(&opt, 0, 4, "1 - Menu Clientes\n2 - Menu Diretor\n3 - Carregar Dados\n"
            "4 - Guardar Dados\n4 - Sair do Menu\n\nInsira a opcao: ");
    switch(opt) {
        case 1: //Menu Cliente
            readInt(&nif,0,999999999,"Insira o seu NIF: ");
            pos = pesquisa(ap_clientes,nElementos,nif);
            if(pos == -1){
            readInt(&opt,0,1,"NÃ£o tem conta deseja criar uma?\n1 - Sim   0 - NÃ£o\n");
            switch(opt){
                case 0:
                    printf("A voltar ao menu..");
                    break;
                case 1:
                    adicionarCliente(&ap_clientes,&nElementos,&tam_max_cli, nif);
                    break;
            }


        }else{
                if((ap_clientes+pos)->status == Desativo) {
                    printf("A sua conta foi desativada\n");
                }
                else {
                    printf("A enviÃ¡-lo para o menu...\n");
                    MenuPrincipal(opt, &ap_clientes, &ap_encomendas,&nElementos, &nEncomendas, &tam_max_cli, &tam_max_enc, &precos, nif);
                }
        }
        break;
        case 2: //Menu Diretor
            MenuPrincipal(opt, &ap_clientes, &ap_encomendas, &nElementos, &nEncomendas, &tam_max_cli, &tam_max_enc, &precos, nif);
        break;
        case 3: carregarDadosCliente(&ap_clientes, &nElementos, &tam_max_cli);
                carregarDadosEncomenda(&ap_encomendas, &nEncomendas, &tam_max_enc);
                carregarDadosPreco(&precos);
        break;
        case 4: guardarDadosCliente(ap_clientes, nElementos);
                guardarDadosEncomenda(ap_encomendas, nEncomendas);
        break;
        case 0: printf("A sair obrigada");
        break;
    }
}
while (opt != 0);
```

Em termos de organização e estruturação o projeto encontra-se organizado por 4 bibliotecas (Header/Source) cujas funcionalidades são:

gestao_utilizadores.h/c - inclui todas as constantes e estruturas utilizadas no projeto, bem a assinatura e declaração das funções respetivas ao cliente, como adicionar, desativar, editar, reativar, pesquisar;

```
// Gestão de Utilizadores
int pesquisa (struct Cliente *ap_clientes, int nElementos, int nif);
void dadosCliente(struct Cliente *ap_clientes, int *pos, int nif);
void adicionarCliente(struct Cliente **ap_clientes, int *nElementos,
        int *tam_max_cli, int nif);
struct Cliente* getCliente(struct Cliente *ap_clientes, int nElementos,
        int nif);
void editarCliente(struct Cliente *ap_clientes, int nElementos, int nif);
void removeCliente(struct Cliente *ap_clientes, int nElementos, int nif);
void ativarCliente(struct Cliente *ap_clientes, int nElementos, int nif);
```

gestao_encomendas.h/c - inclui a assinatura e declaração das funções respetivas às encomendas e aos artigos, devido à abordagem que tomámos de implementar a estrutura Artigo como apontador e subestrutura da Encomenda. As funcionalidades principais desta biblioteca são adicionar, remover, consultar e editar encomendas, mas também tem funcionalidades secundárias para calcular o custo total das encomendas, sendo necessárias para o funcionamento das principais;

```
//Gestão de Encomendas
int pesquisaEncomenda (struct Encomenda *ap_encomendas, int nEncomendas,
        int nif);
void calcularTotal(struct Encomenda *ap_encomendas, int pos, struct Preco *precos);
void dadosEncomenda(struct Encomenda *ap_encomendas, struct Cliente *ap_clientes,
        int *pos, int *nElementos, int nif, struct Preco *precos);
void adicionarEncomenda(struct Encomenda **ap_encomendas, struct Preco *precos,
        struct Cliente *ap_clientes, int *nEncomendas, int *nElementos,
        int *tam_max_enc, int nif);
void consultarEstado(struct Encomenda* ap_encomendas, int nEncomendas, int nif);
void alterarEstado(struct Encomenda *ap_encomendas, int pos);
void removerEncomenda(struct Encomenda **ap_encomendas, int *nEncomendas, int *tam_max_enc);
void alterarEncomenda(struct Encomenda **ap_encomendas, int nEncomendas,
        int nifcliente, struct Preco *precos);
```

menus.h/c - esta biblioteca inclui a assinatura e declaração de todos os menus e submenus como foi referido anteriormente, de modo a facilitar a leitura e legibilidade do código dentro do main.c. Apesar disso não tem só funções respetivas a menus, mas também tem as funções que armazenam dados inseridos pelo cliente e as suas encomendas em ficheiros binários;

```
//Menus, Submenus, Armazenamento de dados
void MenuPrecos(int opt, struct Preco *precos);
void MenuEncomendas(int opt, struct Cliente *ap_clientes, struct Preco *precos,
        struct Encomenda **ap_encomendas, int *nEncomendas, int *nElementos, int *tam_max_enc,
        int nif);
int MenuUtilizadores(int opt, struct Cliente **ap_clientes, int *nElementos,
        int *tam_max_cli, int nif);
void MenuPrincipal(int opt, struct Cliente **ap_clientes,
        struct Encomenda **ap_encomendas, int *nElementos, int *nEncomendas,
        int *tam_max_cli, int *tam_max_enc, struct Preco *precos, int nif);
void guardarDadosCliente(struct Cliente *ap_clientes, int nElementos);
void guardarDadosEncomenda(struct Encomenda *ap_encomendas, int nEncomendas);
void carregarDadosEncomenda(struct Encomenda **ap_encomendas, int *nEncomendas, int *tam_max_enc);
void carregarDadosCliente(struct Cliente **ap_clientes, int *nElementos, int *tam_max_cli);
```

gestao_precos.h/c - inclui a assinatura e declaração das funções respetivas à tabela de preços, como editar e consultar preços. Os preços são influenciados pela morada origem/destino e código-postal definidos na criação do cliente, pelo tipo de serviço de transporte definido nas encomendas e finalmente pelo somatório do peso e volume de todos os artigos

```
//Gestão de Preços
void mudarPrecos(struct Preco *precos);
void consultarPrecos(struct Preco *precos);
```

Para além destas bibliotecas também foi utilizado a biblioteca API_LEITURA fornecida pelos professores, como alternativa à função scanf(); de input de dados e de modo a validar todos os tipos de dados inseridos pelo utilizador e corrigir erros de limpeza de buffer.

```
#include <stdio.h>
#include <stdlib.h>
#include "API_Leitura/API_Leitura.h"
#include "MyLibs/gestao_utilizadores.h"
#include "MyLibs/gestao_encomendas.h"
#include "MyLibs/gestao_precos.h"
#include "MyLibs/menus.h"
```

- ## Funcionalidades implementadas

No decorrer do desenvolvimento deste projeto implementamos várias funções para melhorar a experiência do Diretor e do Cliente.

As funcionalidades que se aplicam ao Diretor são os relatórios do cliente com maior dispêndio e com menor relativamente ao mês e ao ano. O cliente para além do relatório mensal tem também diário e anual para consultar as suas encomendas. O Diretor terá assim um conhecimento do seu melhor e pior cliente mensalmente e anualmente enquanto que o Cliente terá noção dos seus gastos em transporte diariamente e anualmente.

No menu de preços o Cliente pode consultar a tabela de preços que é previamente definida pelo Diretor no seu menu de gestão de preços para além de o mesmo a poder consultar e editar. Por último o Cliente e o Diretor podem ainda aceder ao menu de geração de faturas em que podem gerar faturas e reimprimi-las.

Por fim na gestão de encomendas fizemos uso de bibliotecas de sistema, de modo a registar a data/hora do sistema e transmiti-la juntamente com os dados da encomenda após o diretor a expedir.

```
#include <time.h>
#include <unistd.h>
```

- ## Conclusão

  No desenvolvimento deste projeto conseguimos fortalecer o nosso pensamento lógico de uma maneira prática de acordo com as necessidades de uma empresa em contexto de mercado de trabalho e acima de tudo desenvolver a capacidade de resolução de problemas.

  Ao longo do projeto tivemos que propor algumas funcionalidades ajudando-nos assim a pensar mais profissionalmente e a superar-nos a nós mesmos. Aplicamos os conteúdos dados nas aulas com algumas dificuldades superadas com ajuda do professor e assim obtivemos um melhor entendimento dos conceitos básicos e avançados da linguagem de programação C.