ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Projeto de Laboratório de Programação

Licenciatura em Engenharia Informática

2022/2023

Turma 1

André Eduardo Araújo Faria - 8220787

1. Introdução

Como pedido pela empresa “Móveis para Todos” , este trabalho foi realizado de forma a ser possível ter um gestão de vendedores, de mercados e de comissões . E por isso é possível dividir este trabalho em três elementos.

No primeiro elemento, que se refere a gestão de vendedores, é possível adicionar, editar e remover vendedores, tendo em conta as suas limitações necessárias ao bom funcionamento do programa.

De seguida para a gestão de mercados, acaba por ser bastante parecida ao primeiro elemento, em que também e possível adicionar, editar e remover, mas neste caso mercados.

O último elemento acaba por ser o que mais se destaca neste programa, pois é o mais complexo reunindo os dois elementos anteriormente relatados, sendo assim possível criar as comissões que um vendedor quer fornecer a um mercado.

2. Funcionalidades requeridas

Uma das funcionalidades propostas era o uso de estruturas de dados para os vários tipos de dados solicitados, pois transformam o código muito mais acessível, tanto em termos visuais como para a própria escrita do mesmo, o que acaba por ser bastante vantajoso o seu uso.

Neste trabalho, as estruturas que são usadas para a criação de vendedores, de mercados e de comissões, tem alocação de memoria para iniciar o seu tamanho, e expandem sempre que necessário, para assim pudermos sempre adicionar varias vezes um tipo de dados, sem nos preocuparmos com um limite estipulado.

Nas imagens abaixo, e possível ver o uso destas estruturas e a alocação de memoria para as mesmas.

```c
//Estrutura que representa um vendedor
typedef struct {
    char name[MAX_NAME_INPUT];
    int phone;
    char email[MAX_PHONE_INPUT];
    int seller_code;
    char estado[STATUS_MAX_INPUT];
}Vendedor;


//Representa varios vendedores, utilizando o contador para gerenciar os mesmos.
typedef struct {
    int contador ,tamanho;
    Vendedor *vendedores;
}Vendedores;
```

```c
//Representa varios vendedores, utilizando o contador para gerenciar os mesmos.
typedef struct {
    int contador ,tamanho;
    Vendedor *vendedores;
}Vendedores;

//define um estrutura para um mercado
typedef struct {
    int marketCode;
    char marketName[MAX_MARKET_NAME];
    char estado[STATUS_MAX_INPUT];
}Mercado;
```

```c
//define varios mercados, utilizando o contador para gerenciar os mesmos
typedef struct {
    int contador,tamanho;
    Mercado *mercados;
}Mercados;

//estrura de duas datas
typedef struct{
    int dia,mes,ano;
} Date1,Date2;

//define uma comissao
typedef struct {
    int sellerCode, marketCode;
    float percentage_commision;
    Date1 date_inicio;
    Date2 date_fim;
} Comissao;

//define varias comissoes, utilizando o contador para gerenciar as mesmas
typedef struct {
    int contador, tamanho;
    Comissao *comissoes;
} Comissoes;
```

No header file, também estão contidas a inicialização das várias funções necessárias para o recorrer do programa.

Nas imagens a seguir e possível ver as varias funções utilizadas no programa. Na documentação fornecida no ficheiro .html tem a explicação de todas as funções.

```
//funcoes relativamente ao vendedores

void iniciarVendedores(Vendedores *vendedores);
int procurarVendedor(Vendedores vendedores, int codSeller);
int carregarVendedores(Vendedores *vendedores, char *ficheiro);
void listarVendedores(Vendedores vendedores);
void registrarVendedores(Vendedores *vendedores);
void editarVendedores(Vendedores *vendedores);
void removerVendedor(Vendedores *vendedores,Comissoes *comissoes);
void guardarVendedores(Vendedores *vendedores,char *ficheiro);
void libertarVendedores(Vendedores *vendedores);
```

```
//funcoes relativamente aos mercados

void iniciarMercados(Mercados *mercados);
void listarMercados(Mercados mercados);
int procurarMercado(Mercados mercados, int marketCode);
int carregarMercados(Mercados *mercados, char *ficheiro);
void expandirMercados(Mercados *mercados);
void registarMercados(Mercados *mercados);
void editarMercados(Mercados *mercados);
void eliminarMercados(Mercados *mercados, Comissoes *comissoes);
void libertarMercados(Mercados *mercados);
void guardarMercados(Mercados *mercados, char *ficheiro);
```

```
//funcoes das comissoes

void iniciarComissoes(Comissoes *comissoes);
void imprimirComissoes(Comissoes comissoes);
void listComissoes(Comissoes comissoes);
int procurarComVendedor(Comissoes comissoes, int codeSeller);
int procurarComMercados(Comissoes comissoes, int marketCode);
int carregarComissoes(Comissoes *comissoes, char *ficheiro);
int verificarSobreposicaoDatas(int dia1, int mes1, int ano1);
int registrarComissao(Comissoes *comissoes, Vendedores *vendedores, Mercados *mercados);
void expandirComissoes(Comissoes *comissoes);
void registrarComissoes(Comissoes *comissoes, Vendedores *vendedores, Mercados *mercados);
void libertarComissoes(Comissoes *comissoes);
void guardarComissoes(Comissoes *comissoes,char *ficheiro);
```

3. Funcionalidades propostas

Para este trabalho, foi no pedido para fazer 3 listagens do interesse da empresa.

Com isto fiz 3 listagens que penso serem bastantes uteis a mesma:

1. **Listagem de vendedores**, que consiste na listagem de todos os vendedores existentes no programa, quer estejam em algum mercado com alguma comissão, isto é apresentam -se ativos, ou quer estejam sem nenhuma comissão em algum mercado , encontram -se inativos (também podem ter alguma comissão em algum mercado mas se forem removidos passam para inativos) .

2. **Listagem de mercados,** consiste na listagem de todos os mercados, ativos e não ativos, do programa.

3. **Listagem de Comissões,** esta lista um mercado qualquer, que transmite o código do vendedor, o código do mercado, a percentagem de comissão de um vendedor, e a data em que começou a comissão do vendedor, e a data em que o vendedor já não tem qualquer comissão sobre o mercado.

```c
/**
 * @brief Imprime um vendedor
 * @param vendedores
 */

void listarVendedor(Vendedor vendedor) {
    printf("Nome : %s\n", vendedor.name);
    printf("Email: %s\n", vendedor.email);
    printf("Codigo de vendedor : %d\n", vendedor.seller_code);
    printf("Numero de telemovel: %d\n", vendedor.phone);
    printf("Estado :%s\n\n", vendedor.estado);
}
```

```c
/**
 * @brief Lista os membros de um mercado
 * @param mercado
 */

void listarMercado(Mercado mercado){
    printf("Mercado: %s\n", mercado.marketName);
    printf("Codigo do mercado: %d\n", mercado.marketCode);
    printf("Estado do mercado: %s\n\n", mercado.estado);
}
```

```c
/**
 * @brief Lista as comissoes, com o codigo do mercado, do vendedor, e as datas relativamente
 * ao inicio e ao fim do periodo que o vendedor colocou uma comissao no mercado
 * @param comissao
 */
void listarComissao(Comissao comissao){
    printf("\nCodigo do mercado: %d\n", comissao.marketCode);
    printf("Codigo do vendedor:%d\n", comissao.sellerCode);
    printf("Percentagem da comissao :%f\n", comissao.percentage_commision);
    printf("Inicio da comissao: %02d/%02d/%04d\n",comissao.date_inicio.dia,comissao.date_inicio.mes,comissao.date_inicio.ano);
    printf("Fim da comissao: %02d/%02d/%04d\n\n",comissao.date_fim.dia,comissao.date_fim.mes,comissao.date_fim.ano);
}
```

Estas três funções são executadas nestas outras três para ser possível fazer a listagem de todos os dados.

```c
void listarVendedores(Vendedores vendedores) {
    if (vendedores.contador > 0) {
        for (int i = 0; i < vendedores.contador; i++) {
            printf("Funcionario [%d]:\n", i + 1);
            listarVendedor(vendedores.vendedores[i]);
        }
    } else {
        puts(EMPTY_LIST);
    }
}
```

```c
void listarMercados(Mercados mercados){
    if (mercados.contador > 0){
        for (int i = 0; i < mercados.contador; i++){
            printf("Mercado [%d]:\n", i + 1);
            listarMercado(mercados.mercados[i]);
        }
    } else {
        puts(EMPTY_LIST_MARKET);
    }
}
```

```c
/**
 * @brief Impreme as comissoes existentes na estrura Comissoes atraves da funcao
 * listarComissao
 * @param comissoes
 */

void imprimirComissoes(Comissoes comissoes){
    if(comissoes.contador > 0){
        for (int i = 0; i < comissoes.contador; i++){
            listarComissao(comissoes.comissoes[i]);
        }
    } else{
        puts(EMPTY_LIST_COMMISSIONS);
    }
}
```

4. Estrutura analítica do projeto

O projeto foi desenvolvido a partir das estruturas dos dados, que me permitiu começar a trabalhar na gestão de Vendedores logo em seguida, começando por fazer a função que permite a criação de um vendedor, e em seguida as outras funções, como por exemplo a de listar, de remover e de procurar.

Como a gestão de Mercados não é muito diferente da gestão de vendedores, foi logo seguido a projeção do que tinha de fazer para a mesma. Esta conta coma criação, listagem, e remover que tem algumas diferenças com a função remover da gestão de vendedores.

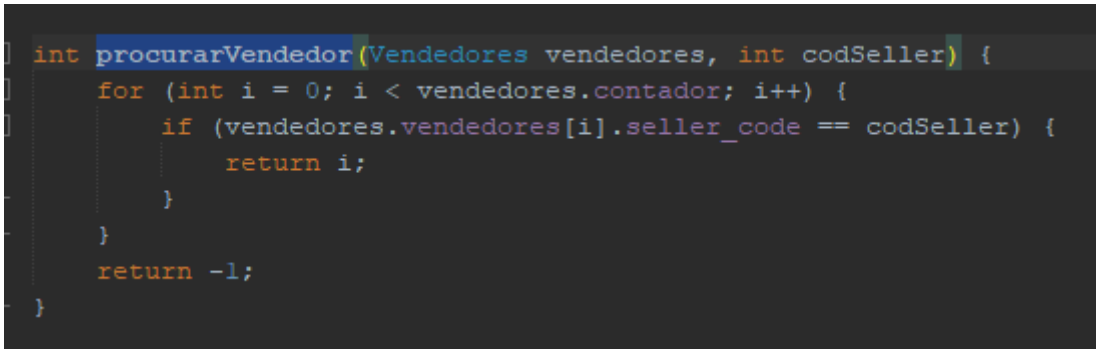
Posteriormente, trabalhei na gestão de comissões, que conta apenas com a função de criar Comissões.

Para finalizar, e para realizar todos os passos pedidos, realizei as funções para carregar e guardar em ficheiros binários as informações para assim haver uma persistência de dados sempre que desejar.

5. Funcionalidades implementadas

Todas as funcionalidades pedidas foram implementadas, e ainda foram realizadas outras funções que auxiliam as que foram pedidas

Exemplo disso são as funções procurar que existem para auxiliar a função registrar, editar, e remover.



```
int procurarVendedor(Vendedores vendedores, int codSeller) {
    for (int i = 0; i < vendedores.contador; i++) {
        if (vendedores.vendedores[i].seller_code == codSeller) {
            return i;
        }
    }
    return -1;
}
```

Como essa, é possível ver varias outras que auxiliam as funções principais deste programa.

6. Conclusão

Em síntese, considero que este projeto seja importante para a minha adaptação e aprendizagem no que toca a programação, pois era necessário ter vários conhecimentos relativamente a linguagem C e penso que, com a realização deste programa foi possível expandir os meus conhecimentos e adaptar o básico que já me havia sido ensinado.

Tive bastantes dificuldades ao desenvolver este programa, mas consegui superar bastante delas e penso que isso seja útil nas minhas próximas façanhas no que toca a programação e a linguagem C.