ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Projeto de Laboratório de Programação

Licenciatura em Engenharia Informática

Licenciatura em Segurança Informática em Redes de Computadores

2022/2023

Grupo 306

Daniela Nóbrega Mendes – 8220342

Índice

1. Introdução

Este relatório tem como função relatar a descrição do trabalho efetuado e as decisões tomadas no desenvolvimento do mesmo na disciplina de Laboratório de Programação, do 1º semestre do curso de Licenciatura de Engenharia Informática.

Este trabalho consiste num programa em linguagem C que gere os funcionários, departamentos e contratos da empresa "Móveis para Todos". Na gestão de funcionários o utilizador poderá criar, editar e eliminar um funcionário e também consegue ver uma lista dos contactos dos funcionários. De seguida, na gestão dos departamentos, o utilizador poderá também criar, editar e eliminar os departamentos. Por último, na gestão de contratos, o utilizador pode atribuir um funcionário a um departamento guardando a data de início e a data de fim do contrato, exportar um ficheiro com os contratos ativos e outro com contratos inativos e por fim pode ver uma listagem de contratos a terminar e outra com funcionários de um certo departamento.

2. Funcionalidades requeridas

Uma das funcionalidades requeridas foi a implementação de estruturas de dados, pois facilita no funcionamento do código. Estas estruturas são um conjunto de uma ou mais variáveis agrupadas sobre um único nome, como se vê nas imagens abaixo (exemplo: Departamento), que facilita a sua referência usada para armazenar e organizar os diferentes tipo de dados.

Nas imagens abaixo podemos ver as estruturas de dados usados no trabalho.

```
typedef struct {
    int codFuncionario;
    int nif;
    int telefone;
    int numUtenteSaude;
    double niss;
    char naturalidade[FUNCIONARIO_MAX_TAM];
    char estado[FUNCIONARIO_ESTADO_MAX_TAM];
    char mail[FUNCIONARIO_MAX_TAM];
    char nome[FUNCIONARIO_MAX_TAM];
    char morada[FUNCIONARIO_MAX_TAM];
} Funcionario;

typedef struct {
    int contador, tamanho;
    Funcionario *funcionarios;
} Funcionarios;
```

```
typedef struct {
    int codDepartamento;
    int codFuncRespomsavel;
    char nomeDepart[DEPARTAMENTO_MAX_TAM];
    char estado[DEPARTAMENTO_ESTADO_MAX_TAM];
} Departamento;

typedef struct {
    int contador, tamanho;
    Departamento *departamentos;
} Departamentos;
```

```
typedef struct {
    int ano_inicio, mes_inicio, dia_inicio;
    int ano_fim, mes_fim, dia_fim;
} Data;

typedef struct {
    int codFuncionario;
    int codDepartamento;
    Data data_carreira;
} Carreira;

typedef struct {
    int contador, tamanho;
    Carreira *carreiras;
} Carreiras;
```

Também foram realizadas funções para o funcionamento do programa identificadas no header onde as estruturas estão localizadas.

```
//Funções usadas para os departamentos
void carregarDepartamentos(Departamentos *departamentos, char *ficheiro);
void registarDepartamentos(Departamentos *departamentos, Funcionarios *funcionarios, char *ficheiro);
void procurarDepartamentos(Departamentos departamentos);
int procurarDepartamento(Departamentos departamentos, int codDepartamento);
void removerDepartamentos(Departamentos *departamentos, Carreiras *carreiras, char *ficheiro);
void listarDepartamentos(Departamentos departamentos);
void libertarDepartamentos(Departamentos *departamentos);
void editarDepartamentos(Departamentos *departamentos, char *ficheiro);
```

```
//Funções usadas para os funcionário
void carregarFuncionarios(Funcionarios *funcionarios, char *ficheiro);
void registarFuncionarios(Funcionarios *funcionarios, char *ficheiro);
void produrarFuncionarios(Funcionarios funcionarios);
int procurarFuncionario(Funcionarios funcionarios, int codFunc);
void removerFuncionarios(Funcionarios *funcionarios, Carreiras *carreiras,char *ficheiro);
void listarFuncionarios(Funcionarios funcionarios);
void libertarFuncionarios(Funcionarios *funcionarios);
void editarFuncionarios(Funcionarios *funcionarios, char *ficheiro);
void listarFuncionariosContactos(Funcionarios funcionarios);
```

3. Funcionalidades propostas

- Gestão de Funcionários

Primeiramente comecei a realização do código da gestão de funcionário que envolve criação, remoção e atualização dos dados dos funcionários.

Para criar um funcionário foi preciso três funções. Uma função verifica se contador e o tamanho são iguais, sendo assim preciso expandir a memória usando a função expandirFuncionarios(), que utiliza a função realloc que permite alterar o número de Bytes redimensionando o bloco de memória. De seguida se o contador for menor que o tamanho da memória registam-se os dados do funcionário e depois é escrito num ficheiro binário.

```
void registarFuncionarios(Funcionarios *funcionarios, char *ficheiro) {
    if (funcionarios->contador == funcionarios->tamanho) {
        expandirFuncionarios(funcionarios);
    }

    if (funcionarios->contador < funcionarios->tamanho) {
        if (registarFuncionario(funcionarios) == -1) {
            puts(FUNC_EXISTE);
        } else {
            registarFuncFX(*funcionarios, ficheiro);
            puts(FUNCIONARIO_REGISTADO_SUCESSO);
        }
    } else {
        puts(FUNC_LISTA_CHEIA);
    }
}
```

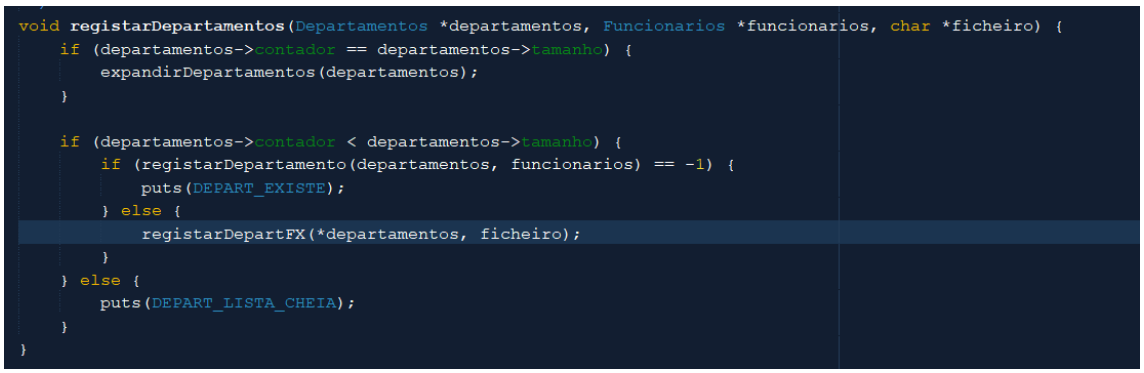
Para remover um funcionário verifica-se se o funcionário existe pelo código que o utilizador insere. Depois da verificação, o registo do funcionário só é eliminado se o funcionário não estiver atribuído a nenhum departamento (gestão de carreiras), caso contrário o estado do mesmo passa para “Inativo”.

```
void removerFuncionarios(Funcionarios *funcionarios, Carreiras *carreiras, char *ficheiro) {
    int i, codFuncionario, numero;

    codFuncionario = obterInt(COD_MIN_FUNCIONARIO, COD_MAX_FUNCIONARIO, OBTER_COD_FUNCIONARIO);
    numero = procurarFuncionario(*funcionarios, codFuncionario);

    if (numero != -1) {
        if (procurarFuncCarreira(*carreiras, codFuncionario) == -1) {
            for (i = numero; i < funcionarios->contador - 1; i++) {
                funcionarios->funcionarios[i] = funcionarios->funcionarios[i + 1];
            }

            apagarDadosFuncionarios(&funcionarios->funcionarios[i]);
            funcionarios->contador--;
            removerFuncFX(*funcionarios, ficheiro);
            puts(FUNCIONARIO_REMOVIDO_SUCESSO);
        } else {
            removerFuncionario(&(*funcionarios).funcionarios[numero]);
            editarFuncFX(*funcionarios, numero, ficheiro);
        }
    } else {
        puts(FUNC_NAO_EXISTE);
    }
}
```

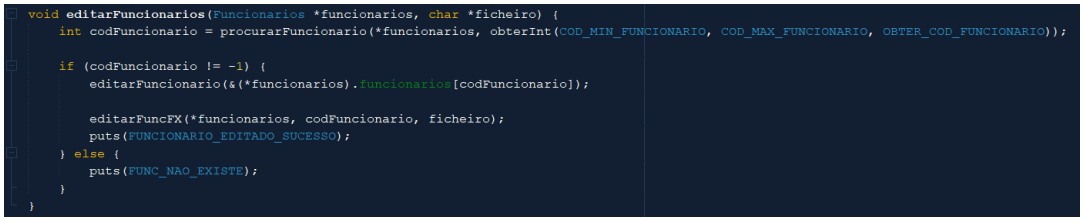
- Gestão de Departamentos

As funções da gestão de departamentos são praticamente iguais às da gestão de funcionários.

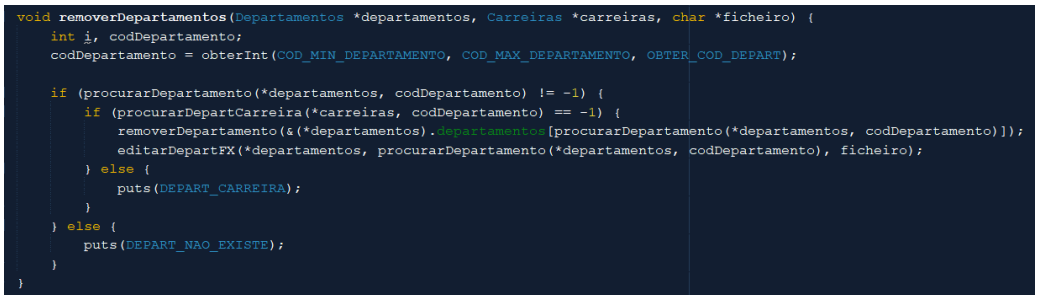
A criação de um departamento é igual à criação de um funcionário. Verifica-se se o contador é igual ao tamanho, sendo que se for usa-se a função realloc para redimensionar a memória. De seguida o utilizador insere os dados do departamento que ficam guardados na estrutura de dados Departamento e é registado também no ficheiro binário.

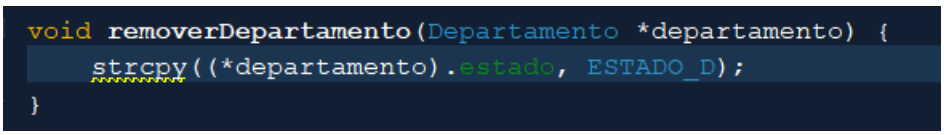


Ao editar um funcionário, também se verifica se o funcionário existe pelo código que o utilizador insere e caso ele exista inicia-se a sua atualização com a função editarFuncionario() e escreve-se no ficheiro binário o seu registo com a função editarFuncFX().



Para remover um departamento simplesmente muda-se o seu estado para “Inativo”. Caso o departamento esteja atribuído a funcionários (gestão de Carreiras) o estado fica “Ativo”, ou seja não muda nada e aparece uma mensagem a informar.





- Gestão de Carreiras

Na gestão de carreiras atribui-se um funcionário a um departamento. O início do código é igual ao criar um funcionário ou departamento, ou seja se o tamanho e o contador da estruturas de dados Carreira for igual chama-se a função expandirCarreiras() que usa o realloc para redimensionar a memória. De seguida regista-se o funcionário, o departamento e a data do início e fim do contrato. Por último regista-se os dados num ficheiro binário na função registarCarreiraFX().

```
void registarCarreiras(Carreiras *carreiras, Funcionarios *funcionarios, Departamentos *departamentos, char *ficheiro) {
    if (carreiras->contador == carreiras->tamanho) {
        expandirCarreiras(carreiras);
    }

    if (carreiras->contador < carreiras->tamanho) {
        if (registarCarreira(carreiras, funcionarios, departamentos) == -1) {
            puts(CARREIRAS_EXISTE);
        } else {
            registarCarreirasFX(*carreiras, ficheiro);
        }
    } else {
        puts(CARREIRAS_LISTA_CHEIA);
    }
}
```

- Outras Funções
    - *Listagens*

No enunciado era pedido para implementar três listagens. Destas três listagens implementei mais três. Estas listagens dividi pela gestão dos funcionários, departamentos e das carreiras.

- Gestão de Funcionários:
    - Uma das listagens consiste na listagem do contacto dos funcionários registados, como o telemóvel e o e-mail
    - Outra delas foi uma lista dos funcionários registados
- Gestão de Departamentos
    - A terceira foi uma lista dos departamentos registados
- Gestão de Carreiras
    - A quarta foi uma lista de funcionários por departamento. Nesta listagem o utilizador insere o código do departamento que pretende, sendo depois possível ver os funcionários do departamento
    - A quinta é uma listagem de contratos que estejam a terminar. O tempo gerido para saber quando está a terminar são trinta dias de diferença, ou seja, os contratos que tenham a data do fim a acabar em 30 ou menos dias aparecerem nesta lista.
    - Por último, temos a lista de todos os contratos registados

    - *Exportar Ficheiros*

Para exportar os dois ficheiros usou-se uma função dataMaior() para verificar a data do fim dos contratos que estão ligados à gestão de Carreiras e a data atual (data do computador). De seguida, registou-se os dados dos contratos no ficheiro de texto.

4. Estrutura analítica do projeto

Na realização do trabalho separou-se o mesmo em cincos partes. Primeiramente começou-se na realização da gestão de funcionários, que consiste na criação, atualização e remoção do registo de funcionários, sendo que se um funcionário estiver atribuído a um departamento, com data de fim expirada, o estado do mesmo seria “Inativo”.

Seguidamente, realizou-se a gestão de departamentos, que consiste nas mesmas funções que a gestão de funcionários, menos na função de remoção de um departamento, uma vez que se existir funcionários a trabalhar nesse departamento a remoção do mesmo é impossível. Só é possível remover um departamento, caso este não tenha funcionários a trabalhar no mesmo, mudando o estado para “Inativo”.

De seguida, elaborou-se a gestão das carreiras, que regista um funcionário e um departamento, juntamente com as datas de início e fim do contrato.

A seguir, implementou-se três listagens, das quais duas estão situadas na gestão de contratos e a outra está na gestão de funcionários. Numa listagem consegue-se visualizar os funcionários de um departamento pedido ao utilizador, na segunda listagem vê-se os contratos a terminar em trinta dias e a última apresenta a lista de funcionários com os respetivos contactos (nº de telemóvel e email).

Por último, criou-se o código para exportar dois ficheiros de texto com registos dos contratos dos funcionários, sendo que um são os contratos com a informação corrente, como os departamentos existentes e os respetivos funcionários e no outro são os contratos com informação antiga, como os departamentos existentes, mas com funcionários que em algum período no tempo estiveram lá afetos.

5. Funcionalidades implementadas

As funcionalidades implementadas foram feitas perante as funcionalidades pedidas pelo enunciado inicialmente. Todas elas foram feitas perante o que o enunciado pedia.

As estruturas de dados foram guardadas num header com o nome de funcionalidades, juntamente com as macros necessárias para o funcionamento do programa. Neste ficheiro também tem guardado as funções utilizadas para as funcionalidades propostas.

6. Conclusão

Em suma, este projeto ajudou-me a perceber mais sobre a linguagem C, estruturas de dados, trabalhar com ficheiros entre outras funções. Posso dizer também que me ajudou em perceber a matéria não só para esta disciplina como também para a disciplina de Fundamentos de Programação.

Sinto que com este projeto consegui superar algumas dificuldades que tinha ao desenvolvê-lo, mas ainda há muito para melhorar.