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 105

Carlos Fernando Teixeira Leite – 8200377

João Pedro Ferreira Teixeira – 8200489

Miguel Ângelo Barbosa Ribeiro – 8170538

Índice

1. Introdução 3
2. Funcionalidades requeridas 4
Clientes 4
Produtos 4
Produção 4
Ficheiros 4
Encomendas 4
3. Funcionalidades propostas 5
4. Estrutura analítica do projeto 7
5. Funcionalidades implementadas 8
Clientes 8
Produtos 8
Materiais 8
Produção 8
Gestão de encomendas 8
Ficheiros 9
Encomendas 9
6. Conclusão 9

1. Introdução

No âmbito da unidade curricular de Laboratório de Programação, foi proposto pelos docentes da mesma desenvolver o presente trabalho com o objetivo de permitir, compreender e dominar os conhecimentos teóricos e práticos sobre algoritmia e programação na linguagem C, na qual são abordados em conjunto com a unidade curricular de Fundamentos da Programação.

O objetivo principal do trabalho é criar um programa que permita o registo de encomendas. Por consequente foram identificados dois perfis: o Cliente e Administrador onde o perfil Administrador ficará responsável pela gestão de clientes, produtos, produção e ficheiros. Para completar as necessidades da gestão de produtos irá ser adicionado ao perfil cliente a gestão de materiais. O perfil Cliente ficará responsável pelo processo de registo de encomendas.

Para suceder no objetivo do projeto será necessário permitir a persistência de dados ao longo da sua utilização assim como a aplicação da memória dinâmica sempre que necessário no armazenamento de informação.

Para o desenvolvimento do programa o grupo de trabalho irá utilizar as seguintes ferramentas:

- CLion, IDE principal usado para edição, desenvolvimento compilação e debug do programa;
- NetBeans, IDE secundário;
- GitHub, será utilizado para o armazenamento do projeto, como também para uma eficaz incorporação de alterações entre os membros do grupo.
- Discord, como ferramenta essencial à comunicação e cooperação entre os membros da equipa;
- Doxygen, documentação do projeto;
- Valgrind, framework para gerir fugas de memória dinâmica.

Todo o procedimento mencionado anteriormente será descrito ao pormenor nos próximos tópicos do atual relatório.

2. Funcionalidades requeridas

Na seguinte secção irão ser descritas todo um conjunto de funcionalidades para a consideração de um trabalho válido, e correspondentes estruturas implementadas.

Clientes

- Criar cliente;
- Editar cliente;
- Remover cliente;
- Listar clientes (incluindo os inativos);

Produtos

- Criar produto;
- Editar produto;
- Remover produto;
- Listar produtos e materiais (incluindo os removidos);

Produção

- Listar materiais que satisfazem as encomendas numa dada data

Ficheiros

- Importar ficheiro
- Exportar ficheiro

Encomendas

- Criar encomenda;

3. Funcionalidades propostas

No que toca a funcionalidades propostas pelo grupo, foram desenvolvidas as seguintes listagens:

- de clientes;
- de produtos e respetivos materiais;
- de produtos removidos;
- de materiais;
- de encomendas;
- de encomendas de um cliente;
- de encomendas canceladas;

Após uma rigorosa avaliação do domínio do projeto, foi decidido implementar funcionalidades capazes de gerir os materiais de forma mais dinâmica, desta forma é possível criar, editar, remover e listar os materiais inseridos no sistema, seja manualmente ou através de um ficheiro .csv.

Para concretizar o armazenamento da informação na memória referente às listagens anteriormente referidas foi necessária a utilização da memória dinâmica assim como também a definição das seguintes estruturas:

Clientes

```
typedef struct {
    int code;
    char name[256];
    char address[256];
    int nif;
    char country[64];
    int removed;
} Client;

typedef struct {
    Client *client;
    int count; // Representa o espaço total ocupado e a posição livre
    int size;  // Tamanho total do array
} ClientsArray;
```

Produtos

```
typedef struct {
    char code[6];
    char name[256];
    char dimensions[11];
    int price;
    int removed;         //Não Removido = 0 ou Removido = 1
    ProductMaterialsArray *productMaterialsArray;  //Lista de Materiais para
cada produto
} Product;

typedef struct {
    Product *product;
    int count;     // Representa o espaço total ocupado e a posição livre
    int size;      // Tamanho total do array
} ProductsArray;
```

Materiais

```
typedef struct {
    char code[6];    //Código de materiais
    int quantity;    //Quantidade de material
    char unit[3];
    char description[128];
} MaterialInProduct;

typedef struct {
    MaterialInProduct *materialInProduct;
    int size;
    int count;
} ProductMaterialsArray;  //Lista de Produtos com Materiais

typedef struct {
    char code[6];
    char description[256];
    char type[5];
} Material;

typedef struct {
    Material *materials;
    int count;         // Representa o espaço total ocupado e a posição livre
    int size;          // Tamanho total do array
} MaterialsArray;
```

Encomendas

```
typedef struct {
    int day;
    int month;
    int year;
} Date;

typedef struct {
    char code[6];   //Codigo do Produto
    int quantity;   //Quantidade de Produtos a encomendar
    float productTotal;
} ProductInOrder;

typedef struct {
    ProductInOrder *productInOrder;
    int size;
    int count;
} OrderProductsArray;  //Lista de Produtos numa encomenda

typedef struct {
    int clientCode;
    int orderCode;
    OrderProductsArray *orderProductsArray;
    Date deliveryDate;
    float orderTotal;
    int canceled;
} Order;

typedef struct {
    Order *order;
    int size;
    int count;
} OrdersArray;
```

4. Estrutura analítica do projeto

A nível de estruturamento analítico do projeto o primeiro passo realizado foi a definição das estruturas para cumprimento dos requisitos exigidos no enunciado do presente projeto e também para o grupo de trabalho conseguir uma orientação do trabalho a desenvolver. As estruturas e tipos de dados foram a primeira etapa a executar, seguindo-se por toda a implementação de funções incluindo a alocação de memória dinâmica para as estruturas anteriormente desenvolvidas. Por fim foi realizada a importação e exportação de ficheiros de forma a garantir a persistência de dados do programa.

O trabalho foi desenvolvido sempre de forma colaborativa através da plataforma Discord e desta forma foi possível que os membros do grupo se mantivessem a par da sua progressão.

5. Funcionalidades implementadas

A presente secção é o reflexo de uma retrospetiva entre as funcionalidades implementadas que eram pretendidas para o funcionamento válido do projeto e as funcionalidades que poderiam estar completas, que não foram muito bem conseguidas ou que ficaram por concretizar.

Clientes

Nos clientes é requisitado ao cliente que insira toda a informação correspondente ao mesmo, como se pode comprovar anteriormente na respetiva estrutura. Para editar os dados cliente o grupo de trabalho recorreu à mesma função que lê o cliente para efetuar a atualização das informações.

A atualização de dados do cliente foi realizada com recurso à função de leitura de clientes.

A listagem de clientes ficou funcional de modo imprimir todos os clientes e informar os que tiverem ativos (0) e inativos (1) no sistema. Caso o utilizador pretenda remover clientes o mesmo é possível também devido à implementação de um algoritmo que reconhece se um cliente tem encomendas.

Em relação à utilização de memória dinâmica a mesma é usada para alocar a lista de clientes como também em algumas variáveis do cliente nomeadamente nome, morada e país.

Produtos

Para a inserção de informação dos produtos é necessário fornecer código, nome, dimensões e preço. Uma vez feita uma análise do trabalho a realizar o grupo teve a perceção de que os produtos iriam conter os Materiais o que significa que na função que lê o produto houve a necessidade de implementar uma função que adicione materiais ao produto.

A nível de memória dinâmica esta foi implementada para o array de produtos e o array de materiais.

Materiais

As informações a ler para o material é o código, descrição e o tipo de unidade. Nos materiais a estrutura para desenvolver o código é semelhante à dos clientes, no entanto, quando o utilizador quiser remover um material este é removido do programa de forma definitiva. A nível de memória dinâmica esta é alocada para o array de materiais e respetiva estrutura inicializando todos os valores a zero com a função calloc().

Produção

As implementações das funcionalidades da produção em todo o trabalho foram aquelas que o grupo de trabalho teve dificuldades em implementar.

Gestão de encomendas

A gestão de encomendas está desenvolvida em conjunto com as encomendas e aqui estão disponibilizadas algumas das listagens que o grupo considerou essenciais para comprovar o correto funcionamento das encomendas.

Ficheiros

Para as funcionalidades nos ficheiros foi possível desenvolver funções para importar os produtos e materiais de um ficheiro .csv assim como também exportar as encomendas, clientes, produtos e materiais.

Encomendas

Nas encomendas é possível encontrar um array de encomendas que guardará a encomenda. Dentro da estrutura da encomenda foi adicionada uma variável que aponta no sentido de um outro array que guardará os produtos.

Uma vez executada a ordem de encomenda será pedido ao cliente que se identifique e insira logo em seguida a data que pretende para a entrega da encomenda. A seguir é pedido que insira o produto e quantidade do mesmo a encomendar.

Em termos de memória dinâmica esta foi aplicada de forma semelhante aos relatos anteriores.

6. Conclusão

Concluindo o projeto o administrador consegue desde registar, editar, remover e listar os clientes, produtos e materiais. Quanto à gestão de produção foi a única funcionalidade que o grupo de trabalho não conseguiu implementar com sucesso.

Quanto aos ficheiros as funcionalidades desenvolvidas ficaram a funcionar de forma eficaz.

A nível de persistência de dados, memória dinâmica e documentação do projeto todas estes objetivos ficaram funcionais e acima de tudo não existe qualquer fuga de memória no que diz respeito à alocação de memória dinâmica.