Estruturas de Dados e Algoritmos

Projeto prático 1 - 2023/2024

(15% da avaliação da UC)

1. Objetivos

O objetivo do projeto é o desenvolvimento de um programa em C++ que simule o funcionamento do "**ArmazemEDA**". O sistema deverá implementar/simular todas as funcionalidades de um armazém de :

- Stock de peças
- Novas Encomendas
- Atendimento a Clientes
- Promoções

Espera-se que o projeto seja desenvolvido utilizando tipos de dados definidos pelos alunos (structs) e vetores (arrays) dinâmicos.

- ➔ **A utilização da classe/biblioteca vector não é permitida.**
- ➔ **A utilização de listas ligadas não é permitida.**
- ➔ **A utilização das classes/bibliotecas Stack e Queue não é permitida.**

O armazém tem um número variável de secções. Uma Secção é uma localização no armazém onde são armazenados produtos de uma determinada categoria (por exemplo Tillers), além disso cada Secção tem ainda um ID único, um registo do número de série de todos os produtos vendidos e uma capacidade máxima. As peças em armazém são identificadas pela sua marca, categoria e preço, por exemplo, Makita Polishers 100. O Armazém deverá manter um registo da faturação total, e faturação por Secção.

O armazém possui também uma "Lista de Chegada" onde todas as peças são colocadas quando chegam ao armazém. Para facilitar a organização, o armazém implementou uma regra onde as peças são arrumadas por ordem de chegada. Contudo, foi também implementada uma regra, que obriga que peças da mesma marca fiquem em posições seguidas desta Lista. Por exemplo, se já existem peças da marca Bosch na Lista de Chegada, e chega uma nova encomenda que inclui peças dessa marca, estas serão inseridas na lista logo após a última peça da sua marca. As peças ficam na Lista de Chegada até existir espaço na Secção indicada. A Lista de Chegada tem uma capacidade máxima de 50 peças, no caso da Lista de Chegada estar cheia as novas encomendas deverão ser ignoradas até haver espaço suficiente.

2. Inicialização

Quando o programa inicializar deverão ser considerados os seguintes pontos:

- O número de Secções da oficina deverá ser calculado, a oficina deverá ter entre 7 e 10 Secções.
- A capacidade de cada Secção deverá ser calculada, esta capacidade será um valor aleatório entre 3 e 6. Diferentes Secções poderão ter tamanhos diferentes.
    - O conteúdo necessário para a inicialização de cada Secção é definido em **2.1**
- Na inicialização do programa deverão ser criadas 10 peças (aleatórias) que serão colocados na Lista de Chegada
    - A criação de uma peça é definida em **2.2**

2.1 Inicialização das Secções.

O Armazém será então um conjunto de Secções. Na sua inicialização a cada Secção deverá ser atribuída uma categoria, ID e capacidade. O id deverá ser um carácter sequencial a partir de 'A'.

2.2. Inicialização das peças

A marca e categoria de uma peça deverão ser retirados aleatoriamente dos ficheiros marca.txt e categorias.txt fornecidos. Como referido anteriormente, após a inicialização todas as peças são colocadas na Lista de Chegada. Para efeitos da simulação cada peça terá ainda uma % de venda, que deverá ser um valor aleatório entre 5% e 50%. Este valor indica a probabilidade de uma determinada peça ser vendida num determinado dia. O número de série da peça é um inteiro aleatório entre 1000 e 9999. O preço da peça deverá ser um valor inteiro múltiplo de 5 entre 10 e 900 euros.

2.3 Funcionamento

Após a inicialização o processamento deverá ocorrer por ciclos. Cada ciclo é iniciado pelo utilizador ao pressionar a tecla 's' seguida de enter, cada ciclo simula um dia no **ArmazemEDA**, em cada ciclo deverão acontecer os seguintes passos (por esta ordem):

1. Venda de peças:
    a. A cada ciclo cada peça (nas secções) terá uma probabilidade de ser vendida (ver Secção 2.2). Caso esta probabilidade seja cumprida, a peça é removida da Secção, o seu preço é registado no total

de faturação do Armazém, e da Secção. E o seu número de série é adicionado ao registo de vendas da Secção.

2. Criação de 5 novas peças aleatórias, que serão adicionadas à Lista de Chegada.
    a. Só poderão ser geradas peças de categorias que possam ser adicionadas ao armazém, ou seja, que possam ser colocadas nas secções.
3. Remoção de 8 peças da Lista de Chegada que serão adicionadas ao armazém.
    a. **NOTA:** Poderão existir casos em que a 1ª peça na Lista de Chegada não consiga ser colocada em qualquer Secção, pois todas as secções da sua categoria estão completas. Nesse caso, a peça fica na fila e o sistema deverá tentar colocar a peça seguinte na Lista de Chegada.
    b. **NOTA II:** Poderão existir situações em que será impossível adicionar 8 peças às secções. Por exemplo, em cenários em que as secções estão todas cheias. Ou quando existe espaço nas secções mas não existe nenhuma peça da categoria correspondente na Lista de Chegada. Nesses casos poderão ser inseridas menos de 8 peças.

3. Gestão

Além da simulação definida acima, o sistema deve fornecer algumas operações de gestão. Estas operações deverão ser fornecidas ao utilizador após pressionar a tecla 'g' + enter. As operações de gestão serão:

3.1 Venda Manual

Para além da simulação apresentada acima, deverá ser possível realizar vendas manualmente. Para isso o utilizador deverá indicar qual a Secção e o nome do produto que pretende comprar. O preço da venda deverá ser adicionado aos registos de facturação do Armazém e da Secção. Imediatamente após a venda, deverá ser retirada uma peça da Lista de Chegada (caso exista) e colocada no lugar da peça vendida no seu setor.

3.2 Implementar promoção

Deverá ser possível ao utilizador adicionar uma promoção a uma determinada Secção do armazém. Para isso o utilizador deverá indicar quantos dias (ciclos) a promoção durará. E uma % de desconto a aplicar às peças nessa Secção. É importante ter em atenção que durante a promoção poderão ser adicionados novos produtos à Secção, estes deverão também ter o desconto promocional. Um produto em promoção terá mais 15% de probabilidade de ser vendido num determinado ciclo.

3.3 Alterar Categoria

Deverá ser possível alterar a categoria de uma determinada Secção para qualquer categoria a indicar pelo utilizador. Após esta alteração, todos os produtos pertencentes Secção deverão ser removidos (não vendidos), e novos produtos da categoria escolhida poderão entrar nesta Secção do armazém

NOTA III: Podem existir casos em que a categoria introduzida para a Secção seja nova para o armazém. Nesse caso esta categoria, deverá passar a ser uma possibilidade aquando da geração de novas peças a cada ciclo.

3.4 Adicionar Secção

Deverá ser possível adicionar uma Secção ao armazém, o sistema deverá pedir ao utilizador os dados necessários para a criação de uma nova Secção, ID único, capacidade máxima e categoria. A partir do momento da adição da Secção esta deverá ser considerada nos próximos ciclos do Armazém.

3.5 Gravar Armazém

O programa deverá permitir ao utilizador gravar o estado atual do armazém. Esta gravação deverá ser implementada utilizando um ou mais ficheiros. Na prática, o programa deverá guardar a totalidade dos dados relevantes para o armazém, por exemplo, os dados da Lista de Chegada, Secções, Peças e Faturação.

3.6 Carregar Armazém

O programa deverá permitir ao utilizador o carregamento de um estado do Armazém previamente gravado (ver ponto 3.6). Esta funcionalidade deverá apagar a simulação atual do armazém, que será substituído pelo estado carregado dos ficheiros. Este carregamento deverá também ser possível ao passar o caminho do(s) ficheiro(s) por argumento na execução do programa.

3.7 Imprimir Armazém

O sistema deverá permitir a um utilizador imprimir uma lista de todas as peças presentes no Armazém, secções + Lista de Chegada. Esta lista deverá ser impressa:

3.7.1. Por ordem alfabética da marca da peça

3.7.2. Por preço

Estas opções deverão ser apresentadas num sub-menu

4. Visualização

O programa deverá seguir o modelo abaixo na sua apresentação, toda a informação apresentada abaixo é meramente indicativa):

```
                ****************************************
                *** Armazém EDA | Total Faturação 2450€ ***
                ****************************************
Secção A | Categoria Toaster-Ovens | Capacidadde :5 | Quantidade :1 | Faturação 30
       Crosley | Toaster-Ovens | 6783 | 250 €
Secção B | Categoria Workbenches | Capacidadde :5 | Quantidade :4 | Faturação 0
       Ames | Workbenches | 5478 | 290 €
       Craftsman | Workbenches | 9638 | 270 €
       Emco | Workbenches | 2089 | 190 €
       Triumph | Workbenches | 9762 | 350 €
Secção C | Categoria Boating | Capacidadde :5 | Quantidade :3 | Faturação 50
       Niagara | Boating | 8333 | 110 €
       Regalware | Boating | 2287 | 40 €
       Snapper | Boating | 1441 | 490 €
( Repetir para todas as secções )
       ****************************************
       **********  Lista de Chegada  ************
       PEC | Tillers | 1970 | 25 €
       PEC | Washer-Dryer | 2999 | 265 €
       PEC | Welders | 6974 | 250 €
       Lifestyler | Toasters | 9791 | 415 €
       Lifestyler | Generators | 2675 | 310 €
( Repetir para o Resto da Lista de Chegada )

Dia (s)eguinte  *********** (g)estão
Seleccione a sua opção:
```

Figura 1 : Exemplo de apresentação do programa

Apresentação do menu de gestão deverá ser a seguinte:

```
***** Bem Vindo Gestor *****
(1).Venda Manual
(2).Promocao
```

```
(3).Alterar categoria
(4).Adicionar seccao
(5).Gravar Armazem
(6).Carregar Armazem
(7).Imprimir Armazem
Seleccione a sua opcao:
```

Figure 2: Apresentação do menu de Gestão

Poderão ainda ser criados outros sub-menos, por exemplo, para pedir ao utilizador informações necessárias para as tarefas de gestão, como a marca ou ID de um peça ou secção.

Durante a interação com o utilizador os alunos poderão ignorar a utilização de caracteres especiais como acentos.

5. Documentação do trabalho

Todas as funções terão de estar devidamente documentadas de acordo com o *template* automático gerado pelo Clion. Nesta documentação deverão estar claramente identificados os argumentos, e tipo de retorno da função. Bem como uma breve descrição do funcionamento da função:

```
/**
* Função que recebe um array de inteiros e retorna a soma dos elementos do
array recebidos
* @param v - array cujo os valores serão somados
* @param tamanho - tamanho do array recebido
* @return - soma do valores de v
*/
float somarElementos(int v[], int tamanho){
   float soma=0;
   for(int i=0;i<tamanho;i++){
       soma = soma+v[i];
   }
   return soma;
}
```

Além da documentação do código, juntamente com o projeto deverá ser entregue um documento que indica a divisão das tarefas de implementação pelos membros do grupo

a. A escrita da documentação não é considerada uma tarefa de implementação
b. Membros de grupo sem tarefas atribuídas/realizadas reprovam o projeto prático da disciplina

6. Entrega

As entregas do projeto serão finalizadas no moodle, na data da entrega será criado um formulário no moodle em que cada grupo deverá submeter um ficheiro .zip ou .rar com o relatório e todo o código e ficheiros necessários para a execução/avaliação.
O nome do ficheiro entregue deverá ser o mesmo do número do grupo, por exemplo Grupo1.zip. Para os alunos que desenvolveram o projeto no Visual Studio, terão que eliminar a pasta oculta .vs na raiz do projeto antes de comprimir o projeto em .zip ou .rar.

6.1 Datas Entregas

1ª Entrega: A primeira entrega está agendada para o dia 16 de Março de 2024, nesta entrega deverão ser entregues as seguintes funcionalidades:

- 2, 2.1 e 2.2 e todos os menus do programa.
- A primeira entrega é **obrigatória** e poderá influenciar até 15% da avaliação final

- A não implementação de cada requisito irá prejudicar a avaliação **final** em 5%, até um total de 15%

- Nesta entrega espera-se uma implementação mínima mas funcional dos requisitos acima. Nesta fase poderão ser utilizados valores *hardcoded*, para substituir as leituras dos ficheiros.

2ª Entrega: A segunda entrega está agendada para dia 2 de Abril de 2024, e deverão ser entregues todas as funcionalidades.

- Nesta entrega, se entenderem, os grupos poderão implementar alterações às funcionalidades entregues na primeira entrega.

Todas as entregas do projeto entregue deverão compilar sem problemas, projetos que não executem serão ignorados. É preferível entregar um projeto mais simples mas funcional, do que um projeto complexo que não seja possível testar

6.2 Defesas dos projetos

Na semana da segunda entrega serão agendadas reuniões com os membros de cada grupo de forma a aferir a contribuição de cada membro para o projeto. Esta reunião pode influenciar a nota individual até 100%, na prática isto significa que no mesmo grupo poderão haver colegas avaliados em 18 enquanto que outros poderão ter uma avaliação abaixo da nota mínima (**o que implica a reprovação da disciplina**). De forma a facilitar esta avaliação individual, é **obrigatório** que todos os grupos mantenham o registo de **todas** as tarefas desenvolvidas no âmbito do projeto (ver ponto 6 do relatório).

6.3 Critérios de avaliação

- Definição de estruturas de dados adequadas;
- Cumprimento dos objetivos;
- Qualidade do código desenvolvido;
- Qualidade de execução do programa desenvolvido;
- Relatório e documentação do código;
- Apresentações/discussões.

7. Notas implementação

Durante a inicialização do programa existem várias variáveis (por exemplo quantidade de secções, e capacidade das mesmas) que serão aleatórias.
Para implementar este comportamento espera-se a utilização da função **`srand`**, a utilização deverá seguir os seguintes passos:

- **No "main" do programa** : importar as bibliotecas **`<stdlib.h>`** e **`<time.h>`**
  - criar o "seed" para a função rand através da instrução **`srand (time(NULL));`**
- Para o cálculo de um número aleatório (em qualquer ficheiro)
  - importar o **`<stdlib.h>`**
  - Utilizar a função **`rand()%NÚMERO`** para calcular um valor aleatório entre 0 e um (**NÚMERO -1**) definido

```
/* valor aleatório entre 1 e 10: */
  valor = rand() % 10 + 1;
```

Cada grupo é livre de implementar soluções para todos os restantes detalhes de implementação, não definidos neste enunciado.

8. Código de ética e honestidade académica

Nesta disciplina, espera-se que cada aluno subscreva os mais altos padrões de honestidade académica. Isto significa que cada ideia que não seja do aluno deve ser explicitamente creditada ao(s) respetivo(s) autor(es). O não cumprimento do disposto constitui uma prática de plágio. O plágio inclui a utilização de ideias, código ou conjuntos de soluções de outros alunos ou indivíduos, ou qualquer outra fonte para além dos textos de apoio à disciplina, sem dar o respetivo crédito a essas fontes. A menção das fontes não altera a classificação, mas os alunos não deverão copiar código de outros colegas, ou dar o seu próprio código a outros colegas em qualquer circunstância. De notar que a responsabilidade de manter o acesso ao código somente para os colegas de grupo é de todos os elementos.