Estruturas de Dados e Algoritmos

Projeto prático 1 - 2019/2020

(15% da avaliação da UC)

1. Objetivos

O objetivo do projeto é o desenvolvimento de um programa em C++ que simule o funcionamento da "**VendingEDA**". O sistema deverá implementar/simular todas as funcionalidades relativas ao funcionamento de uma máquina de vending. De uma forma sumarizada espera-se que o programa implemente funcionalidades para:

- Comprar produtos da máquina
- Repor stock na máquina
- Calcular e devolver troco
- Inserir troco
- Alterar posição dos produtos

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 premitida.

A máquina de vending é caracterizada pelo número de slots. Cada slot é identificado por uma letra (entre a e z) em que o slot superior esquerdo é representado pela letra A e o inferior direito pela letra Z.

A máquina de vending tem também uma caixa, onde são guardados os fundos introduzidos pelos clientes. Esta caixa contém uma determinada quantidade de moedas de 5, 10, 20, 50 cêntimos e de 1 e 2 euros.

Cada slot contém um número variável de produtos. O slot é caracterizado pelo número máximo de produtos possíveis (tamanho) e o número de produtos a cada momento. Cada slot contém apenas produtos do mesmo tipo.

Um produto é caracterizado pelo seu nome, fabricante e preço.

2. Inicialização

Quando o programa inicializar deverão ser considerados os seguintes pontos:

- O tamanho da máquina deverá ser calculado; a máquina deverá ter um número aleatório de slots, entre 9 e 12 slots.
- A capacidade de cada slot deverá ser calculada; esta capacidade será um valor aleatório entre 5 e 10. Diferentes slots poderão ter tamanhos diferentes.
- A caixa deverá ser inicializada com um valor aleatório entre 10 a 20 moedas, para moedas dos tipos: 5,10,20 e 50 cêntimos e 1 e 2 euros. Diferentes tipos de moedas poderão ser inicializadas com quantidades diferentes.

2.1 Inicialização dos produtos.

Quando o programa inicia a máquina deverá ser inicializada de forma a ficar completa. Os produtos e preços deverão ser retirados aleatoriamente dos ficheiros precos.txt e produtos.txt disponíveis aqui. Neste estado inicial não poderá haver produtos repetidos em slots diferentes. Além disso, cada slot só pode conter um tipo de produto e todos deverão partilhar o mesmo preço.

2.2 Visualização

O programa deverá seguir o modelo abaixo na sua apresentação (os preços e nomes de produtos são meramente indicativos):

```
Slot : A
Produto: Cheetos
Preço : 1 €
Quantidade: 4   | Capacidade : 5
---------------------------------
Slot : B
Produto: Mentos
Preço : 0.75 €
Quantidade: 4   | Capacidade : 5
---------------------------------
Slot : C
Produto: Twix
Preço : 1.55 €
Quantidade: 4   | Capacidade : 5
---------------------------------
 (... Repetir para todos os slots ..)
(c)liente  ********** (f)uncionário
Seleccione a sua opção:
```

Figura 1 : Exemplo de apresentação do programa

3. Utilizadores

O sistema deverá suportar utilizadores de 2 tipos, o cliente que quer comprar um produto, e o funcionário que gere diversos fatores relacionados com o funcionamento da máquina de vending. Antes de qualquer operação o utilizador do programa deverá pressionar a tecla 'f' ou 'c' + 'enter' para indicar o tipo de utilizador pretendido (ver Figura 1). Detalhes sobre as funcionalidades permitidas a cada utilizador serão explicadas nas próximas seções.

3.1 Cliente

O cliente representa um cliente normal de uma máquina de vending. Este utilizador tem como objetivos, comprar um produto, inserir moedas e receber o troco restante. As opções disponíveis deverão ser apresentadas após a selecção do utilizador através da tecla 'c'.

3.1.1 Compra

Quando um cliente deseja comprar um produto o programa deverá pedir ao utilizador o slot (identificado pela sua letra) e depois o troco. O troco deverá ser introduzido moeda a moeda. Esta funcionalidade deverá seguir a apresentação abaixo:

```
*** Bem vindo Cliente ***
Introduza o código do produto: B
Escolheu : Mentos
Introduza a quantidade de moedas de 2 euros:0
Introduza a quantidade de moedas de 1 euros:0
Introduza a quantidade de moedas de 50 centimos:1
Introduza a quantidade de moedas de 20 centimos:1
Introduza a quantidade de moedas de 10 centimos:0
Introduza a quantidade de moedas de 5 centimos: 1
*** Produto devolvido obrigado ***
*** Caixa - moedas***
2 euros : 2 | 1 euros : 5 | 0.5 euros : 10 | 0.2 euros : 12 | 0.1 euros : 15 | 0.5 euros :
20
*********************
```

Figura 2: Exemplo de apresentação da selecção de um produto

Caso o utilizador não introduza um valor suficiente, o sistema deverá devolver o troco inserido e voltar a apresentar a máquina. Após a compra bem sucedida o sistema deverá apresentar a quantidade de moedas em caixa. O sistema deverá também manter registo de todos os trocos que saíram da máquina. Além disso o sistema deverá atualizar quantidade de produtos no slot selecionado.

Se todos os produtos de um slot forem vendidos, então o sistema não deverá permitir compras nesse slot. Este slot deverá ser apresentado como 'vazio', ver figura abaixo:

```
...
Slot : A
Produto: Vazio
Preço : 0
Quantidade: 0   | Capacidade : 5
...
```

Figura 3: Exemplo de apresentação de um slot sem produtos

3.1.1.1 Trocos

O troco devolvido pela máquina deverá ser retirado da caixa. Caso não exista troco suficiente então a compra não deverá proceder. Nestes casos o dinheiro inserido pelo utilizador deverá ser removido, e o sistema deverá voltar a apresentar a máquina (Figura 1)

3.2 Funcionário

O funcionário representa o profissional responsável por gerir a máquina de vending. Este utilizador tem algumas tarefas únicas que deverão ser apresentadas num menu:

```
*** Bem vindo funcionário ***
1. Limpar slots
2. Limpar máquina
3. Adicionar produto
```

```
4. Alterar preço
5. Adicionar slots
6. Carregar moedas
7. Imprimir produtos
8. Gravar máquina
9. Carregar máquina
10. Remover trocos
0. Voltar
Introduza a sua opção:
```

Figura 4: Menu do funcionário

3.2.1 Limpar um slot

O sistema deverá permitir a um funcionário limpar o slot, removendo todos os elementos. O programa deverá pedir ao funcionário o slot a remover e remover todos os produtos presentes.

3.2.2 Limpar máquina

O sistema também deverá permitir ao funcionário remover todos os produtos da máquina. Esta opção limpa a máquina completamente, mantendo o limite de produtos por slot e o número de slots.

3.2.3 Adicionar produto

O programa deverá permitir a um funcionário repor produtos na máquina. Para isto o programa deverá solicitar o slot, o número de produtos a repor e o nome do produto. É importante notar que produtos no mesmo slot têm que partilhar o mesmo nome e preço. Caso existam vagas suficientes nesse slot os produtos são adicionados. Caso contrário, deverão ser considerados os seguintes pontos:

3.2.3.1 Se existir algum slot vazio, então os produtos não inseridos no slot solicitado inicialmente são colocados aí

3.2.3.2 O funcionário poderá alterar o tamanho do slot para acomodar todos os produtos inseridos

3.2.3.3 O funcionário poderá decidir adicionar apenas os produtos que conseguem ser inseridos tendo em conta o limite do slot

3.2.4 Alterar Preço

O programa deverá permitir a um funcionário alterar o preço de um produto. Para isso o funcionário deverá indicar o nome do produto e o seu novo preço. É importante ter em conta que o preço deverá ser múltiplo de 5 cêntimos, visto que esta á e menor moeda em caixa.

3.3 Adicionar Slots

O programa deverá permitir ao funcionário adicionar novos slots à máquina, para isso o funcionário deverá indicar a Letra e a capacidade do slot a adicionar.

3.4 Carregar moedas

O programa deverá permitir a um funcionário introduzir moedas na máquina, para isso o programa deverá pedir ao utilizador o número de moedas a introduzir de cada tipo. Estas moedas deverão passar imediatamente para a caixa da máquina. Esta operação deverá seguir a apresentação abaixo:

```
Escolheu adicionar moedas à Caixa
Introduza a quantidade de moedas de 2 euros:10
Introduza a quantidade de moedas de 1 euros:10
```

```
Introduza a quantidade de moedas de 50 cêntimos:1
Introduza a quantidade de moedas de 20 centimos:1
Introduza a quantidade de moedas de 10 cêntimos:0
Introduza a quantidade de moedas de 5 centimos: 1
```

Figura 5: Representação possível para a introdução de moedas por parte de um funcionário

3.5 Imprimir produtos

O sistema deverá permitir a um funcionário imprimir uma lista de todos os produtos presentes na máquina. Esta lista deverá ser impressa. 3.5.1. Por ordem alfabética, 3.5.2. Por preço e 3. Por quantidade. Estas opções deverão ser apresentadas num sub-menu. Por exemplo:

```
Escolheu imprimir produtos
    1.  Por ordem alfabética
    2.  Por preço
    3.  Por quantidade disponível
Escolha a sua opção:
```

Figura 6: Menu para a listagem de produtos na consola

3.6 Gravar máquina

O programa deverá permitir ao funcionário gravar o estado atual da máquina. Esta gravação deverá ser implementada utilizando um ou mais ficheiros. Na prática o programa deverá guardar a totalidade dos dados relevantes para máquina de vending, por exemplo, número de slots, tamanhos dos slots, produtos e caixa.

3.7 Carregar máquina

O programa deverá permitir a um funcionário o carregamento de um estado da máquina de vending previamente gravado (ver ponto 3.6). Esta funcionalidade deverá apagar o estado atual da máquina 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.

3.8 Remover fundos

O sistema deverá permitir a um funcionário remover moedas da máquina, para isto o funcionário deverá indicar o número de moedas de cada tipo que pretende remover. Antes desta operação o sistema deverá apresentar a quantidade de moedas em caixa.

4. Caixa

4.1 Falta de fundos

O sistema deverá imprimir uma mensagem de alerta quando o número de moedas em caixa (para qualquer tipo) seja inferior a 3. Por exemplo:

```
*** Atenção existem apenas 3 moedas de 1 euro ***
```

5. Relatório

Cada grupo, juntamente com o código deverá entregar um relatório em que apresenta a decisões de implementação necessárias para resolver cada um dos pontos mencionados neste enunciado.
Uma possível estrutura para este relatório é apresentada abaixo:

1. Capa

2. Índice
3. Introdução
4. Implementação (descrever ponto a ponto do enunciado)
5. Conclusão (o que foi feito, o que ficou a faltar implementar, reflexão sobre as decisões de implementação, foi correta? o que poderia ter sido implementado de outra forma?)

6. Entrega

A entrega do projeto será finalizada no moodle, nada data de 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 ficheiro necessários para a execução/avaliação. O nome do ficheiro entregue deverá ser o mesmo do número do grupo, por exemplo Grupo1.zip.
A entrega deste projeto está agendada para o dia 30/04/2020 às 23:59:59.
Na semana seguinte à entrega serão agendadas reuniões com os membros de cada grupo de forma a aferir a contribuição de cada um 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**).

6.1 Critérios de Avaliação

- Aplicação adequada do paradigma de programação imperativa;
- Definição de estruturas de dados adequadas;
- Cumprimento dos objetivos;
- Qualidade do código desenvolvido;
- Qualidade de execução do programa desenvolvido;
- Qualidade da interação com o utilizador (usabilidade);
- Relatório e documentação do código;
- Apresentações/discussões.

7. Notas implementação

Durante a inicialização do programa existem várias variáveis (por exemplo tamanho da máquina, tamanho dos slots ou número de moedas em caixa) 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.