Universidade Federal de Mato Grosso do Sul
Campus de Três Lagoas
Bacharelado em Sistemas de Informação
Algoritmos e Programação II

Trabalho Prático

1 Descrição

O minimercado CapiLagoa vem enfrentando uma série de desafios, incluindo a concorrência de grandes redes varejistas e a necessidade de se adaptar às novas tecnologias. O proprietário do minimercado, popularmente conhecido como "seu Canindé", ficou sabendo que um grupo de pessoas eram especialistas em resolver problemas de tecnologia em uma Universidade de sua cidade. Seu Canindé decidiu então procurar ajuda para implantar um sistema informtatizado para controloar seu negócio. Sua missão é ajudar seu Canindé criando um sistema que atenda as necessidades do CapiLagoa. Para isso, o sistema deve atender os requisitos descritos nas seções 1.1 e 1.2.

1.1 Dados dos Produtos

Seu Canindé considera importante que as seguintes informações sejam guardadas para cada produto:

- *Código* (Inteiro, maior que zero, código do produto. O código possui valor único);
- *Descrição* (Texto, descrição do produto. A descrição não possui espaços (" ") e muito menos acentos);
- *Quantidade* (Inteiro, maior ou igual a zero, quantidade armazenada em estoque);
- *Preço de Venda* (Real, maior ou igual a zero, representa o preço de venda).

1.2 Operações

O sistema de controle do minimercado deve funcionar como um interpretador de comandos e dar suporte para as seguintes operações[1]:

1. ***Inserção***: valida e cadastra um novo produto no sistema. Não existe um limitante quanto à quantidade de produtos a serem cadastrados. Sua forma geral é:
   `inserir` `<codigo> <nome> <quantidade> <valor>`
   Sucesso: `Produto <codigo> inserido com sucesso!`
   Falha: `Erro ao inserir o produto <codigo>.`

2. ***Exclusão***: exclui um produto através do seu código. Sua forma geral é:
   `excluir` `<codigo>`
   Sucesso: `Produto <codigo> excluído com sucesso!`
   Falha: `Produto <codigo> não cadastrado!`

[1] Nos exemplos, os comandos aparecem na cor azul e logo abaixo as respostas esperadas no caso de sucesso ou falha da sua execução.

3. ***Atualização***: através do código do produto, esta operação permite que o usuário altere a quantidade em estoque ou o valor de venda do produto. Esse comando recebe um parâmetro extra (p), que indica qual campo que será atualizado: "-q" para atualizar a quantidade em estoque ou "-v" para atualizar o valor de venda do produto. Sua forma geral é:
   atualizar < p > <codigo> <novo valor>
   Sucesso: `Produto <codigo> atualizado!`
   Falha: `Produto <codigo> não cadastrado.`

4. ***Consulta***: através de uma sequência de caracteres fornecida pelo usuário, esta operação apresenta todas as informações cadastradas dos produtos que possuem essa sequência de caracteres no seu nome[2].
   consultar <sequencia de caracteres>
   Sucesso:
   `<codigo> - <nome> - <qtd em estoque> - <valor>`
   ⋮
   Falha: `Nenhum produto encontrado!`

5. ***Relatório: Relação de Produtos***: apresenta na tela os dados de todos os produtos, organizados em ordem alfabética pelo nome do produto e agrupados pela letra inicial e salva essas informações no arquivo `Relatorio.txt`.
   relatorio
   Sucesso:
   ```
   <letra inicial>
       <codigo> - <nome> - <qtd em estoque> - <valor>
       <codigo> - <nome> - <qtd em estoque> - <valor>
   <letra inicial>
       <codigo> - <nome> - <qtd em estoque> - <valor>
   <letra inicial>
       <codigo> - <nome> - <qtd em estoque> - <valor>
       <codigo> - <nome> - <qtd em estoque> - <valor>
       <codigo> - <nome> - <qtd em estoque> - <valor>
   ⋮
   ```
   Falha: Nenhuma mensagem é mostrada.

6. ***Importação***: realiza a importação de vários produtos salvos em um arquivo texto com a seguinte formatação: a primeira linha contém o número de produtos no arquivo e em cada linha seguinte há a descrição de um produto com seus campos separados pelo caractere ';' (ponto-e-vírgula). A ordem dos dados do produto no arquivo é a mesma que na operação de inserção.
   importar <nome do arquivo>
   Sucesso: `<quantidade de produtos> produtos importados!`
   Falha: `Erro ao importar do arquivo <nome do arquivo>.`

[2] Dica: dê uma olhada na função `strstr` do arquivo `string.h`.

7. ***Registro de venda***: esta operação é talvez a que mais interessa a seu Canindé. Para esse comando são passados um conjunto de códigos de produtos (um por linha após a linha do comando) terminado pelo código 0 (zero).

   Para cada código informado, é necessário verificar se o produto pode ser vendido ou não. Um produto não pode ser vendido quando o código informado não está cadastrado ou quando a quantidade do produto em estoque é igual a zero. Nesse caso, uma mensagem indicando o motivo de não-venda do produto deve ser mostrada. Por outro lado, um produto só pode ser vendido se o código estiver cadastrado e a quantidade em estoque for maior que zero. No caso de sucesso de venda, deve-se decrementar a quantidade do produto em estoque e exibir uma mensagem que contenha o código, o nome e o valor de venda do produto.

   Ao final (código zero), deve ser mostrado na tela a soma total dos valores dos produtos vendidos com sucesso. Exemplo de execução do comando:

```
vender
<codigo>
<codigo> - <nome> - <valor>
<codigo>
<codigo> - produto em falta no estoque
<codigo>
<codigo> - <nome> - <valor>
<codigo>
<codigo> - produto não cadastrado
<codigo>
<codigo> - <nome> - <valor>
0
---------
Total <soma dos valores>
```

8. ***Help***: Como você já deve ter percebido, você terá que utilizar alocação dinâmica de memória para resolver esse problema[3]. Você pode optar pela estrutura que achar mais conveniente para armazenar os dados dos produtos (lista simplesmente encadeada, lista circular, lista duplamente encadeada, etc.). A essa altura da disciplina você deve ser capaz de olhar para essas estruturas e saber qual melhor se aplica para a solução desse problema em questão de eficiência de tempo das operações e de espaço utilizado.

   Esta operação simplesmente mostra na tela um breve texto justificando a estrutura escolhida por você e os motivos que o levaram a essa escolha.

```
help
Aqui queremos ler um texto que nos convença que você fez a
melhor escolha possível para seu programa!  :)
```

[3]Lembre-se de, sempre que tentar alocar um trecho de memória, verificar se a memória foi realmente alocada. Caso haja algum erro na alocação, você deve salvar o estado atual da estrutura, liberar a memória utilizada e terminar o programa.

9. ***Sair***: esta operação lista, em ordem não-decrescente pela quantidade em estoque, o código, o nome e a quantidade em estoque de todos os produtos que estão com o estoque baixo (menos de 15 unidades). Essa lista, além de ser mostrada na tela, também deve ser salva em um arquivo texto de nome `Comprar.txt`. Depois disso, os produtos cadastrados no sistema são salvos no arquivo `Produtos.dat`, mantendo o formato do arquivo, e finalizando o sistema.

   `sair`

   ```
   <codigo> - <nome> - <qtd em estoque>
   ⋮
   ```

2 Arquivo `Produtos.dat`

Os produtos cadastrados no sistema estão armazenados em um arquivo binário de nome `Produtos.dat`, ordenados pelo código do produto. Os produtos salvos nesse arquivo devem ser carregados automaticamente para seu sistema assim que ele for inicializado. O arquivo possui a seguinte formatação: o primeiro dado salvo é um inteiro n que indica a quantidade de produtos salvos no arquivo, considere que seu sistema não possui um limite quanto à quantidade de produtos a serem cadastrados. Logo em seguida, aparecem os n produtos serializados e salvos no arquivo.

3 Exemplo de execução do sistema

Seja o arquivo `Produtos_importacao.txt` com o seguinte conteúdo:

```
5
31;ARROZ;10;5.90
32;ACUCAR_MASCAVO;5;5.50
41;SABAO_EM_BARRA;12;4.00
42;SABAO_EM_PO;20;6.95
43;ESCOVA_DE_DENTE;40;6.95
```

Considere inicialmente que nenhum produto foi cadastrado no sistema. Um exemplo de execução do sistema é o seguinte[4]:

```
importar Produtos_importacao.txt
5 produtos importados!
inserir 43 DESODORANTE 15 13.49
Erro ao inserir o produto 43.
inserir 45 DESODORANTE 15 13.49
Produto 45 inserido com sucesso!
excluir 33
Produto 33 não cadastrado!
```

[4]Os comandos aparecem na cor azul e as respostas em preto.

```
excluir 45
Produto 45 excluído com sucesso!
atualizar -q 41 16
Produto 41 atualizado!
atualizar -v 30 15.50
Produto 30 não cadastrado.
consultar AR
31 - ARROZ - 10 - 5.90
32 - ACUCAR_MASCAVO - 5 - 5.50
41 - SABAO_EM_BARRA - 16 - 4.00
consultar DES
Nenhum produto encontrado!
importar Produtos_importacao.txt
0 produtos importados!
vender
43
43 - ESCOVA_DE_DENTE - 6.95
51
51 - produto não cadastrado
32
32 - ACUCAR_MASCAVO - 5.50
31
31 - ARROZ - 5.90
31
31 - ARROZ - 5.90
0
---------
Total 24.25
consultar AR
31 - ARROZ - 8 - 5.90
32 - ACUCAR_MASCAVO - 4 - 5.50
41 - SABAO_EM_BARRA - 16 - 4.00
sair
32 - ACUCAR_MASCAVO - 4
31 - ARROZ - 8
```

Ao final da execução os seguintes arquivos terão sido criados:

- `Produtos.dat` - arquivo binário com os produtos cadastrados. O arquivo possui a seguinte formatação: o primeiro dado salvo é um inteiro que indica a quantidade de produtos salvos no arquivo. Logo em seguida, aparecem os produtos serializados e salvos no arquivo.

- `Comprar.txt`

```
32 - ACUCAR_MASCAVO - 4
31 - ARROZ - 8
```

4 Entrega

Instruções para entrega do seu trabalho:

1. **Cabeçalho**

   Todos os arquivos do código fonte devem ter um cabeçalho com o seguinte formato:

```
/*****************************************
*                                        *
* Nome do(a) estudante                   *
* Trabalho Prático                       *
* Disciplina: Algortimos e Programação II *
* Professores: Ivone, Mauro e Ronaldo    *
*                                        *
*****************************************/
```

2. **O que entregar?**

   Você deve entregar um único arquivo com extensão `.rar` ou `.zip` contendo o(s) arquivo(s) com sua implementação. Dê o nome do seu usuário do laboratório do CPTL para seu arquivo e adicione a extensão `.rar` ou `.zip`. Por exemplo, `ivone.matsuno.rar` é um nome válido.

   Lembre-se: seu arquivo compactado deve conter APENAS os arquivos fontes `.c` da sua implementação. NÃO entregue qualquer outro arquivo, tais como arquivos `.o`, já compilados.

3. **Forma de Entrega**

   A entrega será realizada diretamente no Ambiente Virtual de Aprendizagem da UFMS (AVA) diretamente no *AVA*, na disciplina de Algoritmos e Programação II. Você pode editar a submissão da sua implementação quantas vezes quiser até às **23:59 horas** do dia **08/11/2023**. A última versão entregue é aquela que será corrigida. Encerrado o prazo, não serão mais aceitas implementações.

4. **Verificação dos dados de entrada**

   Não se preocupe com a verificação dos tipos de dados de entrada do seu programa. Seu programa não precisa fazer consistência dos tipos de dados de entrada. Isto

significa que se, por exemplo, o seu programa espera um número inteiro positivo e o usuário digitar uma letra, um cifrão, um número decimal, etc, o seu programa pode fazer qualquer coisa, como travar o computador ou encerrar a sua execução abruptamente com respostas erradas.

Você pode assumir também que todos os comandos passados para seu interpretador estão corretos, não havendo a necessidade de verificar a quantidade de parâmetros ou se o comando realmente existe.

ATENÇÃO: Considere que todas as palavras digitadas no seu interpretador de comando não são compostas e nem possuem acento algum.

5. **Arquivo com o programa fonte**

   Seu arquivo contendo o programa fonte na linguagem C deve estar bem organizado. Um programa na linguagem C tem de ser muito bem compreendido por uma pessoa. Verifique se seu programa tem a indentação adequada, se não tem linhas muito longas, se tem variáveis com nomes significativos, entre outros. Não esqueça que um programa bem descrito e bem organizado é a chave de seu sucesso.

6. **Conduta ética**

   O trabalho deve ser feito INDIVIDUALMENTE. Cada estudante tem responsabilidade sobre cópias de seu trabalho, mesmo que parciais. Não faça o trabalho em grupo e não compartilhe seu programa ou trechos de seu programa. Você pode consultar seus colegas para esclarecer dúvidas e discutir idéias sobre o trabalho, mas NÃO copie o programa!

   Trabalhos considerados plagiados terão nota ZERO.