UNIVERSIDADE FEDERAL RURAL DO SEMI-ÁRIDO
CURSO DE CIÊNCIA DA COMPUTAÇÃO
PROGRAMAÇÃO DE COMPUTADORES

TRABALHO PRÁTICO 3

Depois de vender muitas cópias do seu jogo de corrida de caminhões e fazer seu dinheiro render na bolsa de valores, Joãozinho decidiu usar parte dos seus ganhos para empreender. Ele quer entrar no ramo de Delivery de Fast Food e decidiu ele mesmo implementar o sistema de vendas e controle de estoque da sua empresa. Seu objetivo é vender pastéis, bolos, sanduiches, pizzas, sucos etc.

O sistema de vendas funcionará da seguinte forma:

- Os clientes farão seus pedidos através do aplicativo RapiZinho. O aplicativo vai mostrar um menu com os produtos disponíveis, coletar o pedido, atualizar o estoque e gravar o pedido em um arquivo texto, como no exemplo abaixo:

```
 RapiZinho
===========
(A) Pastel
(B) Bolo
(C) Pizza
(D) Suco
(S) Sair
===========
Opção: [a]
```

```
Pedido
======
Pastel
R$5.50
======
Quantidade: [2]
```

```
 RapiZinho
===========
2 x Pastel de R$5.50 = R$11.00
===========
(A) Pastel
(B) Bolo
(C) Pizza
(D) Suco
(S) Sair
===========
Opção: [d]
```

```
Pedido
======
Suco
R$4.00
======
Quantidade: [2]
```

Se a quantidade solicitada não estiver disponível em estoque, o programa deve apresentar uma mensagem informativa e retornar ao menu para que o cliente possa continuar fazendo seu pedido, sem alterar os produtos já inseridos na cesta.

```
 RapiZinho
===========
2 x Pastel de R$5.50 = R$11.00
2 x Suco de R$4.00 = R$8.00
===========
(A) Pastel
(B) Bolo
(C) Pizza
(D) Suco
(S) Sair
===========
Opção: [S]
```

```
 RapiZinho
===========
2 x Pastel de R$5.50 = R$11.00
2 x Suco de R$4.00 = R$8.00
Taxa de entrega = R$6.00
===========
Total = R$25.00

[P] Pix
[C] Cartão

Pagamento: [p]
```

```
 RapiZinho
===========
2 x Pastel de R$5.50 = R$11.00
2 x Suco de R$4.00 = R$8.00
Taxa de entrega = R$6.00
Desconto de 10% = R$2.50
===========
Total = R$22.50

Confirma Pedido (S/N): [s]
```

Pedidos no Pix tem desconto de 10%, pedidos no cartão tem desconto de 5%.

Após a realização, ou cancelamento do pedido, o programa deve retornar para a tela inicial com a lista de produtos e nenhum produto na cesta. Se o cliente **sair do programa com a cesta vazia**, nenhum arquivo deve ser gravado e o estoque deve permanecer inalterado.

Após a confirmação de um pedido, o programa deve gravar um recibo de compra. O recibo é um arquivo texto, sendo o nome do arquivo dado pela palavra "pedido_" concatenada com o número do pedido e com a extensão ".txt". Por exemplo, o arquivo abaixo se chamaria "pedido_1.txt":

```
Pedido #1
------------------------------------------------
2 x Pastel de R$5.50 = R$11.00
2 x Suco de R$4.00 = R$8.00
Taxa de entrega = R$6.00
Desconto de 10% = R$2.50
------------------------------------------------
Total = R$22.50
```

Uma vez gerado o recibo de compra, a quantidade em estoque de cada produto deve ser atualizada no sistema. O estoque deve ser guardado em um **vetor dinâmico** de produtos. Ao sair do programa, o conteúdo do vetor deve ser gravado em um **arquivo binário** de estoque, sobrescrevendo o arquivo antigo, se ele existir.

- Um produto deve ser representado por um registro contendo:
    - Nome do produto: 24 caracteres (incluindo o \0)
    - Preço: ponto-flutuante com precisão simples
    - Quantidade em estoque: inteiro de 32 bits sem sinal

  O aplicativo RapiZinho deve **abrir e ler o arquivo de estoque** mais recente **sempre que iniciar**. Os produtos devem ser lidos para um vetor dinâmico e o arquivo deve ser fechado. O arquivo só deve ser manipulado no início e no fim do programa. Todas as demais atualizações de estoque devem ser realizadas no vetor.

O programa, quando iniciado com a opção -c na linha de comando, deve entrar em modo de controle de estoque. Neste modo, o seguinte menu de opções deve ser apresentado:

```
Painel de Controle
==================
(A)dicionar
(E)xcluir
(L)istar
(S)air
==================
Opção: [_]
```

Ao entrar, o programa deve abrir o arquivo de estoque e ler os elementos para um vetor dinâmico com a quantidade exata de produtos gravados no arquivo. O arquivo deve ser fechado logo em seguida. Todas as modificações no estoque devem ser realizadas em cima do vetor.

Se o **arquivo de estoque ainda não existir**, inicie com um vetor vazio (nullptr). Não é necessário criar o arquivo, ele será criado ao finalizar o programa. Também não é necessário alocar o vetor dinâmico, ele será alocado no momento de adicionar o primeiro produto.

Cada opção deve permitir que o usuário entre com os dados necessários para completar aquela tarefa. As opções são acionadas pelo pressionamento da primeira letra (maiúscula ou minúscula) de cada opção. Abaixo está uma descrição para cada uma delas:

- **Adicionar**: deve adicionar um novo registro ao vetor de estoque. Este registro deve conter o nome do produto, o preço e a quantidade do produto que está chegando ao estoque. Caso o produto já exista, deve-se apenas atualizar o preço e acrescentar a quantidade informada ao valor atual do estoque.

  ```
  Adicionar
  ---------
  Produto: Pastel
  Preço: 5.50
  Quantidade: 10
  ```

- **Excluir**: deve mostrar uma lista numerada (a partir de 1) dos produtos existentes no estoque. Quando um número for escolhido, o programa deve, após confirmar a exclusão, remover o registro do produto do vetor de estoque.

  ```
  Excluir
  -------
  1) Pastel
  2) Bolo
  3) Suco

  Produto: [2]
  Deseja excluir "Bolo" (S/N)? s
  ```

- **Listar**: deve mostrar a lista de produtos existentes no vetor de estoque, com seu nome, preço e quantidade.

  ```
  Listagem
  --------
  Pastel - R$5.50 - 10 und.
  Bolo   - R$8.00 - 20 und.
  Suco   - R$4.00 -  8 und.
  ```

- **Sair**: deve sobrescrever o arquivo binário de estoque com os valores atuais do vetor de estoque.

Observe que o vetor de estoque inicia vazio ou exatamente com a quantidade de elementos do arquivo. Caso um novo produto seja adicionado, não haverá espaço para ele no vetor. Sempre que isso acontecer será necessário **expandir o vetor**. Construa uma função para expandir o vetor, aumentando o seu tamanho em uma unidade cada vez que a função for chamada.

EXIGÊNCIAS

O programa deve ser construído apenas com os assuntos vistos durante o curso, entretanto os seguintes assuntos que foram abordados na disciplina não devem ser usados no trabalho:

- Não utilize variáveis globais
- Não utilize a biblioteca <string>
- Não utilize a biblioteca <vector> ou <array>

ENTREGA DO TRABALHO

Grupos: Trabalho individual

Data da entrega: 17/04/2023 (até a meia noite)

Valor do Trabalho: 3,0 pontos (na 3a Unidade)

Forma de entrega: enviar apenas os arquivos fonte (.cpp) e os arquivos de inclusão (.h) compactados no formato **zip** através da tarefa correspondente no SIGAA.

O não cumprimento das orientações resultará em **penalidades:**

- Programa não executa no Visual Studio 2022 (3,0 pontos)
- Programa contém partes de outros trabalhos (3,0 pontos)
- Programa utiliza recursos não vistos na disciplina (3,0 pontos)
- Atraso na entrega (1,5 pontos por dia de atraso)
- Arquivo compactado em outro formato que não zip (0,5 ponto)
- Envio de outros arquivos que não sejam os .cpp e .h (0,5 ponto)
- Programa sem comentários e/ou desorganizado (0,5 ponto)