MINISTÉRIO DA EDUCAÇÃO

UNIVERSIDADE FEDERAL RURAL DO SEMIÁRIDO

DEPARTAMENTO DE CIÊNCIAS EXATAS, TECNOLÓGICAS E HUMANAS

CURSO DE TECNOLOGIA DA INFORMAÇÃO

MONITORIA DE ALGORITMOS – PEX-1236

PROPOSTAS DE PROJETOS A SEREM DESENVOLVIDOS NA MONITORIA DE ALGORITMOS COM OS DISCENTES DA DISCIPLINA

ORIENTADOR: PROF. DR. REUDISMAM ROLIM DE SOUSA

MONITOR: TÁSSIO FERNANDES COSTA

PAU DOS FERROS – RN

2019

Sumário

1. Apresentação

O objetivo com este documento é apresentar duas propostas de projetos a serem implementados durante as monitorias de Algoritmos com os alunos do referido componente curricular.

A primeira proposta é desenvolver um simples jogo da velha utilizando a linguagem C. Espera-se desenvolver este jogo durante as duas monitorias que ocorrerão nos dias 09 e 12 de julho de 2019.

A segunda proposta é desenvolver um sistema simples de controle de refeição do restaurante universitário da UFERSA. Espera-se desenvolver este sistema durante as duas monitorias que ocorrerão nos dias 16 e 18 de julho de 2019.

Maiores detalhes de cada proposta são apresentados, respectivamente, nos tópicos 1 e 2.

2. Jogo da Velha

No jogo da velha, dois jogadores tomam turnos alternados marcando uma casa de um tabuleiro, inicialmente sem marca alguma, de 3 x 3 casas, alternadamente com um 'X' ou um 'O'. Cada jogador só pode jogar em alguma casa que esteja vazia. Vence o jogador que conseguir marcar as 3 casas de uma mesma linha, coluna ou diagonal. Caso não haja mais casas vazias e nenhum jogador tenha vencido, a partida termina em empate.

Sendo a interface em modo texto, cada jogador deverá, na sua vez de jogar, digitar um número correspondente à casa em que quer jogar. O número correspondente a cada casa do tabuleiro é de acordo o seguinte padrão:

```
1|2|3
-+-+-
4|5|6
-+-+-
7|8|9
```

Se o jogador jogar em uma casa já ocupada, o pedido de jogada deve ser refeito. Caso o jogador jogue numa casa vazia, a casa será marcada com a marca correspondente ao jogador e o jogo prossegue.

A implementação deve ser em modo texto usando as entradas e saídas padrão (*stdin* e *stdout*) da linguagem C. O programa deve mostrar o estado do tabuleiro e pedir o lance do jogador. O processo deve ser repetido enquanto a partida não tiver terminado. Um exemplo de uma sessão de uma partida com alguns lances é como se segue:

```
 | |
-+-+-
 | |
-+-+-
 | |
Entre com a casa a ser jogada: 1
O| |
-+-+-
 | |
-+-+-
 | |
Entre com a casa a ser jogada: 2
O|X|
-+-+-
 | |
-+-+-
 | |
Entre com a casa a ser jogada: 4
O|X|
-+-+-
O| |
-+-+-
 | |
```

Neste exemplo, os números 1, 2 e 4 correspondem aos lances feitos pelos jogadores.

Para a implementação do jogo da velha é sugerido implementar as seguintes funções:

- `void inicializaTabuleiro(char tabuleiro[3][3])`: função responsável por inicializar o tabuleiro com algum valor para possibilitar a manipulação dos valores do tabuleiro.
- `void mostraTabuleiro(char tabuleiro[3][3])`: função responsável por mostrar o estado do tabuleiro.
- `int preencheCasaDoTabuleiro(char tabuleiro[3][3], int casa, int jogador)`: função responsável por preencher o tabuleiro dado a casa

selecionada e o jogador. **Atenção**: tratar a situação em que a casa selecionada já está preenchida.

- `void jogar(char tabuleiro[3][3], int jogador)`: função responsável por efetuar uma jogada de um jogador. **Atenção**: permitir que a jogada seja realizada apenas se a casa selecionada é válida e, sendo válida, que a casa esteja vazia.
- `int statusDoJogo(char tabuleiro[3][3])`: função responsável por verificar se existe algum ganhador, se o jogo ficou empatado ou se o jogo deve continuar. Neste último caso, deve ser informado quem é o próximo a jogar.
- `void resultadoFinal(int situacao)`: função responsável por apresentar o resultado final do jogo. Os possíveis resultados finais devem ser:
  - `“Partida empatada!”`
  - `“Jogador 1 venceu!”`
  - `“Jogador 2 venceu!”`
  - `“Isto não deveria acontecer!”`

3. Sistema de Controle de Refeição do RU da UFERSA

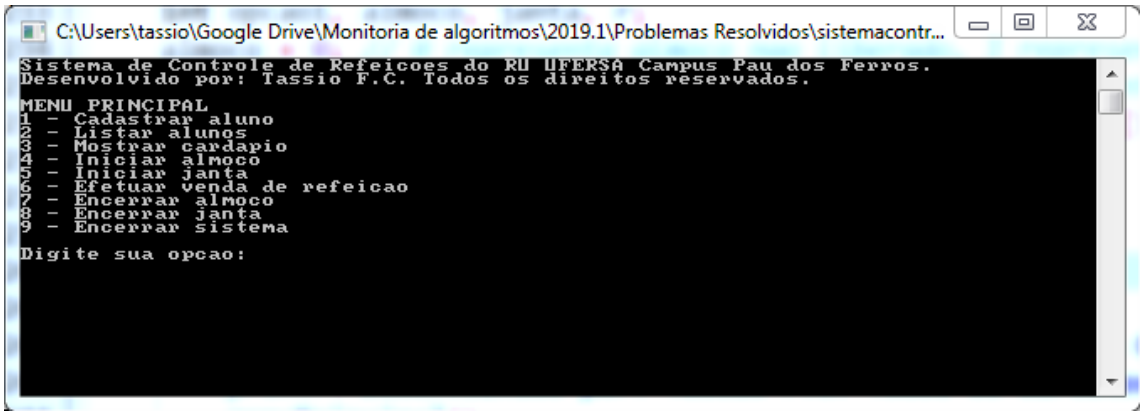
O sistema de controle de refeição do RU da UFERSA é um sistema simples que é responsável por efetuar a venda de uma refeição (almoço ou janta) a um aluno (professores, técnicos administrativos e público externo não são considerados) que se encontra cadastrado no sistema. O sistema deve provê as seguintes funcionalidades:

- **Cadastrar alunos**: para que uma venda seja realizada é necessário que exista algum aluno cadastrado. Neste sentido, esta funcionalidade é para permitir que todos os alunos que requeiram o serviço de compra de refeição no RU sejam cadastrados no sistema, pois, a venda deve ser permitida somente para quem tiver cadastro feito.
- **Listar alunos**: funcionalidade auxiliar que permite ao usuário do sistema visualizar os dados de todos os alunos cadastrados.
- **Mostrar cardápio**: funcionalidade responsável por exibir os cardápios do almoço e da janta. Para simplificar, vamos considerar que existe um único cardápio para o almoço e um único cardápio para a janta.

- **Iniciar almoço**: funcionalidade responsável por permitir que o almoço seja liberado para venda. Devemos garantir que, enquanto o almoço estiver liberado, a janta não seja liberada e vice-versa.
- **Iniciar janta**: funcionalidade responsável por permitir que a janta seja liberada para venda.
- **Efetuar venda de refeição**: funcionalidade responsável por efetuar a venda de uma refeição ao aluno. Para isso, deve ser informado a matrícula do aluno. A venda só pode ser efetuada caso aquele aluno não tenha feito nenhuma compra para a refeição que se encontra liberada.
- **Encerrar almoço**: funcionalidade responsável por encerrar a venda de almoço.
- **Encerrar janta**: funcionalidade responsável por encerrar a venda de janta.
- **Encerrar sistema**: funcionalidade responsável por encerrar o sistema.

Portanto, o menu de interação do sistema deve ser idêntico ao apresentado na Figura 1.

Figura 1 - Menu de Interação do Sistema com o Usuário



Fonte: próprio autor.

Para a construção do referido sistema é sugerido definir três matrizes de *strings* para armazenar os dados dos alunos. As três matrizes sugeridas são para armazenar os nomes, as matrículas e os status de refeição de todos os alunos. Por questão de simplificação, recomenda-se considerar que na UFERSA Campus Pau dos Ferros existe no máximo 100 alunos. Além disso, é sugerido implementar as seguintes funções utilizando a linguagem de programação C:

- `void inicializarAlunos(char nomes[100][100], char matriculas[100][11], char statusRefeicao[100][13])`: função responsável por inicializar os dados de todos os alunos com valores padrões para auxiliar na manipulação dos mesmos. A inicialização pode ser, por exemplo, um simples espaço em branco na primeira posição de cada *string*. É sugerido considerar como dados dos alunos os seguintes:
    - **Nome**
    - **Matricula**
    - **Status de refeição** (disponível, quando o aluno não realizou nenhuma compra para a refeição que está liberada; ou indisponível, quando o aluno já realizou uma compra para a refeição que está liberada). Após o encerramento de uma refeição, os alunos indisponíveis devem voltar a ficar novamente disponíveis.
- `void tornarRefeicaoDisponivel(char statusRefeicao[100][13], int posicao)`: dado a matriz que representa o status de todos os alunos registrados e dado a posição de um aluno nesta matriz, esta função é responsável por tornar o referido aluno disponível para efetuar compra de uma refeição.
- `void tornarTodosDisponivel(char statusRefeicao[100][13])`: dado a matriz que armazena o status de todos os alunos registrados, esta função é responsável por tornar todos os alunos disponíveis para efetuar compra de uma refeição.
- `void tornarRefeicaoIndisponivel(char statusRefeicao[100][13], int posicao)`: dado a matriz que representa o status de todos os alunos registrados e dado a posição de um aluno nesta matriz, esta função é responsável por tornar o referido aluno indisponível para efetuar compra de uma refeição.
- `int nomeEhValido(char nome[100])`: função responsável por validar um nome. Um nome é válido quando possui apenas letras minúsculas ou maiúsculas ou espaço em branco.
- `int matriculaEhValida(char matricula[11])`: função responsável por validar uma matrícula. Uma matrícula é válida quando possui exatamente 10 dígitos.

- `int matriculaExiste(char matriculas[100][11], char matricula[11])`: função responsável por verificar se uma dada matrícula já se encontra cadastrada.
- `int cadastrarAluno(char nomes[100][100], char matriculas[100][11], char statusRefeicao[100][13])`: função responsável por cadastrar um aluno. Para isso, precisa receber as três matrizes de *strings* que armazenam os dados dos alunos. No cadastro devemos nos atentar a algumas validações:
    - Primeiro, precisamos solicitar a matrícula do aluno e verificar se ela é válida (usar a função que `matriculaEhValida`). Enquanto não for devemos ficar solicitando uma matrícula válida.
    - Após isto, precisamos verificar se a matrícula informada já se encontra registrada (usar a função `matriculaExiste`). Se sim, não deve ser feito um novo cadastro. Se não, o cadastro deve prosseguir solicitando ao usuário o nome do aluno a ser cadastrado. Este nome também precisa ser validado (usar a função `nomeEhValido`).
- `void listarAlunos(char nomes[100][100], char matriculas[100][11], char statusRefeicao[100][13])`: função responsável por mostrar os dados de todos os alunos cadastrados.
- `void cardapioAlmoco()`: função responsável por mostrar o cardápio do almoço.
- `void cardapioJanta()`: função responsável por mostrar o cardápio da janta.
- `void menuPrincipal()`: função responsável por mostrar o menu principal do sistema.
- `int refeicaoDisponivel(char matricula[11], char matriculas[100][11], char statusRefeicao[100][13])`: função responsável por verificar se um aluno está disponível para comprar uma refeição. Se sim, retorna a posição do status deste aluno na matriz `statusRefeicao`. Se não, retorna -1.
- `void efetuarVenda(char matriculas[100][11], char statusRefeicao[100][13])`: função responsável por efetuar uma venda de uma refeição para um aluno. Para isso, deve ser informado a matrícula

do aluno. Atentar-se ao que foi mencionado anteriormente sobre esta funcionalidade.

- `int le_opcao()`: função responsável por lê um opção do usuário dentre as disponibilizadas no menu. Observe que uma opção válida é aquela que está entre 1 e 9.
- `void limpa_tela(void)`: função responsável por limpar a tela facilitando a interação do usuário com o sistema.