Aula prática - Semana 13 - ERE

Nesta lista, a questão 1 constitui a atividade avaliativa que será utilizada para contabilizar notas de atividades práticas.
Os seguintes pontos serão considerados na avaliação:

- Atender os requisitos do enunciado.
- Indentação e organização de código.
- Utilização dos tipos corretos de variáveis.
- Utilização correta de constantes nomeadas quando necessário.
- Observação das boas práticas de programação vistas em aula.
- Realização de validação quando a questão requisitar.

ATENÇÃO:

- As atividades desta semana são focadas em estratégias que podem ser usadas para implementar funcionalidades do trabalho final. Notem que não é esperado que vocês apliquem as ideias exatamente da forma que são necessárias para realizar as tarefas práticas. Em alguns casos, dependendo da estrutura geral do programa de vocês, será necessário realizar adaptações. No entanto, as estratégias trabalhadas nesta prática podem servir de inspiração.
- Utilizem como arquivos de entrada para a realização dos exercícios o arquivo de mapa que foi disponibilizado no moodle.

1. Você deve fazer um programa para ler informações do mapa do jogo do trabalho final de vocês:
   - Defina uma estrutura para representar a posição de elementos dinâmicos do jogo (como jogador, ninjas, etc). Ela deve armazenar a linha e a coluna em que o elemento se encontra na matriz que representa o mapa. Desta forma a estrutura pode ter campos chamados linha e coluna, ou x e y, etc.

   - Defina uma função que recebe como parâmetros: uma string que representa o nome do arquivo do mapa, o ponteiro para uma estrutura POSICAO que representa uma referência para a posição do jogador, um vetor de estruturas POSICAO que representa as posições dos ninjas no mapa, um ponteiro para inteiro que retorna o número de ninjas que o mapa possui e uma matriz de caracteres que representa todas as informações do mapa, tais como estão no arquivo. Considere incluir outros parâmetros se considerar necessário. A função deve abrir o arquivo de mapa, efetuar a leitura das informações, retornar por referência a posição do jogador, preencher o vetor de posições de ninjas no mapa, retornar por referência a quantidade de ninjas que o mapa possui e preencher a matriz com os dados do mapa. Considere que a posição nesse caso é a linha e a coluna da matriz. A função também retorna (via return) 1 caso a operação de leitura tenha sido um sucesso e 0 caso tenha ocorrido uma falha. **ATENÇÃO**: Lembre-se de que no fim de cada linha do mapa há um caractere que indica

quebra de linha (‘\n’). Isso pode dificultar a aplicação de estratégias mais diretas para ler a matriz.

- Defina uma função que recebe como parâmetros: uma string que representa o nome de um arquivo, uma estrutura POSICAO que representa a posição do jogador, um vetor de estruturas POSICAO que representa as posições dos ninjas e um inteiro que representa a quantidade de ninjas. Considere usar outros parâmetros se considerar necessário. A função deve salvar as informações de posição (linha e coluna) do jogador e dos ninjas em linhas separadas no arquivo cujo nome foi passado como parâmetro, gerando um arquivo com a seguinte estrutura (os valores são meramente exemplos ilustrativos):

```
Posição do jogador: 10,22
Posição do ninja 1: 2,32
Posição do ninja 2: 2,53
Posição do ninja 3: 11,44
Posição do ninja 4: 14,32
Posição do ninja 5: 20,8
Posição do ninja 6: 20,36
```

- O programa principal deve requisitar o nome do arquivo do mapa que deve ser lido e o nome do arquivo em que serão armazenadas as informações de posições. A seguir, o programa deve efetuar a leitura das informações (posição do jogador, posições dos ninjas e a matriz de caracteres) do mapa, exibir na tela as posições do jogador e dos ninjas, bem como a matriz de caracteres que representa o mapa. Por fim, o programa deve salvar as informações de posição no arquivo informado pelo usuário. O programa deve emitir avisos informando se operações de leitura ou escrita em arquivos falhou.

- **Atenção**: Considere essa atividade no contexto do trabalho final. Lembre-se que o enunciado do trabalho define um número máximo de ninjas que um mapa pode ter.

2. (Desafio) Ideia para utilizarem no trabalho final. Crie uma função que conta quantos arquivos de mapa existem no diretório do programa. Esta função pode receber como parâmetro o nome base do arquivo (“mapa” no nosso caso) e retornar um inteiro que representa o número de mapas disponíveis. Uma ideia geral para a função é utilizar um contador que começa em 1 e vai incrementando. Para cada valor do contador, cria-se um novo nome de arquivo concatenando o nome do arquivo base com o contador (atenção para o caso em que ele só tem um dígito) e a extensão do arquivo. Uma vez elaborado o nome do possível mapa, tenta-se abri-lo. Se o diretório só tiver N mapas, isso significa que na tentativa de abrir o mapa N+1 a operação resultará em erro e, com isso, o programa retorna N como o número de mapas disponíveis.

3. (Desafio) Implemente a sua própria versão simplificada da função atoi. A função deve receber uma string cujo conteúdo representa um número inteiro, e deve retornar o número inteiro correspondente. Assuma que a string passada como parâmetro sempre terá um número inteiro maior ou igual a zero bem formado (ou seja, não conterá outros caracteres não numéricos). Essa suposição simplifica a função, já que a atoi original é implementada para ser robusta e ser capaz de lidar com strings com números mal formados. A partir da versão simplificada, se quiser, tente aperfeiçoar a função para reproduzir exatamente o comportamento da função atoi original (capaz de lidar com strings que não possuem apenas caracteres numéricos).