Universidade de Brasília
Departamento de Ciência da Computação/IE
Semestre: 2/2017
Disciplina: Algoritmos e Programação de Computadores

**Trabalho Prático – Parte 2**

O objetivo da parte 1 deste trabalho foi aprender a usar **estruturas de dados do tipo matriz, subalgoritmos (funções), macros do pré-processador, bibliotecas diferentes das utilizadas até então, variáveis globais de forma correta.**

Agora, na parte 2 do trabalho, o aluno deverá aplicar os conhecimento sobre **registros e arquivos.**

**Especificação da segunda etapa:**

Nesta segunda etapa o aluno deverá implementar as opções 2 e 3 do Menu Principal da Parte 1 do trabalho prático. Ou seja: "2. CONFIGURAÇÕES" e "3. RANKING".

A seguir é apresentado o detalhamento para cada opção.

**Opção 2. CONFIGURAÇÕES:**

As configurações do jogo devem ser salvas em um arquivo texto, e carregadas toda vez que o jogo for iniciado. Ou seja, toda vez que a opção JOGAR for selecionada o programa deverá consultar a configuração atual que está gravada no arquivo texto. Caso o jogador faça alguma alteração na configuração atual, então a nova configuração é que passará a valer para a próxima jogada.

O menu de configurações deverá ter as seguintes opções:

1. Tabuleiro
2. NPCs
3. Ativar Modo Rankeado
4. Voltar

Nas opções 1 e 2 deverão ser alteradas as seguintes variáveis:

- **altura, largura –** dimensões do tabuleiro.
- **quantO, quantB, quantX –** quantidade de cada tipo de inimigo que o jogo conterá em uma partida.
- **tamanhoB, loucuraX, tamanhoQ –** variáveis que representam características de cada inimigo:
  - **tamanhoB –** armazena o maior tamanho do rastro que o inimigo 'B' pode gerar.
  - **loucuraX –** representa o intervalo de tempo, em turnos, em que o 'X' se movimenta aleatoriamente.
  - **tamanhoQ –** representa a área de explosão da bomba 'Q'.

Para cada variável alterada deve ser definido e informado na tela os valores limites. Além disso, o valor digitado deve ser validado e deve ser mostrada uma mensagem de erro caso o valor fique fora dos limites estabelecidos; neste caso a leitura deve prosseguir até que um valor válido seja digitado.

Na opção 4 o usuário poderá ativar o modo ranqueado, que não utilizará as configurações gravadas no arquivo tipo texto. Ao invés disto, este modo usará sempre os valores padrões abaixo:

- altura e largura do tabuleiro: 30
- quantidade de B: 3
- quantidade de X: 7
- tamanho do B: 7
- tamanho do Q: 10
- loucura: utilizar o mesmo valor utilizado na Parte 1 do seu trabalho.

Lembrando que a pontuação será salva no ranking somente quando o modo ranqueado estiver ATIVO.

**Opção 3. RANKING:**

Cada vez que o jogador iniciar um jogo com o modo ranqueado ATIVADO deve ser exigido que ele digite seu nome (ou *nickname*), que será utilizado para identificá-lo no ranking. O nome máximo do jogador deve ser de 10 caracteres e no mínimo de 1 caractere. Isso deve ser testado e informado mensagem de erro caso o valor esteja fora dos limites. Enquanto a leitura não estiver correta deve ser lido novamente.

Após o final de cada partida devem ser gravados no arquivo binário do ranking os dados do jogador se e somente se ele tiver jogado no modo ranqueado. Abaixo são fornecidas as instruções sobre o registro de uma partida no ranking.

Quando o usuário consultar a opção "2. RANKING", o arquivo binário contendo os dados do ranking deve ser lido e os 10 jogadores com melhor pontuação devem ter seus nomes e pontuação mostrados na tela, em ordem DECRESCENTE de pontuação. Após mostrar o Ranking, coloque na tela uma mensagem "Tecle <ENTER> para continuar!" e use um getchar(); no programa para segurar a tela e assim que usuário visualizar o ranking.

**Registro de uma partida no ranking:**

O aluno deverá implementar a função sugerida `registerMatch()` para registrar a pontuação obtida durante a partida em um arquivo binário com as informações de ranking. O jogo deverá guardar apenas o ranking dos 10 melhores resultados em ordem DESCRESCENTE.

Para implementar esta função, será necessário definir o seguinte tipo:

```
typedef struct {
     char nick[11];
     int score;
} Player;
```

O nome do arquivo binário de ranking deve ser: ranking.bin

Utilizando o tipo definido, crie a função `registerMatch()` com a seguinte lógica: tentar abrir o arquivo para leitura (opção "rb"); caso não seja possível, abrir o arquivo para escrita (opção "wb") e gravar um vetor de 10 posições do tipo `Player`, em que o primeiro elemento descreve o

resultado da partida que acabou de terminar e os próximos 9 elementos devem possuir a string `nick` vazia, ou seja, com '\0' na primeira posição, e encerrar a função; caso seja possível, utilizar o arquivo aberto para ler um vetor de 10 posições do tipo `Player`; após ler o vetor, a função deve procurar uma posição para inserir um registro com o resultado da partida que acabou de terminar; se foi possível inserir o registro, abrir o arquivo para escrita e escrever o novo vetor de 10 posições. Lembrando para tomar cuidado para não sobrescrever em cima do registro anterior, ou seja, o novo registro de ranking deve ser INSERIDO na posição adequada, e caso o vetor já esteja com 10 elementos, o último (ou de menor pontuação) será descartado da lista do ranking.

Observe que se a pontuação obtida for menor do que a última pontuação do ranking, esta NÃO deverá ser gravada no ranking atual.

**IMPORTANTE: deverá ser enviado também um arquivo .txt contendo as informações sobre alterações os extras que forem implementados.**

**Observações Gerais:**

1. Incluir cabeçalho como comentário (ou seja, entre /* */), no programa fonte, de acordo com os **critérios de avaliação dos trabalhos (Disponível no Moodle).**
2. A data de entrega do programa é: **09/12/2017 (sábado) até às 23:55 hs**.