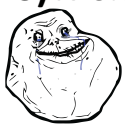
Universidade Federal do Ceará
Centro de Ciências
Departamento de Computação

Trabalho Final – Jogo Batalha Naval
Disciplina Programação (CK0110) – Semestre 2015.2

Prof. Miguel Franklin

***** PARA EQUIPES DE NO MÍNIMO 2 (DOIS) e NO MÁXIMO 3 (TRÊS) ALUNOS *****

(A não ser que você seja o último que não conseguiu equipe.) 

Desenvolver um programa em linguagem C representando o Jogo Batalha Naval, onde um usuário deverá ser capaz de jogar contra o computador.

Requisitos Funcionais:

Obs: Como o próprio nome diz, um "Requisito Funcional" é um requisito: é obrigatório. Portanto, a palavra DEVE tem que ser interpretada como em: "Mininu, você DEVE comer todo o seu brócolis!", e não como "Deve ser o repolho que eu comi ontem."

1. O tabuleiro DEVE ter dimensão 10x10.
2. Cada jogador DEVE dispor do seguinte conjunto de embarcações:
    a. 01 Porta-aviões (ocupa 5 casas);
    b. 02 *Destroyers* (cada um ocupa 3 casas);
    c. 03 Fragatas (cada um ocupa 2 casas);
    d. 02 Submarinos (cada um ocupa 1 casa);
    e. 01 Jangada (ocupa 1 casa).
3. O computador, no início de cada jogo, DEVE distribuir aleatoriamente todas as embarcações de cada jogador (humano e computador) no tabuleiro na horizontal e na vertical.
4. Cada coluna do tabuleiro DEVE ser representada por uma letra ("A" a "J"), enquanto cada linha do tabuleiro DEVE ser representada por um número (1 a 10).
5. O jogo DEVE mostrar, lado a lado, a cada interação, em modo texto, o tabuleiro do humano mostrando o posicionamento de suas embarcações, e o tabuleiro do computador, escondendo as embarcações deste, e mostrando apenas os conteúdos dos espaços que já foram atacados.
    a. Opcionalmente, o tabuleiro PODE ser mostrado também em modo gráfico.
6. Um espaço do tabuleiro do computador ou do humano que já tenha sido atacado e acertou a água DEVE ser mostrado como "O".
7. Um espaço do tabuleiro do computador que já tenha sido atacado e acertou uma embarcação ou um espaço do tabuleiro do humano deve mostrar:
    a. Para uma embarcação disposta na **vertical** que ocupe duas ou mais casas: "^" (superior) ou

"v" (inferior) se uma extremidade da embarcação tiver sido atingida. No caso de ocupar mais de duas casas, o caractere "#" DEVE ser usado quando o corpo da embarcação for atingido.

b. Para uma embarcação disposta na **horizontal** que ocupe duas ou mais casas: "<" (esquerda) ou ">" (direita) se uma extremidade da embarcação tiver sido atingida. No caso de ocupar mais de duas casas, o caractere "#" DEVE ser usado quando o corpo da embarcação for atingido.

c. Um submarino DEVE ser identificado pelo caractere "@".

d. Uma jangada DEVE ser identificada pelo caractere "&".

e. Após afundada uma embarcação, seja do humano ou do computador, os espaços que eram ocupados por esta embarcação devem ser marcados pelo caractere "*".

```
     Humano       Computador

   ABCDEFGHIJ     ABCDEFGHIJ
  +----------+   +----------+
 1| O <###>  +  1|          |
 2|          +  2| *    O   |
 3|^  O  ^   +  3|          |
 4|v     #   +  4|          |
 5|   @  v   +  5|       ^  |
 6| <#>      +  6|       #  |
 7|       <> +  7|  O       |
 8|  O       +  8|          |
 9|&      ** +  9|          |
10|     @    + 10|          |
  +----------+   +----------+

Última jogada do computador: C7
Última jogada do humano: D3

Entre a sua jogada:
```

Figura 1: Exemplo de representação do tabuleiro.

8. Após distribuir aleatoriamente as embarcações de cada jogador, o jogo DEVE pedir, do humano, uma jogada.
9. Uma jogada DEVE ser recebida do humano da forma mais flexível possível, como nos seguintes exemplos de formato: "A5", "f3", "B,3", "D , 7", "g 5", "3A" etc.
10. O jogo DEVE verificar a validade da jogada. Por exemplo, as jogadas "M,3" e "F 13" não são válidas.
11. Após receber e validar a jogada do humano, o jogo DEVE calcular a jogada do computador, da seguinte forma:
    a. Se, na jogada anterior, o computador tiver acertado uma embarcação e ainda não a tiver afundado, este DEVE tentar prosseguir nas casas adjacentes até terminar de afundá-la.
    b. Se, na jogada anterior, o computador não tiver acertado embarcação ou tiver terminado de afundar uma embarcação, este DEVE atacar uma posição aleatoriamente, que ainda não tiver sido atacada. O mesmo serve para a primeira jogada.
12. Um jogador (computador ou humano) que tiver acertado uma embarcação (exceto uma jangada) DEVE jogar novamente logo em seguida.
13. Um jogador que tiver acertado uma jangada do oponente DEVE perder, automaticamente, um de

seus submarinos aleatoriamente (se ainda dispuser). Este jogador NÃO DEVE ter direito a jogar novamente logo em seguida.

14. Após calculada a jogada do computador, o jogo DEVE mostrar novamente os dois tabuleiros atualizados, informando as coordenadas das últimas jogadas do computador e do humano, como exemplificado na Figura 1. Logo em seguida, DEVE pedir uma nova jogada do humano.
15. Caso um dos jogadores tenha terminado de afundar uma embarcação do oponente, o jogo deverá informar o fato. Exemplo: "O humano acabou de afundar um destroyer do computador."
16. O procedimento compreendido entre os requisitos 8 e 14 DEVE ser repetido até que um dos jogadores afunde todas as embarcações de seu oponente, ou afunde todas as embarcações de seu oponente EXCETO a jangada. Acontecendo isto, o jogo deve exibir uma mensagem de quem foi o ganhador, mostrar um placar atualizado das partidas ganhas por cada jogador, e perguntar se o humano gostaria de jogar mais uma vez. Caso escolha não mais jogar, o jogo DEVE ser concluído. Se escolher continuar, o jogo DEVE retornar ao requisito 3.

Requisitos Não Funcionais:

1. O programa DEVE utilizar o mínimo de memória possível.
2. O programa DEVE conter a implementação de **pelo menos** uma estrutura de dados utilizando registros e gerenciamento dinâmico de memória, que DEVE representar o tabuleiro do jogo, onde cada casa é representada por um nó, e cada nó tem referência para os nós de cima, de baixo, da esquerda e da direita, conforme exemplificado abaixo:

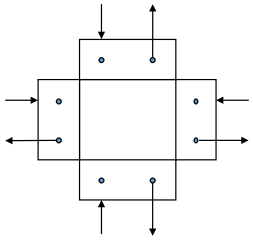


Figura 2: Estrutura de um nó do tabuleiro.

3. O projeto deverá ser constituído por códigos fontes separados, de acordo com características funcionais (Ex. estrutura de dados, programa principal, funções auxiliares, etc.). Todos os arquivos-fonte com o respectivo Makefile para construir o projeto devem ser disponibilizados na entrega.
4. O dimensionamento das variáveis a serem utilizadas deve otimizar a ocupação de memória.

Critérios de Avaliação

A avaliação será realizada em três fases:

1. Análise do código-fonte;
2. Análise da execução do programa (teste);
3. Apresentação do protótipo em laboratório (quando e se solicitado pelo professor).

O código-fonte será avaliado de acordo com os seguintes critérios qualitativos:

i. Eficácia do programa em suprir todos os requisitos funcionais e não funcionais;
ii. Eficiência do programa (otimização);
iii. Organização do código (uso racional de subprogramas, estruturas, etc.);
iv. Legibilidade do código (uso de endentação e semântica dos identificadores de variáveis);
v. Documentação (comentários dentro do código fonte).

Obviamente, funcionalidades adicionais às que foram solicitadas neste documento são bem-vindas e serão gratificadas na nota (na medida do possível). O código-fonte deve conter, em comentário no início, os nomes e matrículas dos alunos que compõem o grupo, assim como uma descrição sucinta da atuação de cada um no desenvolvimento do projeto. O código-fonte deve ser submetido na data fixada através de servidor de *upload*, a ser definido. A apresentação do protótipo, se necessária, será marcada em seguida.

Lembramos que todos os programas serão submetidos a análise léxica automática, que pode evidenciar cópia de código.

Os trabalhos serão corrigidos no **Linux**. Portanto, certifique-se que o trabalho feito no Windows também compila e roda no Linux.

Prazo de Entrega: 1º de fevereiro de 2016, até 23:59.

Upload através do SIGAA (Turma Virtual)

Não serão aceitas entregas por e-mail (a não ser que haja algum problema com a submissão pelo *SIGAA*).