Universidade de Brasília
Departamento de Ciência da Computação/IE
Semestre: 02/2010
Disciplina: Computação Básica

Trabalho Prático 3

O objetivo desta terceira etapa é que o aluno aprenda a usar **estruturas de dados do tipo vetor, matriz e subalgoritmos (funções)** em algoritmos, bem como implementá-los corretamente em programas.

Especificação da terceira etapa:

Problema: Implementar um jogo similar ao jogo Dominó, que conterá apenas 10 pedras. Cada pedra possui dois lados (A e B), sendo que em cada lado os números podem variar de 0 a 3. Os dois jogadores serão controlados pelo usuário, que fará todas as escolhas de jogadas. Cada jogador receberá metade das pedras do Dominó, ou seja, 5 pedras. Começa o jogador que tiver saído com a pedra PELADO E PELADO, ou seja, a pedra que tem o número 0 (zero) tanto no seu lado A, quanto no seu lado B. A primeira jogada que este jogador irá fazer, será: colocar a pedra PELADO E PELADO no centro da mesa (que será representada por um vetor com 19 posições). Só serão feitas (possíveis) as jogadas em que o usuário escolher encaixar uma de suas pedras pelo lado que contenha um número igual ao número do lado aberto da pedra que está na ponta da mesa em que o usuário escolheu jogar. Não é possível, por exemplo, encaixar a pedra PELADO (número zero) E TERNA (número 3) pelo lado da TERNA na pedra da mesa cujo lado aberto é o ÁS (número um), pois ÁS e TERNA são diferentes. Neste caso o usuário será obrigado a escolher outra jogada. O jogo estará encerrado em duas situações: (1) quando acabar as pedras de um dos jogadores (neste caso ele é o vencedor), ou (2) quando nenhum dos dois jogadores tiver mais pedras que se encaixem em pelo menos uma das duas pontas da mesa (neste caso o vencedor será aquele que ficou com menos pedras na mão). O jogo deverá detectar automaticamente o final do jogo e deverá informar qual jogador foi o vencedor e quantas pedras restaram na mão de cada jogador.

Nomenclatura para número nas pedras:

0 (zero): PELADO

1 (um): ÁS

2 (dois): DUQUE

3 (três): TERNA

Detalhamento do problema:

1. As unidades de entrada deste trabalho deverão ser o teclado para inserção das escolhas de jogadas que o usuário deverá fazer. A unidade de saída será o monitor de vídeo.

2. Para facilitar o manuseio do programa, deverão ser dadas mensagens explicativas em alguns casos conforme indicado posteriormente nesta especificação.

3. A seguir é apresentada uma SUGESTÃO de vetores, matrizes e variáveis a serem criadas para uma possível implementação. Caso o aluno pense em uma solução alternativa mais otimizada não há problema em modificá-la ou fazer diferente:
   - Dois vetores de 10 posições que conterão o número (int) do lado A e do lado B de cada PEDRA. Estes vetores deverão ser preenchidos no início da execução do programa. Eles não serão modificados durante o resto da execução;
   - Um vetor de 10 posições que conterá um número (int) para indicar o ESTADO DE CADA PEDRA de cada pedra: 0 (zero) significa que a pedra não está jogada, 1 (um) significa que a pedra está jogada na mesa com o lado A para cima e o lado B para baixo, 2 significa que a pedra está jogada na mesa com o lado B para cima e o lado A para baixo. Este vetor deverá ser inicializado com o valor 0 (zero) em todas as suas posições no início da execução. Conforme avança o jogo, seu valores irão sendo modificados;
   - Dois vetores de 5 posições que conterão um número (int) que irá representar o índice de uma pedra. Estes vetores representam as MÃOS DOS DOIS JOGADORES, ou seja, as pedras que cada jogador tem na mão. Seus valores serão inicializados randomicamente no início da execução do programa. Todos os valores deverão ser diferentes entre si (pois nenhuma pedra se repete no Dominó). EXEMPLO para o significado do vetor: pedras_ladoA[jogador0_indices[0]] = 2, significa que o lado A da primeira pedra do jogador 0 vale 2 (DUQUE); se pedras_ladoB[jogador1_indices[4]] = 3, significa que o lado B da última pedra do jogador 1 vale 3 (TERNO); se pedras_estado[jogador0_indices[2]] = 2, significa que a terceira pedra que saiu com o jogador 0 não está mais na sua mão, agora está jogada na mesa com o lado B voltado para cima e o lado A voltado para baixo;
   - Um vetor de 19 posições que conterá um número (int) com o mesmo significado do vetor acima, mas este irá representar a MESA. Exemplo: se pedras_estado[mesa_indices[10]] = 1, significa que a pedra encaixada no lado de baixo da pedra PELADO E PELADO (sempre fica na posição 10-índice9 do vetor mesa_indices) está com o lado A voltado para cima e o lado B voltado para baixo. Não é necesário inicializar nenhuma posição deste vetor antes da execução dos turnos. Conforme o jogo avança, este vetor deve ser modificado, recebendo como valores os índices das pedras que estão sendo jogadas na mesa. Lembre-se que a primeira pedra (PELADO E PELADO) sempre deve estar armazenada no meio deste vetor;
   - Duas variáveis inteiras para indicarem onde a MESA COMEÇA e onde a MESA TERMINA. Exemplo: se mesa_inicio = 7 e mesa_fim = 12, a primeira posição do vetor mesa_indices que pode ser lida/modificada/etc é a oitava posição (índice 7) e a última posição do vetor mesa_indices que pode ser lida/modificada/etc. é a décima terceira posição (índice 12);
   - Uma matriz de caracteres 19x125 que será inicializada com espaços (o caracter ‘ ‘). Esta matriz deve ser atualizada e impressa conforme o andamento do jogo. Ela REPRESENTARÁ GRAFICAMENTE A MESA DO JOGO (ver as imagens do anexo). No Linux, as janelas do Terminal são “flexíveis” em seu comprimento, portanto é possível “aumentar” a janela manualmente com o mouse, de forma que hajam mais colunas para a impressão desta matriz. Já no Windows, não é possível “aumentar” a janela do prompt de comando. É necessário clicar na barra superior de

título da janela com o botão direito do mouse, clicar na opção Propriedades do menu que aparece e, na aba Layout, na área Tamanho da janela, aumentar o campo Largura (número de colunas) para o tamanho necessário para uma boa visualização da saída do programa;

- Uma matriz de caracteres 9x40 deverá ser criada, APENAS DENTRO DE UMA FUNÇÃO, que irá representar graficamente as pedras que estão na mão do jogador da vez. Ou seja, ela será criada e preenchida toda vez que esta função for chamada, diferentemente da matriz que representará a mesa do jogo, que será declarada na main(). Abaixo da impressão desta matriz, deverão ser impressos números para representar as opções que o usuário terá para escolher. Ver as imagens do anexo.

4. A execução de um turno deverá ser a seguinte:
   - Informar na tela qual jogador irá jogar naquela vez, seguido de um "Tecle <enter> para prosseguir";
   - Imprimir as pedras do jogador;
   - O computador verifica se o jogador da vez pode jogar (isto é, se ele tem alguma pedra que serve na mesa). Caso positivo, deverá solicitar ao usuário que informe o número da pedra que deseja jogar, o lado e em qual ponta da mesa ele deseja encaixar (esta parte da execução não deve acontecer no primeiro turno, pois a jogada é pré-definida, isto é, jogar a pedra PELADO E PELADO no centro da mesa). Isto deve se repetir enquanto o jogador não escolher uma jogada possível ou não confirmar que deseja fazer a jogada. Caso o jogador da vez não possa jogar (isto é, não tem pedra que serve na mesa), o jogo deverá imprimir uma mensagem explicativa dizendo que este jogador passou a vez;
   - Atualizar os vetores e variáveis conforme a jogoda que foi feita;
   - Atualizar a matriz da mesa com a última jogada que foi feita;
   - Imprimir a mesa (não é necessário quando o jogador tiver passado a vez, ou seja, quando ele não pôde jogar);
   - Imprimir uma mensagem explicativa de qual pedra o jogador jogou.

5. Para fazer o sorteio das pedras, faça a seguinte chamada: `srand (time (NULL));` Esta chamada "plantará uma semente" para o algoritmo de pseudo-randomização de valores inteiros. Para randomizar um valor entre 0 e x-1, chame a expressão `rand () % x`. Ou seja, para o trabalho 3 a chamada será `rand () % 10`. Sugestão: faça um vetor de 10 posições para, nele, distribuir randomicamente (aletoriamente) os valores entre 0 e 9 e depois passe metade do vetor para um dos vetores de índices de um jogador, depois passe a outra metade para o outro vetor de índices do outro jogador.

6. Neste trabalho você DEVE modularizar o seu programa criando funções. Conforme já dito em sala de aula, funções tornam o seu programa mais legível, mais fácil de ser entendido, e mais curto, já que evita a duplicação de código para uma mesma ação.
   *** Deverão ser criadas, no mínimo, as seguintes funções:**
   1 – Função para fazer todas as inicializações necessárias no início do programa.
   2 – Função para fazer todos os procedimentos da execução de um turno (lembre-se que o primeiro turno está pré-definido: quem saiu com a pedra PELADO E PELADO, joga esta pedra).
   3 – Função para testar se o jogo acabou, ou seja, se um dos jogadores não tem mais pedras (isto é, ele "bateu"), ou se os dois jogadores passaram a vez (isto é, ninguém tem

mais pedras que servem na mesa). O retorno dessa função deverá ser utilizada na condição de um loop do-while para dar continuidade no jogo ou não.
4 – Função para verificar se um jogador (cujo vetor de índices deve ser passado como parâmetro) tem alguma pedra que encaixe na mesa.
5 – Função para contar quantas pedras um jogador (cujo vetor de índices deve ser passado como parâmetro) ainda tem em sua mão (usar o vetor que indica o estado de uma pedra para fazer as verificações: 0 (zero) neste vetor indica que a pedra não está na mesa).
6 – Duas funções que retornam o número das pedras que estão nas pontas, respectivamente, de cima e de baixo da mesa. Ou, ao invés disso, podem também ser criadas duas variáveis na main() que devem ser atualizadas com a chamada desta função.
7 – Função para criar, preencher e imprimir a matriz (isto é, a matriz só estará no escopo desta função) que representa graficamente a mão do jogador da vez.
8 – Função para atualizar a matriz da mesa, conforme a última jogada.
9 – Função para imprimir a matriz da mesa.
Você pode criar outras funções (não obrigatórias) caso ache necessário!

7. Sugestão: para a mensagem explicativa de qual pedra foi jogada, faça uma função que receba como parâmetro a referência de uma string, modifique esta string com o nome da pedra e retorne a própria referência da string, ou seja, poderá ser feita uma chamada como esta:
```
char nome_da_pedra[16];
printf ("%s", nomeDaPedra (nome_da_pedra, ... ));
```
E obter uma saída como esta:
PELADO E TERNA
Um protótipo para uma função como esta seria:
```
char* nomeDaPedra(char *nome_da_pedra, ...);
```
E o valor seria retornado na forma:
```
return nome_da_pedra;
```

8. Crie variáveis e nomes de funções mnemônicos!!!

9. NÃO utilize variáveis globais!

10. Nas funções, utilize **passagem de parâmetros por valor e por referência**, conforme necessário.

11. Faça diversos testes no seu jogo, jogando é claro. Bom trabalho e boa diversão!

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: **09/01/2011 (domingo) até às 23:55 hs**.