**Relatório do Programador**

**Batalha**

**Naval**

**Aluno: Marcelo Queirós Pinto**

**Aluno 60102**

**Engenharia Informática**

**Unidade Curricular: Arquitetura de Computadores**

**Docente: José Cardoso**

**2015**

**Começo por especificar alguns dos objectivos mais importantes do meu código:**

* Criar uma matriz 10x10, ou seja com 100 casas, onde serão do ponto de vista do utilizador ‘escondidos’ os alvos do jogo.
* Criar 3 barcos com uma casa, 1 submarino com 3 casas horizontais e um porta-aviões com 4 casas (3 horizontais e uma por cima da casa do meio).
* Esconder de forma aleatória todas as peças do jogo, não podendo obviamente nenhuma delas calhar no mesmo sítio de outra.
* Criar um contador de tentativas que actualiza a cada jogada.
* Não permitir jogar 2 vezes na mesma posição onde está um barco, depois de acertar nessa posição e tentar segunda vez e próximas, o programa irá informar que já acertou nesse alvo (repetido) e não irá contabilizar esse barco segunda vez nem irá contabilizar essa tentativa.
* Fazer do jogo o mais parecido com o verdadeiro jogo “Batalha Naval” e com o máximo de recursos.

**Linha 1-92:**

Nestas primeiras linhas estão todas as linhas não executáveis do código, a maioria não necessita de explicação por serem de análise óbvia, mas vou falar de algumas delas no decorrer deste relatório conforme forem utilizadas.

**Linha 93-1344: (Aleatório)**

Criei 5 variáveis nas quais o meu código vai preencher aleatoriamente:

* barco1 db '000'
* barco2 db '000'
* barco3 db '000'
* submarino db '000','000','000'
* portaavioes db '000','000','000','000'

Cada um destes grupos ‘000’ representa uma casa, lá estarão as coordenadas para uma casa que no máximo ocuparão 3 posições cada uma (exemplo: 10B), caso só ocupem 2 posições ficará no formato ‘2B0’, com um ‘0’ na última posição.

Utilizo a função MOV AH, 2Ch // INT 21h em toda a base desta parte do código, esta função copia para dl um valor de 0 a 100 (busca os centésimos de segundo ao relógio).

Para o barco 1, utilizo essa função e divido 100 em 10 partes e consoante o número que tiver em dl, será copiado uma letra de A a J para BARCO1[1]. Depois utilizo a mesma função, e consoante o número que tiver em dl é copiado para BARCO1[0] um número de 1 a 10, da mesma forma que fiz para as letras. A exceção é se o número for 10, nesse caso, a letra passa para BARCO1[2], ‘1’ é copiado para BARCO1[0] e ‘0’ para BARCO1[1]. O processo é exactamente igual para o barco 2 mas agora o barco 2 não pode ser igual a barco 1. Então no fim é feita uma verificação para ver se as coordenadas do barco 2 são iguais ás do barco 1, se forem iguais, volta a ser repetido o processo do barco 2 para gerar outras coordenadas. O barco 3 é o mesmo processo mas no fim compara com o barco 1 e 2, caso as coordenadas do barco 3 sejam iguais a algum dos barcos anteriores volta a ser repetido o processo aleatório do barco 3.

Para o submarino e porta-aviões utilizo o mesmo processo mas só utilizo a função MOV AH, 2Ch // INT 21h para gerar a parte do meio do submarino e do porta-aviões, as outras são criadas a partir dessa para os lados e no caso do porta-aviões para cima. Verifico se o submarino é igual ao barco 1, barco 2 ou barco 3, faço isto para as 3 partes do submarino, o que dá um total de 9 condições, se alguma das partes for igual a algum barco, é gerado outro submarino e apagado o anterior. Para verificar o porta-aviões, verifico se cada uma das 4 partes do porta-aviões é igual ao barco 1, barco 2, barco 3, ou a uma das 3 partes do submarino, o que dá um total de 6 condições para cada parte do porta-aviões, no total dá 4 partes x 6 condições = 24 condições.

**Linhas 1345-1519:**

Já terminado o processo aleatório, das linhas 1347 a 1357 apresento o titulo e informação inicial e quando o utilizador pressionar uma tecla, o ecrã é limpo com a função MOV AX, 03h // INT 10h e começa verdadeiramente o jogo.

Na linha 1361 encontramos o bloco INICIO declarado, que para onde salta o código depois de ser verificado se o jogador ganhou ou não, perto do fim do código, se ainda não ganhou salta sempre para este bloco.

A seguir á declaração deste bloco imprime a tabela principal do jogo, o contador de tentativas, se o utilizador escolheu o modo hacker, imprime as coordenadas dos barcos, a primeira coordenada do submarino e a primeira do porta-aviões, que já chega para identificar totalmente todo o submarino e porta-aviões, e imprime a legenda, sempre utilizando a função 9, INT 21h.

**Linhas 1520-1579: (pedir número e letra)**

Feito isto, é pedido ao utilizador a linha e a coluna, primeiro a linha que corresponde ao número, se o número for menor que ‘1’ ou maior que ‘9’ significa que corresponde a algo diferente do pretendido na tabela Ascii e portanto é pedido novamente o número, não aceitando esse. Se o número for igual a ‘1’, o programa espera que o utilizador introduza outro carater, isto porque pode ser 1 ou 10, então se o segundo carater for enter, corresponde ao numero 1, se o segundo carater for 0 corresponde ao numero 10, se não for enter nem 0, o programa volta a pedir o número todo. Depois pede a letra, o programa aceita apenas letras de A a J, tanto maiúsculas como minúsculas, pois se forem minúsculas o programa converte para maiúsculas.

**Linhas 1580-1883: (verificar se falhou ou acertou)**

Depois de receber o número e letra do utilizador, vou verificar se acertou num barco/submarino/porta-aviões, se falhou ou se acertou num barco/submarino/porta-aviões que já tinha acertado e se tal acontecer essa jogada não será validada nem contará como tentativa, para isso declarei variáveis que servem como ‘flags’, cada alvo tem uma flag. Quando o jogador acerta nas coordenadas, a flag incrementa, se a flag é ‘1’ (primeira vez que acerta), o jogador acertou o alvo, se a flag for maior que ‘1’ (significa que já não é a primeira vez que acerta) o jogador não acertou no alvo.

* verificabarcos db '000' ; barco 1 / 2 / 3
* verificasubs db '000' ; 3 partes do submarino
* verificapavioes db '0000' ; 4 partes do porta-aviões

Então primeiro comparo a jogada do jogador com o barco 1, depois barco 2, barco 3, submarino e porta-aviões, se bater certo com as coordenadas de algum alvo e a respectiva flag for ‘1’, então o jogador acertou e passa para o bloco de código respectivo, se o jogador acertou e a flag for maior que 1 então passa para o bloco de código respectivo que imprime um texto a dizer que repetiu coordenadas, se falhou todas as coordenadas passara para o bloco de código respectivo que dirá que falhou.

Para ver se acertou num alvo faz o seguinte: compara o registo ch (que guarda o segundo algarismo caso o número seja ‘10’) com a respectiva posição do vetor que guardaria esse numero (ex: barco1[1]). Se ch for diferente de ‘0’, vai comparar a jogada do jogador com as duas primeiras posições do vetor do alvo. Se ch for ‘0’, vai comparar a jogada do jogador com a primeira posição do vetor e a ultima, pois nesse caso a jogada tem 3 digitos. Faz isto para todos os alvos.

**Linhas 1884-2140:**

Depois disto, o programa passa para o bloco de código correspondente. Há 5 blocos fundamentais para que o programa pode passar:

* Acertou barco (acertoubarco);
* Acertou submarino (acertousub);
* Acertou porta-aviões (acertouavi);
* Falhou alvo (falhoubarco);
* Alvo repetido (mesmobarco).

Os primeiros 3 são muito semelhantes, apenas mudando na mensagem que vão apresentar no ecrã. Nestes 3 blocos a primeira coisa que acontece é o incremento do número de tentativas, é incrementado o primeiro dígito, se este for 9, então incrementa o segundo mas se o segundo também for 9 então vai incrementar o terceiro, se incrementou o segundo põe o primeiro a ‘0’ e se incrementou o terceiro põe os dois primeiros a ‘0’. O número máximo de tentativas para isto funcionar é ‘999’. Depois o registo bp passa a ser igual a 1, já vamos ver mais á frente o objectivo (quando falha bp=0), e é escrita a jogada do jogador e o respectivo texto.

O quarto bloco é em tudo semelhante exceto no texto, o texto diz que o jogador falhou e que a variável bp é igualada a 0.

O quinto bloco é diferente pois ao entrar neste bloco não é incrementado o número de tentativas uma vez que o jogador acertou num alvo que já tinha acertado antes, é apenas escrita a respectiva jogada e texto e salta logo para o início novamente uma vez que não é preciso fazer mais nada.

**Linhas 2141-2302**

Nesta parte do código, inicialmente vou descobrir a posição do vetor que vou alterar, depois mais adiante vou descobrir a linha. Vou começar por explicar como funciona a tabela:

Cada linha da tabela é um vector com 42 posições, mas apenas 10 posições desse vector vão ser alteradas no decorrer do código. Isto porque toda a formatação da linha ocupa posições do vector mas só vou alterar as 10 posições que têm um ‘~’ que são as que interessam. Assim o desafio é descobrir o número das posições que quero alterar: O vector começa em 0, de 4 em 4 posições há um ‘~’, então quero alterar as posições 3, 7, 11, 15, 19, 23, 27, 31, 35 e 39 de cada linha, seguindo uma sucessão 4n-1, em que n representa a posição real que queremos alterar, por exemplo a posição 10 que corresponde á letra J será 4x10-1=39. Funciona assim para todas as linhas. Então se o utilizador introduzir a coordenada ‘7C’, vou alterar a linha 7 na posição 4x3-1=11 (letra c corresponde a 3).

Assim vou ver a letra que o utilizador inseriu (está guardada em bl), e consoante a letra vou atribuir a SI o valor correspondente (exemplo: se em bl está G, em SI vai estar 27), seguindo o que expliquei no parágrafo anterior.

Depois vou comparar bp (registo que guarda 1 se acertou e 0 se falhou) com 1, se for 1 vai por o símbolo que identifica que acertou num alvo (☺) na linha e posição corresponde, se for 0 vai por o símbolo que identifica que falhou (♦) na linha e posição correspondente.

Para isto apenas compara o registo cl com ‘2’, ‘3’, ‘4’, ‘5’, ‘6’, ‘7’, ‘8’ e ‘9’ e salta para o bloco de código correspondente que altera a linha. Não compara com ‘1’ porque um pode significar 10 ou 1, então no fim compara ch com ‘0’ (ch é 0 quando o numero é 10), se for 0 vai alterar a linha 10, se não for 0 altera a linha 1.

Quando saltar para o bloco de código corresponde vai alterar a linha certa na posição SI e atribuir o símbolo correspondente.

**Linhas 2303-2403: (END)**

Utilizei o registo DI para guardar o número de barcos em que o jogador já acertou, o submarino conta como 3 barcos (tem 3 partes) e o porta-aviões conta como 4 barcos (tem 4 partes), assim 3 barcos + submarino (3 partes) + porta-aviões (4 partes) = 10 barcos. Então o jogo acaba quando DI for igual a 10.

Então, nesta parte, a primeira coisa que faço é ver se BP é igual a 1, se for é necessário incrementar DI e existe a possibilidade de o jogador ganhar o jogo (pois DI pode ficar a 10), se BP for 0, salta logo pro inicio novamente.

Então, se BP for 1, o programa incrementa DI e verifica DI é igual a 10, se não for 10 salta para o início, se for 10 vai terminar o jogo, actualiza a tabela e o contador de tentativas (uma vez que só são actualizados no inicio), e apresenta a mensagem a dizer que ganhou, o número de tentativas, o texto “Jogo terminado” e “pressione qualquer tecla para fechar”, quando o utilizador premir uma tecla o programa termina e retorna o controlo para o sistema operativo. Ao mostrar o número de tentativas no final não apresenta os zeros, isto é, se as tentativas forem ‘5’ por exemplo ele apresenta ‘5’ e não ‘005’, para isto comparo o terceiro dígito com ‘0’, e só mostra se não for igual, para o segundo dígito igual, mas se mostro o terceiro digito também vai mostrar o segundo obviamente.