MAC 0211 - Laboratório de programação - 1º semestre 2015

Grupo:

Gabriel Baptista nUSP:8941300

Helio Hideki Assakura Moreira nUSP:8941064

Pedro Ivo Siqueira Nepomuceno nUSP:8941321

**EP2 - Bombardeio Naval - Documentação de Desenvolvedor**

Este EP consiste na implementação de um programa que servirá como um joguinho inspirado em batalha naval, onde o usuário deve digitar via teclado o arquivo que contém o

mapa do oceano com as devidas embarcações já posicionadas, também em qual coluna do mapa seu barco será posicionado. O jogo consiste em movimentar o seu barco(a remo)

da primeira linha do mapa com a coluna desejada até uma das colunas na última linha do mapa. Caso obtenha êxito o usuário ganha.

O código em si:

Criamos um struct para o barco a remo. Nele temos a posição do barco(linha,coluna) e um char que representará o barco(B),um struct para representar o mapa com uma matriz e o tamanho da mesma(linhas,colunas) e um struct para um tiro disparado numa coordenada (c\_linha,c\_coluna).

Conforme o enunciado, dividimos o programa em 10 grandes partes:

(1)cpp leia\_mapa(char\* arq)

Para isso, criamos a função, que recebe como parâmetro uma string com o caminho do arquivo que contém o mapa. Dentro da função a memória necessária é alocada e são colocadas as embarcações em questão. A função retorna um ponteiro para o campo.

(2)void escreva\_mapa\_tela(cpp cp)

Função que recebe como parâmetro o campo, e escreverá o mesmo na tela, tendo cuidado para não dar informações que privilegiariam o usuário. Sendo assim, inicialmente o campo é preenchido todo com '-' e a posição inicial do barco fica marcada com 'B'. Após os movimentos e os tiros, pode-se ver na tela os seguintes caracteres:

'T' : representa uma posição pela qual o barco já passou

'B' : representa a atual posição do barco

'=' : se o tiro acertou o local correspondente a água

'\*' : se o tiro acertou alguma embarcação do local

'+' : se o tiro acertou alguma posição na qual o barco havia passado

'!' : se o tiro acertou a atual posição do barco

Se houver qualquer caracter diferente dos descritos acima o programa colocará '-' no local.

(3)void escreva\_mapa\_arquivo(char \*arq,cpp cp)

Função que recebe como parâmetro uma string que representa o caminho para o qual o mapa deve ser salvo em um arquivo, e um apontador para o campo.

(4)barco posiciona\_barco(cpp c)

Função que recebe como parâmetro o campo. Pedirá para o usuário informar em qual posição deseja que seja inserido o barco e devolve um tipo barco que descreve B.

(5)int rema\_barco(cpp c, barco ba, char cmd)

Função que recebe como parâmetro o campo, o barco e o char que representa o movimento desejado pelo usuário e o fará se possível, retornando zero. Caso contrário, retorna 1, que servirá para contar nos movimentos inválidos.

(6)void dispara\_tiros(cpp c)

Função que recebe como parâmetro o campo c e chama a função (7)coordenadas\_tiro que sorteia uma coordenada onde será disparada um tiro.

(7)tiro coordenadas\_tiro(int l, int c)

Função que recebe dois inteiros(l,c) que representam as dimensões do mapa e sorteia valores entre 0-(l-1) e 0-(c-1) e devolve esses valores numa variável do tipo tiro.

(8)int sorteia(int k)

Função que recebe como parâmetro um inteiro(k) e sorteia um número entre 0 e k-1 usando a função rand() própria de C.

(9)int identifica\_alvo\_atingido(cpp c,tiro t)

A função recebe como parâmetro o campo e o último tiro realizado. Verifica se a coordenada em questão possui alguma das seguintes letras : 'C','P','H','D','S' , chamando a respectiva função afunda\_<nome\_da\_embarcação> (exceto no caso do submarino).

O campo é alterado da seguinte forma:

'=' : se o tiro acertou o local correspondente a água;

'\*' : se o tiro acertou alguma embarcação do local;

'+' : se o tiro acertou alguma posição na qual o barco havia passado;

'!' : se o tiro acertou a atual posição do barco

Se acerta a coordenada atual do barco, o jogo é finalizado.

(10)afunda\_embarcação

O algoritmo leva em conta que não há embarcações coladas, sendo assim a ideia básica para todos é analisar a vizinhança da coordenada em que ocorre o tiro.

*(10.1)void afunda\_destroyer(int i,int j,cpp c)*

Função que recebe como parâmetro 2 inteiros(i,j) que representam a coordenada em que o tiro atingiu no mapa e o mapa(c) em questão. Como o destroyer tem apenas 2 partes inicialmente,basta atribuir '\*' a posição vizinha de (i,j) que contém 'D', pois em (i,j) já temos '\*' colocado pela função (9).

*(10.2)void afunda\_cruzador(int i,int j,cpp c)*

Função que recebe como parâmetro 2 inteiros(i,j) que representam a coordenada em que o tiro atingiu no mapa e o mapa(c) em questão. Como o cruzador tem 4 partes, é preciso atribuir '\*' a 3 outras posições que são adjacentes entre si e com (i,j) contendo 'C'. Sendo assim, é necessário criar 2 vetores para guardar as coordenadas em que foi atribuido '\*' para posterior análise de seus vizinhos e achar outro contendo 'C'.

*(10.3)void afunda\_porta\_aviao(int i,int j,cpp c)*

Função que recebe como parâmetro 2 inteiros(i,j) que representam a coordenada em que o tiro atingiu no mapa e o mapa(c) em questão. Como o porta\_avião tem 5 partes, é preciso atribuir '\*' a 4 outras posições que são adjacentes entre si e com (i,j) contendo 'P'. Sendo assim, é necessário criar 2 vetores para guardar as coordenadas em que foi atribuido '\*' para posterior análise de seus vizinhos e achar outro contendo 'P'.

*(10.4)void afunda\_hidro\_aviao(int i,int j,cpp c)*

Função que recebe como parâmetro 2 inteiros(i,j) que representam a coordenada em que o tiro atingiu no mapa e o mapa(c) em questão. Como o hidro\_avião tem 3 partes, é preciso atribuir '\*' a 2 outras posições que são adjacentes entre si e com (i,j) contendo 'P'. Sendo assim, é necessário criar 2 variáveis para guardar as coordenadas em que foi atribuido '\*' para posterior análise de seus vizinhos e achar outro contendo 'P'.

Além dessas 10 partes, temos algumas funções que nos auxiliam, sendo elas:

int acabou(cpp c, barco b, int cfim)

Função que verifica se o jogo acabou ou não, retornando 1 e imprimindo uma mensagem caso o jogo acabe. Devolve 0 caso contrário.

void imprime\_campo(cpp c)

Função que recebe um campo e o imprime na tela.

Parte extra:

void escreva\_mapa\_resposta(char \*arq,cpp cp)

Função que serve como uma espécie de resposta para usuário. Mostra, num arquivo de saída, o que tem em cada posição do campo a cada rodada. O usuário pode analisar qual teria sido o melhor comando para cada movimento. O arquivo só poderá ser acessado após o final do jogo.

O algoritmo em si:

O usuário deve entrar com o caminho do arquivo que contém o mapa que deseja atravessar. Em seguida o barco é posicionado. Após posicioná-lo, o mapa é impresso na tela e nos arquivos de saída. Em seguida, o jogo é iniciado em um laço. O usuário deve digitar b,c,d ou e, que serão os comandos de movimento do barco. A função rema\_barco é chamada, que verificará se é possível fazer o movimento. Se for possível retorna 1, senão 0, valores estes salvos na variável gotit que será utilizada mais pra frente. Após o movimento,o mapa é escrito na tela e são disparados 3 tiros na rodada, escrevendo novamente o mapa na tela. O valor de gotit é verificado. Se for 1 significa que o movimento foi válido, assim count é zerado. Caso contrário, count recebe count+1, para futura verificação(count==3)... após isso se verifica se o jogo acabou: caso não, o laço é continuado.