PONTIFÍCIA UNIVERSIDADE CATÓLICA DE MINAS GERAIS

Trabalho Prático
Algoritmos e Técnicas de Programação

Guilherme Henrique, Jean Carlo, Vinícius Morais, Lucca Cenisio

Campus São Gabriel,
2023

Objetivo do programa:

O trabalho prático consiste na implementação de um jogo, Campo Minado. O objetivo trata-se de ler um arquivo de texto onde encontram-se as informações sobre a quantidade de linhas, colunas e bombas. Após isso, construir um mapa correspondente às informações e fazer com que o usuário consiga jogá-lo ao inserir a linha e coluna desejada .

Detalhes de implementação:

Neste tópico iremos apresentar a detalhação do programa, explicando suas funcionalidades, métodos criados, testes e resultados obtidos.

Métodos:

- **VerificaDerrota:**

  O método para verificar a derrota do jogador é simples, recebe como parâmetro uma matriz em tipo char e dois números (a , b) de tipo int, eles representam a coordenada que o jogador inseriu, caso a coordenada (a , b) da matriz recebida como parâmetro seja igual a 'B', que é a letra que representa uma bomba o jogador perde e uma mensagem de derrota é exibida a ele. Esse método é recebido dentro do while onde o jogo acontece, assim toda vez que o jogador digitar uma coordenada ela será verificada para checar a derrota, a derrota é retornada em tipo boolean, sendo true caso a posição seja igual a 'B'.

- **MapaCoberto:**

  Nesse método recebemos apenas dois parâmetros (int l, int c), referentes ao tamanho de linhas e colunas lidas no arquivo, com isso criamos uma matriz com l linhas e c colunas e preenchemos com a letra 'X', simbolizando uma posição ainda não descoberta, sendo assim seu retorno conside tem uma matriz do tipo char totalmente preenchida por letras X.

- **Gabarito:**

  Aqui criamos um mapa totalmente descoberto, para isso recebemos os 3 números lidos no arquivo que representam respectivamente o número de linhas, colunas e bombas. O método é separado em duas partes, a primeira criamos uma matriz com a quantidade de linhas e colunas lidas, adicionamos de maneira randômica as bombas com auxílio de um for que rodará 3 vezes caso a quantidade de bombas seja 3, por exemplo. A segunda parte consiste em ler posição por posição dessa matriz e contar quantas bombas adjacentes a essa posição lida existem, com um contador fizemos essa contagem e adicionamos seu valor a posição, por exemplo gabarito [i,j] tem 2 bombas adjacentes então

gabarito [i,j] = contador. Assim, criamos um mapa de campo minado completo, com bombas, espaços livres e números que indicam a presença de bombas.

- **VerificaWin:**

  O método VerificaWin recebe como parâmetro duas matrizes que estão relacionadas a matriz denominada campo e a matriz com o mapa do jogo, esse método lê as duas matrizes de forma separada, ao ler a matriz campo ele conta quantas letras 'X' ainda restam, ou seja, quantas posições ainda não foram reveladas, durante a leitura da matriz mapa ele conta quantas bombas o mapa possui, caso o contador de letras X seja igual ao número de bombas o jogador ganha o jogo. O método está inserido dentro do while onde o mapa é exibido ao jogador, assim acontecendo a verificação a cada rodada.

- **RevelaMapa:**

  O método mais complexo é o RevelaMapa, nele recebemos como parâmetro duas matrizes relacionadas respectivamente ao mapa com o gabarito e ao mapa com X, também recebemos as coordenadas digitadas. Pegamos a coordenada digitada pelo jogador e verificamos se em seu entorno existem posições, caso haja posições pegamos a posição do gabarito e inserimos na posição respectiva ao mapa X, por exemplo, a posição digita foi m1[a, b] verificamos se a posição acima (m1[a + 1, b]) é igual a uma bomba, caso não seja adicionamos a coordenada a matriz de retorno do método mais a sua posição de cima, verificando assim todo seu entorno criando uma terceira matriz que é retornada pelo método, nela temos todo o mapa por X e a coordenada digita e seus adjacentes viram a posição do gabarito.

- **PrintaMapaCoberto:**

  Esse método apenas recebe o número de linhas e colunas e printa um mapa coberto por X ao jogador antes do jogo começar, com isso ele poderá saber o tamanho do mapa que está jogando.

Main:

- **Leitura do arquivo:**

  Nosso primeiro passo foi definir uma variável do tipo string para receber o caminho que contém o arquivo txt. Logo em seguida declaramos as variáveis: "linha", "coluna" e "bombas" para armazenar os valores do mapa. Criamos um bloco try-catch para tratar as exceções que possam ocorrer. Definimos um objeto "StreamReader" dentro do bloco "using".

Usamos o método ReadToEnd() no objeto StreamReader para ler todo o conteúdo do arquivo e armazená-lo na variável “conteúdo” em forma de uma única string. Separamos a string utilizando o método .Split() em três partes. Usamos um for para percorrer a string e com um if(int.TryParse) tentamos transformar os valores em números inteiros. Se a conversão for realizada com sucesso os valores são atribuídos respectivamente em “linha” “coluna” e “bomba” e encerra a validação com o “break;” Logo em seguida printamos no terminal a quantidade de linhas colunas e bombas. o catch no fim do código trata a exceção de arquivo não encontrado ou “FileNotFoundException”.

- **Recebimento dos métodos:**
  Os métodos foram colocados em outro arquivo para melhor leitura e entendimento do código, com isso tivemos que acrescentar Métodos.NomeDoMetodo, assim referenciamos ele de outro arquivo.

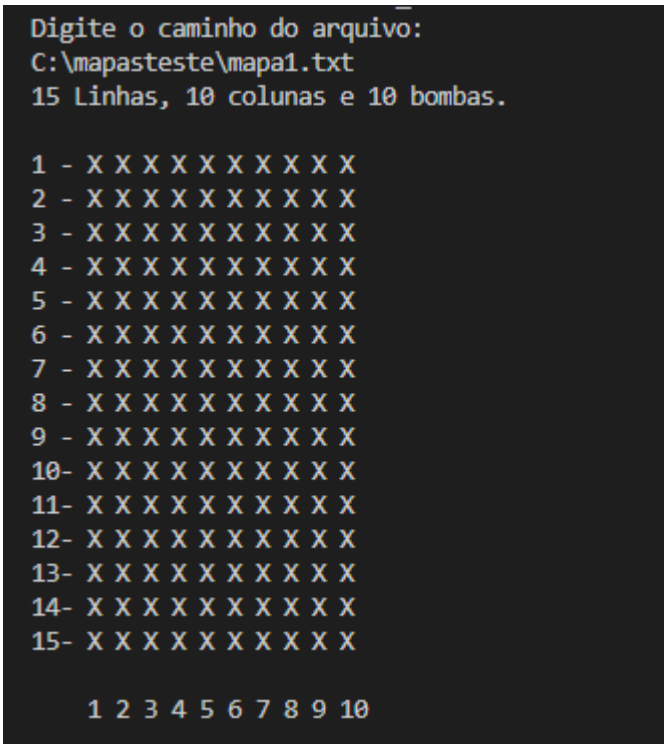
- **Repetição do mapa:**
  Todo o jogo acontece dentro de um while, lembrando que ele é encerrado com um break caso entre no if de verificação de derrota ou vitória. dentro do while o usuário digita as coordenadas de linha e coluna, o mapa é printado atualizado a cada ciclo.

Como executar:

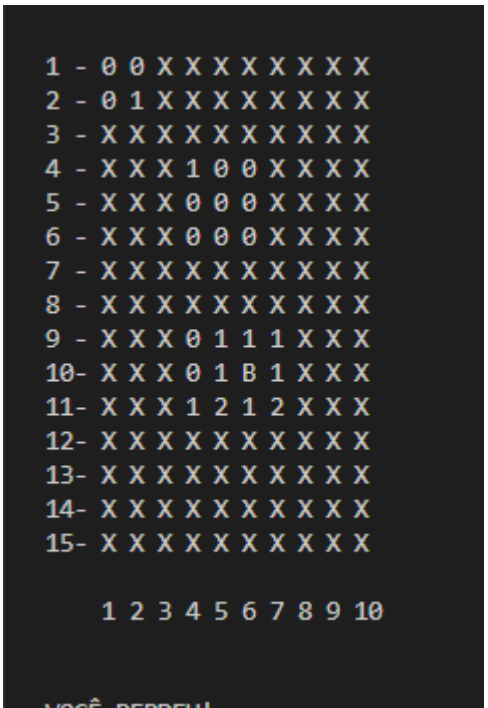
Crie um arquivo de texto na sua máquina, nele insira a quantidade de linhas, colunas e bombas desejadas, lembre-se de adicionar as vírgulas dessa forma, por exemplo: **“4 linhas, 4 colunas, 2 bombas”**. Salve o arquivo em alguma pasta ou crie uma nova pasta, ao executar o código digite o local referente ao arquivo de texto, **“C:\mapasteste\mapa1.txt”**, dessa forma. caso queria mudar os valores do mapa altere no arquivo, salve e rode o código de novo.

Testes realizados:

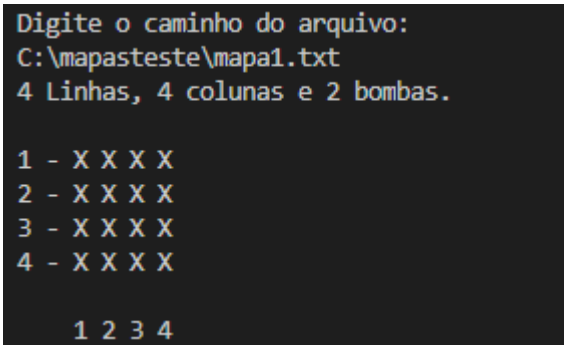
Teste com arquivo grande: “15 linhas, 10 colunas, 10 bombas”



Print correto da quantidade de linhas, colunas e colunas para orientar digitação da coordenada. Após digitar as coordenadas: (5,5), (10,5), (1,1), (10,6) o resultado foi:



Teste com arquivo pequeno: “4 linhas, 4 colunas, 2 bombas”



Após digitar as coordenadas: (1,1), (3,1), (1,3), (3,4) o resultado foi: