	Centro Tecnológico Departamento de Informática	
Disciplina: Programação II		Código: INF09330
Trabalho Prático	Prof. Dr. Thiago Oliveira dos Santos	

Especificação do Trabalho

Campo Minado

1 Introdução

Esse trabalho tem como objetivo avaliar, de forma prática, os conceitos ensinados no curso de Programação II. O trabalho pode ser feito de forma incremental, isto é, à medida que o conteúdo for ensinado. A meta desse trabalho é implementar um jogo de campo minado seguindo as especificações descritas nesse documento.

Campo minado é um popular jogo de computador para um jogador que tem como objetivo revelar um campo de minas sem que alguma seja detonada¹, ver ilustração de uma de suas implementações na Figura 1. Esse jogo segue algumas regras básicas descritas abaixo.

- A área de jogo consiste num campo de quadrados retangular, i.e. posições.
- A cada jogada o jogador escolhe um quadrado (ou seja, uma posição) para ser revelada.
 - Se houver uma mina naquela posição, então o jogo acaba.
 - Se, por outro lado, a posição não contiver uma mina, uma de duas coisas poderá acontecer: um número aparece, indicando a quantidade de quadrados adjacentes que contêm minas (ou seja, o número de vizinhos que são minas); ou, nenhum número aparece e, neste caso, o jogo revela automaticamente os quadrados que se encontram adjacentes ao quadrado vazio, já que não podem conter minas.
- O jogo é ganho quando todos os quadrados que não têm minas são revelados.

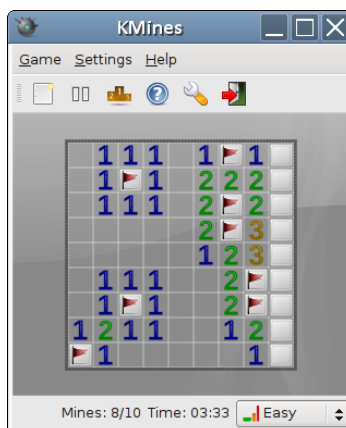



Figura 1: Ilustração do jogo de campo minado para sistemas de janelas¹.

2 Descrição do Trabalho

Nesse trabalho, um programa para jogar campo minado deverá ser implementado. O programa deverá rodar no terminal, assim como todos os outros exercícios feitos em sala. Considere que os jogos podem ser realizados com tabuleiros de configurações variadas (i.e. diferentes tamanhos, diferentes símbolos para representar o estado das posições, e os locais das bombas), e que esses dados (chamados de configuração do tabuleiro) serão passados para o programa antes do jogo iniciar. Percebam, que normalmente alguns desses dados (ex. posição das bombas)

¹ https://pt.wikipedia.org/wiki/Campo_minado

	Centro Tecnológico Departamento de Informática	
Disciplina: Programação II		Código: INF09330
Trabalho Prático	Prof. Dr. Thiago Oliveira dos Santos	

seriam criados aleatoriamente pelo programa, porém serão passados como entrada para possibilitar a correção do trabalho.

De maneira geral, o programa irá iniciar pela linha de comando (terminal) e irá ler um arquivo contendo configurações do tabuleiro. A localização desse arquivo será passada como parâmetro (pela linha de comando) para o programa. Com o tabuleiro iniciado, o programa pedirá o nome do jogador e o jogo iniciará. O jogador fará suas jogadas consecutivamente até que o jogo acabe, ou seja, até que o jogador ganhe ou perca. Antes de cada jogada, o programa deverá pedir para que o jogador digite uma posição (x e y identificando a coluna e a linha respectivamente) a ser revelada e exibirá o tabuleiro com o estado atual do jogo. No final do jogo, o programa deverá exibir o resultado (vencedor ou perdedor) e perguntar se o jogador quer jogar novamente. Em caso afirmativo, o jogo reinicia do zero, em caso negativo o jogo termina. No final de cada jogo, o programa deverá salvar um arquivo contendo as jogadas feitas pelo jogador ordenadas pelo impacto de cada jogada (o impacto será definido mais a frente), para que se possa fazer uma análise de impacto.


2.1 Funcionamento Detalhado do Programa

O programa será chamado de *trabalhoprog2.exe* (após a compilação), portanto o projeto no Netbeans também deverá ser chamado de *trabalhoprog2*. Sua execução será feita através da linha de comando (i.e. pelo cmd, console, ou terminal) e permitirá a passagem de parâmetros, descritos mais adiante. Ao iniciar, o programa deverá executar uma série de operações na sequência, e informar ao usuário o estado atual do jogo. As operações a serem realizadas pelo programa são descritas a seguir (obedecendo a ordem): inicializar jogo, realizar jogo, gerar arquivo de jogadas para análise. A descrição dos parâmetros de inicialização e das operações a serem realizadas pelo programa são apresentadas em detalhes a seguir.

Parâmetros de inicialização (passados na chamada do programa pela linha de comando): Para garantir o correto funcionamento do programa, o usuário deverá informar, ao executar o programa pela linha de comando, o caminho do diretório contendo os arquivos a serem processados (ex. de execução: `“./trabalhoprog2.exe /maquinadorprofessor/test_1”`). O programa deverá verificar a presença do parâmetro. Caso o usuário tenha esquecido de informar o nome do diretório, o programa deverá exibir (ex. "ERRO: O diretório de arquivos de configuração não foi informado!"), e finalizar sua execução.

Inicializar jogo: Nessa operação, o programa deverá ler informações do arquivo de configuração do tabuleiro para a memória e preparar o ambiente de jogo. O programa deverá procurar o arquivo de configurações do jogo, denominado *tabuleiro.txt* e descrito abaixo, no diretório informado pelo usuário durante a chamada do programa, e ler sua informação para memória. Caso o programa não consiga ler o arquivo, ele deverá ser finalizando e imprimir a mensagem informando que não conseguiu ler o arquivo (OBS: a mensagem deve conter o caminho e nome do arquivo que ele tentou ler). Em seguida, o programa deverá pedir o nome do jogador e imprimir o estado inicial do tabuleiro.

- *Organização do arquivo de configuração do tabuleiro (tabuleiro.txt):* A primeira linha do arquivo *tabuleiro.txt* conterá três caracteres sendo o primeiro para representar posições fechadas do tabuleiro (ex. ‘-’), o segundo para representar posições abertas e com bomba (ex. ‘+’) e o terceiro para representar posições abertas e vazias (ex. ‘.’). Esses três caracteres definirão a aparência do tabuleiro impresso no terminal. A linha seguinte informará o número de linhas e colunas do tabuleiro (o tabuleiro será sempre um quadrado e terá linhas e colunas com tamanho de no máximo 1000). As linhas subsequentes informarão o conteúdo de cada posição do tabuleiro, i.e. 1 para bomba e 0

	Centro Tecnológico Departamento de Informática	
Disciplina: Programação II		Código: INF09330
Trabalho Prático	Prof. Dr. Thiago Oliveira dos Santos	


para vazio. Perceba que os vizinhos das bombas deverão ser calculados pelo seu programa.

```
--+.
20 20
10000000001000000001
01000000001000000010
00000000000000000000
00000000000000000000
00001010000001110000
00001010011001010000
00001110000001110000
00000000000000000000
00000000000000000000
00000000000000000000
00000000000100000000
10001001001100010001
00000000000010000000
00001000000000000000
00001000000000000000
00000000000000000000
00000101001001001000
00000000000000000000
00000000000000000000
01000000000100000010
10000000000000000001
```

- Nome do jogador: Ao iniciar o jogo, o programa deverá pedir o nome do jogador com a mensagem "Digite o nome do jogador:". O nome do jogador poderá ter até 1000 caracteres. Após informar o jogador informar seu nome, o programa deverá exibir a mensagem "Digitado (%s)" informando o nome digitado.

Realizar jogo: Uma vez preparado, as jogadas poderão começar e seguir até o fim do jogo. Em cada jogada o jogador informa um posição que deseja revelar (abrir) no tabuleiro. Se houver uma mina naquela posição, então o jogo acaba. Se, por outro lado, a posição não contiver uma mina, uma de duas coisas poderá acontecer: um número aparece, indicando a quantidade de quadrados adjacentes que contêm minas (ou seja, o número de vizinhos que são minas); ou, nenhum número aparece e, neste caso, o jogo revela automaticamente os quadrados que se encontram adjacentes ao quadrado vazio até que um número seja encontrado (DICA: algoritmo de Flood Fill). O jogo terminará quando uma bomba for aberta (nesse caso com derrota) ou quando o jogador abrir todas as posições sem bomba (nesse caso com vitória). As jogadas deverão ser lidas da entrada padrão de dados, permitindo dessa forma, um redirecionamento a partir de arquivo (igual fazemos no BOCA). O resultado final impresso na tela poderá ser redirecionado para arquivo para permitir comparações com os arquivos fornecidos como exemplo, *jogadasOut.txt* (igual fazemos no BOCA). O fluxo de jogo é descrito abaixo:

- Jogada: Antes de cada jogada, o programa deverá informar o nome do jogador e o número da jogada corrente com a mensagem "%s Jogada: %d"; e deverá pedir, com a mensagem "Digite a posicao (x e y):", a posição (x e y) a ser revelada. O formato esperado é de dois números inteiros (indo de zero até o tamanho do tabuleiro menos um) representando a linha e a coluna da posição desejada, respectivamente, separados por espaço. Em seguida, o programa deverá informar a posição digitada com a seguinte mensagem "Digitado (%d,%d)". Se a posição digitada for inválida (i.e. se já estiver aberta ou estiver fora do tabuleiro), o programa deverá informar que ela é inválida com as seguintes mensagens, "Posicao invalida (JA ABERTA)!" se ela já estiver aberta e "Posicao invalida (FORA DO TABULEIRO)!" se ela estiver fora do tabuleiro. Uma jogada inválida não conta para incrementar o número de jogadas, portanto as jogadas só avançam quando uma jogada válida é fornecida.

	Centro Tecnológico Departamento de Informática	
Disciplina: Programação II		Código: INF09330
Trabalho Prático	Prof. Dr. Thiago Oliveira dos Santos	

- Tabuleiro: O tabuleiro deverá ser impresso a cada jogada. Cada posição do tabuleiro possui quatro estados possíveis (fechado; aberto e vazio; aberto e com bomba; e aberto e vizinho de bomba). O estado *aberto e vizinho de bomba* será mostrado com um número representando a quantidade de bombas ao seu redor. Os outros estados serão mostrados com símbolos dados juntamente com a configuração do tabuleiro (*tabuleiro.txt*). O tabuleiro deverá ser impresso conforme exemplificado na **Error! Reference source not found.** Perceba que os números identificando as linhas e colunas são impressos com até três zeros a esquerda e isso permite uma padronização da impressão. Cada posição é separada por um espaço e é impressa com 3 caracteres para alinhar com os números das colunas (as posições fechadas repetem o seu caractere representante 3 vezes e as abertas colocam seu caractere representante entre dois espaços).

```


- 000 001 002 003 004 005 006 007 008 009 010 011 012 013 014 015 016 017 018 019
000 + 2 1 . . . . . 2 --- 2 . . . . . 1 --- ---
001 2 --- 1 . . . . . 2 --- 2 . . . . . 1 --- ---
002 1 1 1 . . . . . 1 1 1 . . . . . 1 1 1
003 . . . 1 1 2 1 1 . . . 1 2 3 2 1 . . .
004 . . . 2 --- 4 --- 2 1 2 2 1 2 --- --- ---
005 . . . 3 --- 7 --- 3 --- --- --- --- ---
006 . . . 2 --- --- 2 1 2 2 1 2 --- --- ---
007 . . . 1 2 3 2 1 . . . 1 2 3 2 1 . . .
008 . . . . . . . . . 1 1 1 . . . . .
009 1 1 . 1 1 1 1 1 1 1 3 --- 2 . 1 1 1 . 1 1
010 --- 1 . 1 --- --- --- --- --- 3 1 1 --- 1 . 1 ---
011 1 1 . 2 --- 2 1 1 1 1 2 3 --- 1 1 1 1 . 1 1
012 . . . 2 --- 2 . . . . . 1 1 1 . . . . .
013 . . . 2 --- 2 . . . . . . . . . . . .
014 . . . 1 2 2 2 1 1 1 1 1 1 1 1 1 1 1 . .
015 . . . . 1 --- 2 --- 1 --- 1 1 --- 1 --- 1 . .
016 . . . . 1 1 2 1 1 1 1 1 1 1 1 1 1 1 . .
017 1 1 1 . . . . . . . 1 --- 1 . . . . 1 1 1
018 2 --- 1 . . . . . . 1 --- 1 . . . . 1 --- 2
019 --- 2 1 . . . . . . 1 --- 1 . . . . 1 2 ---

```

Figura 2: Ilustração do tabuleiro a ser impresso usando '-' para posição fechada, '+' para posição aberta e com bomba (ex. 000, 000), '.' para posição aberta e vazia, e números para posição aberta e vizinha de bomba.

- Resultado: Ao finalizar, cada jogo deverá imprimir o resultado informando o nome do jogador e o resultado com uma das seguintes mensagens "%s Venceu!" se o jogador venceu e "%s Perdeu!" se o jogador perdeu.
- Jogar novamente: Após o final de cada jogo, o programa deverá perguntar se o jogador deseja jogar novamente com a mensagem "Jogar novamente? (s,n)". Ele responderá 's' caso queira, e 'n' caso contrário. Em seguida, o programa deverá informar a opção digitada com a seguinte mensagem "Digitado (s)" para sim e "Digitado (n)" para não.

Gerar arquivo de jogadas para análise: Nessa operação, o programa deverá utilizar as informações das jogadas coletadas durante a partida e exportar para um arquivo denominado "*analiseJogo##.txt*" em que ## representa o número do jogo daquele arquivo. Por exemplo, *analiseJogo01.txt* representa o arquivo de análise do primeiro jogo desde quando o programa foi executado, *analiseJogo02.txt* o segundo e assim por diante. Esse arquivo deverá conter uma lista de todas as jogadas válidas daquele jogo, sendo uma jogada por linha e ordenadas. A ordenação será da jogada de maior impacto para a de menor impacto e no caso de empate a jogada que ocorreu primeiro terá prioridade. O impacto é medido pelo número de posições que aquela jogada abriu. Por exemplo, uma jogada que abriu bomba terá impacto 0; uma jogada que abriu um número terá impacto 1; uma jogada que abriu um vazio e consequentemente seus vizinhos terá impacto *n*, sendo *n* o número de posições abertas com ela. Se houver falha na operação, o programa deverá imprimir "ERRO: Nao foi possível salvar arquivo para analise."

	Centro Tecnológico Departamento de Informática	
Disciplina: Programação II		Código: INF09330
Trabalho Prático	Prof. Dr. Thiago Oliveira dos Santos	

Função bônus (opcional): Essa função permitirá que o programa seja executado com um tabuleiro aleatório. Para executar essa função, o usuário deverá informar um segundo e um terceiro parâmetros na linha de comando, isto é, o tamanho t do tabuleiro a ser gerado (ex. 20, 10, 30, etc.) e o número b de bombas que ele deve ter. Caso esses parâmetros extras estejam presentes na linha de chamada do programa, ao invés de executar diretamente as funções descritas acima, o programa deverá primeiro colocar t bombas no tabuleiro de tamanho $t \times t$ e salvar esse tabuleiro no seu respectivo arquivo no diretório informado para que ele possa ser lido pelo programa posteriormente. Após salvar o arquivo, o jogo pode continuar com o tabuleiro criado. O jogo deve usar os caracteres ('-', '+' e '.') como configuração das posições do tabuleiro aleatório. É necessário salvar o arquivo tabuleiro para que seja atestada o correto funcionamento da função bônus.

Alguns arquivos de entrada e seus respectivos arquivos de saída serão fornecido para o aluno como exemplo. O aluno deverá utilizar tais arquivos para testes durante a implementação do trabalho. É responsabilidade do aluno criar novos arquivos para testar outras possibilidades do programa e garantir seu correto funcionamento. O trabalho será corrigido usando outros arquivos (específicos para a correção e não disponibilizados para os alunos) seguindo a formatação descrita nesse documento e utilizada nos arquivos exemplos. Em caso de dúvida, pergunte ao professor. O uso de arquivos com formatação diferente poderá acarretar em incompatibilidade durante a correção do trabalho e consequentemente na impossibilidade de correção do mesmo (nota zero). Portanto, siga estritamente o formato estabelecido.

2.2 Implementação

A implementação deverá seguir os conceitos de modularização e abstração apresentados em sala. O trabalho terá uma componente subjetiva que será avaliada pelo professor para verificar o grau de uso dos conceitos ensinados. Portanto, além de funcionar, o código deverá estar bem escrito para que o aluno obtenha nota máxima.


É extremamente recomendado utilizar algum programa para fazer as comparações do resultado final do programa, isto é, os arquivos de saída gerados, poderão ser comparados com o arquivo de saída esperada (fornecido pelo professor) utilizando o comando *diff*, como visto em sala. O *meld* é uma alternativa gráfica para o *diff*, se você preferir. Esse programa faz uma comparação linha a linha do conteúdo de 2 arquivos. Diferenças na formatação poderão impossibilitar a correção.

3 Regras Gerais

O trabalho deverá ser feito individualmente e pelo próprio aluno, isto é, o aluno deverá necessariamente conhecer e dominar todos os trechos de código criados. O aluno deverá necessariamente utilizar o Netbeans para desenvolvimento do trabalho!

Cada aluno deverá trabalhar independente dos outros não sendo permitido a cópia ou compartilhamento de código. O professor irá fazer verificação automática de plágio. Trabalhos identificados como iguais, em termos de programação, serão penalizados com a nota zero. Isso também inclui a pessoa que forneceu o trabalho, sendo portanto, de sua obrigação a proteção de seu trabalho contra cópias ilícitas. **Proteja seu trabalho e não esqueça cópias do seu código nas máquinas de uso comum (ex. laboratório).**

3.1 Entrega do Trabalho

	Centro Tecnológico Departamento de Informática	
Disciplina: Programação II		Código: INF09330
Trabalho Prático	Prof. Dr. Thiago Oliveira dos Santos	

O trabalho deverá ser entregue por email (para: todosantos@inf.ufes.br) até o dia **19/06/2016**. O assunto da mensagem deverá seguir o padrão: [TRABALHO_PROG2_2016_1] <nome_do_aluno>.

Exemplo do assunto da mensagem para o aluno Fulano da Silva:

- [TRABALHO_PROG2_2016_1] FulanoDaSilva

Todos os arquivos do programa (*.c e *.h), incluindo o diretório “nbproject” e o arquivo “Makefile” gerados pelo netbeans deverão ser compactados em um único arquivo <nome_do_aluno>.zip e enviado como anexo da mensagem. Lembrando que o anexo não deverá conter arquivos compilados “.o” ou executáveis “.exe” sob o risco de bloqueio de email contendo arquivos executáveis. O código será compilado posteriormente em uma outra máquina. A correção poderá ser feita de forma automática, portanto a entrega incorreta do trabalho, ou seja, fora do padrão, poderá acarretar na impossibilidade de correção do trabalho e consequentemente na atribuição de nota zero. A pessoa corrigindo não terá a obrigação de adivinhar nome de arquivos, diretórios ou outros. Siga estritamente o padrão estabelecido!

Dica para teste de envio do trabalho: siga os passos anteriores; envie um email para você mesmo; descompacte o conteúdo para uma pasta; abra um terminal e mude o diretório corrente para a pasta de descompactação; execute o comando “make all” que deverá compilar todo o projeto e gerar um arquivo executável em algum lugar da pasta “./dist/Release/.../trabalhprog2.exe”; coloque os arquivos de teste em um diretório qualquer (ex. “/minhapasta/test_1”); execute o programa gerado passando o caminho do diretório com os arquivos de teste, e veja se funciona corretamente. Por fim, abra o projeto com o Netbeans para ver se está tudo como esperado. **Se não funcionou para você, não vai funcionar para mim!**


Os dados necessários para executar o comando “make” são gerados automaticamente pelo Netbeans dentro da pasta do projeto.

3.2 Pontuação

Trabalhos entregues após o prazo não serão corrigidos (nota zero). O trabalho será pontuado de acordo com sua implementação e a tabela abaixo. Os pontos descritos na tabela não são independentes entre si, isto é, alguns itens são pré-requisitos para obtenção da pontuação dos outros. Por exemplo, gerar análise depende de jogar. Código com falta de legibilidade e modularização pode perder ponto conforme informado na tabela. Erros gerais de funcionamento, lógica ou outros serão descontados como um todo.

Percebam que no melhor dos casos os pontos da tabela abaixo somam 11 ao invés de 10. Isso foi feito propositalmente para ajudar os alunos esforçados com um ponto extra. Esse ponto, caso obtido, irá complementar uma das notas, do trabalho ou das provas parciais do semestre. Prioridade será dada para a nota com maior peso, porém não ultrapassando a nota máxima.

Item	Quesitos	Ponto
Jogar	Inicializar o jogo	8
	Realizar o jogo	
	Gerar resultados corretamente	
Gerar análise	Gerar corretamente arquivo de jogadas para análise	2

	Centro Tecnológico Departamento de Informática	
Disciplina: Programação II		Código: INF09330
Trabalho Prático	Prof. Dr. Thiago Oliveira dos Santos	

Legibilidade Modularização	e Uso de comentários Identação do código Uso de funções Uso de tipos de dados definidos pelo usuário e respectivos arquivos (.c e .h)	-2
Função bônus	Geração de tabuleiro aleatório	1

Com intuito de identificar alunos que tiveram o trabalho feito por terceiros, a nota obtida com a avaliação acima (NI) será ponderada com uma nota de entrevista (NE) sobre o trabalho. A nota final do trabalho será calculada com $NI \times NE$, onde NE vai de zero a um. A entrevista avaliará o conhecimento do aluno a respeito do programa desenvolvido. A nota da entrevista não servirá para aumentar a nota do trabalho, mas sim para diminuir quando for identificada uma falta de conhecimento sobre o trabalho entregue. A necessidade da entrevista será avaliada no decorrer do semestre, portanto sejam honestos e façam o trabalho com seu próprio esforço.

4 Considerações Finais

Não utilizar acentos no trabalho.

5 Erratas

Esse documento descreve de maneira geral as regras de implementação do trabalho. É de responsabilidade do aluno garantir que o programa funcione de maneira correta e amigável com o usuário. Qualquer alteração nas regras do trabalho será comunicada em sala e no portal do aluno. É responsabilidade do aluno frequentar as aulas e se manter atualizado em relação as especificações do trabalho. Caso seja notada qualquer tipo de inconsistência nos arquivos de testes disponibilizados, comunique imediatamente ao professor para que ela seja corrigida.