UNIVERSIDADE DE ÉVORA

Escola de Ciências e Tecnologia

Licenciatura de Engenharia Informática

Programação 1

Ano Letivo 2020/2021

Docentes:

- Miguel Barão
- Teresa Gonçalves
- Vítor Nogueira

Discentes:

- João Rodrigo Condeço nº48976
- Pedro Leitão nº48602

Índice

Introdução

Este trabalho prático tem como objetivo conciliar os conhecimentos adquiridos na cadeira de Programação 1 para desenvolver o jogo Minesweeper na linguagem de programação C.
Este jogo é constituido por um tabuleiro no qual se encontram escondidas minas. O objetivo é o jogador ir selecionando coordenadas sem acertar nas minas e, através da função flag, assinalar os locais onde acha que se encontram as mesmas. Uma vez destapadas todas as casas que não são minas o jogo termina e o jogador vence, caso o jogador acerte numa mina, perde.
Para auxilio do utilizador o jogo informa-o de quantas minas existem no tabuleiro e quantas bandeiras lhe falta colocar.
Pode-se assim verificar que o jogador apenas tem de se preocupar com 3 regras:

- Não destapar nenhuma mina;
- Assinalar com uma bandeira todas e apenas as coordenadas das minas;
- Destapar todos os espaços que não sejam minas.

Software utilizado:

- Apple Xcode (compilador);

Descrição do Programa

Menu Principal

Quando entra no jogo o utilizador é confrontado com um menu onde terá de escolher entre as seguintes opções:

- **Opção 0 (Sair)** -> Quando selecionada esta opção é mostrada uma mensagem de despedida e o programa termina.
- **Opção 1 (Tutorial)** -> Quando esta opção é escolhida o utilizador irá ver uma mensagem onde as instruções do jogo são mostradas.
- **Opção 2 (Gerar Tabuleiro)** -> Esta opção irá levar o utilizador a efetuar os passos necessários a iniciar o jogo com um tabuleiro personalizado por este. Sendo estes passos recolha de dados (número de linhas, colunas e minas). Após recolhidos os dados necessários o utilizador pode iniciar o jogo e no fim deste regressa ao Menu Principal.
- **Opção 3 (Ler Tabuleiro)** -> Nesta opção o utilizador terá de indicar um ficheiro onde se encontram os dados necessários à construção do tabuleiro onde irá decorrer o jogo. Caso o ficheiro seja validado pelo programa o jogo irá começar e após este o jogador regressa ao Menu Principal.

Assim podemos verificar que o programa anda sempre em torno do Menu principal, a única opção que não volta ao menu é a que termina o programa.

Menu Principal

Gerar Tabuleiro:

Caso no menu seja selecionado a opção gerar tabuleiro o utilizador irá ser questionado acerca das características do tabuleiro (formato linhas, colunas, minas) através da função data( ) e armazena na estrutura estes valores. Se os valores inseridos não tiverem dentro das características pedidas é mostrada uma mensagem de erro e o user é forçado a tentar novamente até inserir valores válidos.

De seguida a função generate_board gera aleatoriamente as coordenadas das minas e armazena-as nas nested structs criadas para esse propósito. Inicialmente devido a uma confusão na interpretação inicial do enunciado, nesta fase do processo a função generate_board registava num ficheiro os dados no entanto como no enunciado não mostrava necessidade de tal, foi tomada a decisão de retirar esta ação.

```
Insira, separado por espaços, o número de linhas, colunas e minas.
Por exemplo: 10 15 5
Dados: 10 10 10


 ---------------------------------------------------
| Número de minas: 10   Número de bandeiras restantes: 10 |
 ---------------------------------------------------

   A B C D E F G H I J
 1 . . . . . . . . . .
 2 . . . . . . . . . .
 3 . . . . . . . . . .
 4 . . . . . . . . . .
 5 . . . . . . . . . .
 6 . . . . . . . . . .
 7 . . . . . . . . . .
 8 . . . . . . . . . .
 9 . . . . . . . . . .
10 . . . . . . . . . .
```

Recolha de Dados

Ler Tabuleiro:

Se for selecionada a opção Ler Tabuleiro na fase inicial do programa, será solicitada a localização de um ficheiro onde estejam as informações necessárias à criação de um tabuleiro e no formato correto (formato exemplificado na imagem 1). O programa testa a existência e validade do ficheiro indicado e caso tudo esteja dentro do necessário as informações são recolhidas e armazenadas na estrutura.

```
número_de_linhas número_de_colunas número_de_minas
linha_da_mina coluna_da_mina
```

Formato do ficheiro

```
10 10 10
3 A
1 F
2 B
3 G
8 E
8 G
7 A
7 G
9 F
9 H
```

Imagem 1

Jogo:

Após os processos necessários para gerar o tabuleiro, quer pelo processo Gerar Tabuleiro quer pelo processo Ler Tabuleiro, podemos iniciar o jogo. Irá ser mostrado o tabuleiro ao utilizador e ser-lhe-á pedida uma jogada.

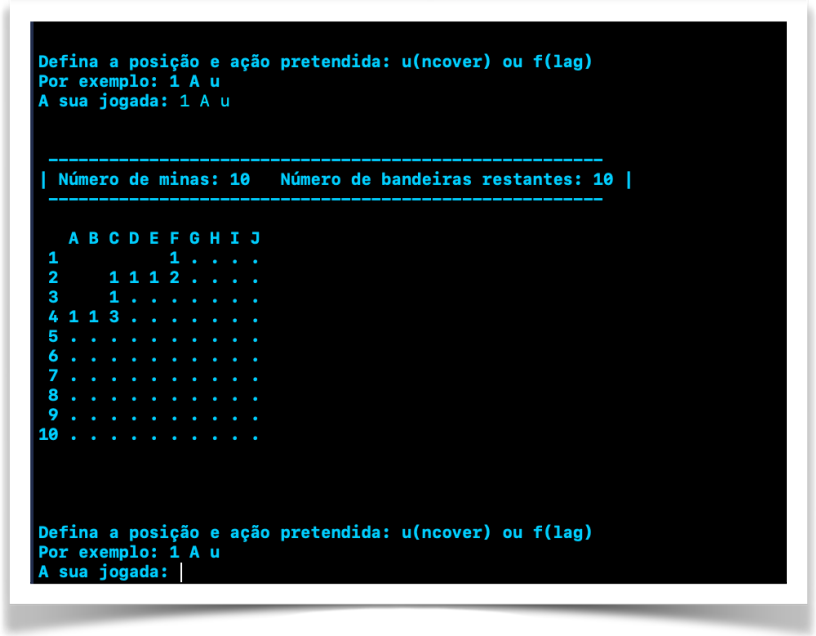
Nesta situação as opções são:

- Destapar uma coordenada;
- Assinalar uma mina;

Quando recolhida a informação relativa à jogada, se for válida, os dados recolhidos são convertidos para a letra maiúscula e minúscula (conforme a necessidade) através da função toupper e tolower. Caso esta jogada seja válida o programa analisa a situação da seguinte forma:

- Caso a coordenada seja uma mina e a ação escolhida seja destapar a coordenada, é mostrada a mensagem de derrota e o jogador pode ver o tabuleiro com as minas à mostra.
- Caso contrário é calculado o valor da coordenada com base no número de minas que a rodeiam e se este for zero irá expandir a seleção aos espaços à volta do espaço selecionado de forma a revelar uma área livre de minas. Se o valor for maior que zero apenas será mostrado o número correspondente a essa casa.
- Se a ação selecionada for para assinalar uma mina, a coordenada selecionada é marcada com uma bandeira. Se a coordenada em causa já conter uma bandeira, deixará de a ter.

Assim podemos concluir que o jogo irá acabar se o jogador destapar todas as coordenadas sem minas ou se acertar numa mina.



Exemplo de Jogada

Descrição de Funções:

Ficheiro minesweeper.c:

Nome da função: wish

Declaração da mesma: int wish( )

Funcionalidade: Esta função foi criada para, sempre que solicitada, perguntar ao utilizador se deseja jogar novamente. O valor de retorno desta é usado na condição do ciclo while no menu da função main. Assim esta função é usada no fim dos dois métodos de jogo (ler tabuleiro ou gerar tabuleiro).

Valor de retorno: É devolvido o valor correspondente à intenção do jogador continuar ou sair do jogo.

Ficheiro mine_functions.c:

Nome da função: data

Declaração da mesma: void data( )

Funcionalidade: Esta função é utilizada a fim de recolher as informações necessárias para gerar o tabuleiro, sendo estas o número de linhas, número de colunas e número de minas (devem ser inseridos por esta ordem). Após recolhidas as informações a função determina se:

- As informações inseridas estão no formato correto;
- Se o número de linhas, colunas e minas se encontram dentro dos limites;
- Se o número de minas é maior do que o mínimo calculado pela função.

Caso alguma destas condições não se verifique, o programa indica que algo está errado e repete a recolha de informação.

Para o jogo não ser demasiado fácil foi implementada uma fórmula que calcula um número de mínimo minas inserido com base nas colunas e linhas.

Após a recolha da informação, a mesma é adicionada à estrutura e é executada a função generate_board.

Nome da função: generate_board

Declaração da mesma: void generate_board(struct board d; int nlines; int ncols; int nmines)

Funcionalidade: Esta função através de um ciclo for, que funciona enquanto a variável i for menor que o número de minas, gera coordenadas de minas aleatoriamente com recurso à função rand. É verificado se as coordenadas já existem na estrutura através da função existe_no_array e se não existirem são adicionadas à mesma, caso existam não são adicionadas e a variável i vai ser igual a i - 1 para repetir o ciclo novamente e assim evitar minas repetidas. No fim desta função é executada a função game onde irá decorrer o jogo.

Nome da função: existe_no_array

Declaração da mesma: int existe_no_array(int arr[ ], int arr2[ ], int tested, int tested2)

Funcionalidade: Esta função não constava no enunciado no entanto notou-se a necessidade de criar uma função que verifica-se se uma determinada coordenada se encontrava na estrutura. Esta função necessita do array com a coordenada X (arr[ ]), do array com a coordenada Y (arr2[ ]) e testa respectivamente se o inteiro tested e o inteiro tested2 se encontram neles. Se se verificar que existem as coordenadas na estrutura a função retorna 1, caso contrário retorna 0.

Valor de retorno: Retorna o valor referente à existência ou não da coordenada no array.

Nome da função: game

Declaração da mesma: void game(struct board d, int nlines, int ncols, int showmines)

Funcionalidade: É nesta função que decorre o jogo em si, através de um ciclo while, enquanto nenhuma mina for destapada e ainda existirem espaços (que não sejam minas) tapados, o jogo corre da seguinte forma:

- É mostrado tabuleiro através da função print_board juntamente com o número de bandeiras disponíveis e o número de minas.
- É pedida uma jogada (uncover ou flag) com as respetivas coordenadas.
- O jogo avalia a validade da jogada e ação pretendida.
- Executa a jogada (se for válida).
- Repete tudo mas com o tabuleiro atualizado com as jogadas.

Se a opção for uncover irá ser utilizada a função uncover para realizar esta operação e de seguida a função on_off confirma se ainda existe algum espaço por selecionar, se for flag é invocada a função flag.

Quando quebrado o ciclo while, a variável showmines passa a 1 e assim o tabuleiro é mostrado mas com as minas visíveis desta vez. Com base na função its_on é mostrada a mensagem de fim de jogo (game over ou parabéns).

Para se evitar erros todos os inputs são convertidos a letra maiúscula ou minúscula conforme a necessidade.

Nome da função: print_board

Declaração da mesma: void print_board(struct board d, int nlines, int ncols, int showmines)

Funcionalidade: Esta função, sempre que solicitada, mostra ao utilizador o tabuleiro (com ou sem minas, dependendo da variável showmines). A função através de dois ciclos for imprime linha a linha o tabuleiro, primeiro verifica se é mina (e se a variável showmines é igual a 1), de seguida verifica se é bandeira e por fim verifica se é espaço selecionado. Se for mina imprime um “*”, bandeira imprime um “f” e espaço selecionado varia consoante o número de minas que o rodeiam (calculado pela função number).

Se nenhuma das condições se verificar é colocado um “.” na coordenada em questão.

Nome da função: uncover

Declaração da mesma: struct board uncover(struct board d, int x, int y, int counter)

Funcionalidade: Esta função recursiva testa todas as coordenadas à volta da coordenada em causa e assim sucessivamente o mesmo se repete para as casas que são adicionadas às jogadas automaticamente por esta função. Tirando a primeira coordenada (daí o uso da variável counter) todas as coordenadas passam pela verificação de que não são minas nem foram já adicionadas à estrutura e nem excedem os limites do tabuleiro. Cada vez que uma função passa pelas condições anteriormente mencionadas o número de jogadas aumenta uma unidade e é adicionada a coordenada em questão à estrutura.

Valor de retorno: Devolve a estrutura com os valores atualizados.

Nome da função: flag

Declaração da mesma: struct board flag(struct board d, int x, int y)

Funcionalidade: Esta função é executada para adicionar/remover as bandeiras da estrutura quando o jogador opta pela ação flag. Com auxilio da função existe_no_array é verificado se as coordenadas inseridas já existem na estrutura, caso existam são eliminadas, caso contrário são adicionadas.

Valor de retorno: Devolve a estrutura com os valores atualizados.

Nome da função: print_chars

Declaração da mesma: void print_chars(int n, char c)

Funcionalidade: A função print_chars imprime o caracter desejado (variável c) as vezes indicadas (variável n). Apenas serve para auxiliar nas mensagens para o utilizador.

(I-1,J-1)	(I-1, J)	(I-1, J+1)
(I, J-1)	(I , J)	(I, J+1)
(I+1, J-1)	(I+1, J)	(I+1,J+1)

Modo de funcionamento da função number

Nome da função: number

Declaração da mesma: int number(struct board d, int i, int j)

Funcionalidade: Quando invocada a função number usa dois ciclos for para percorrer as 8 posições que rodeiam a coordenada (i , j), para isto usa a função existe_no_array para verificar a existência de minas, se existirem é adicionado 1 à variável n.

Valor de retorno: Número de minas que rodeia a coordenada

Nome da função: acha_minas

Declaração da mesma: int acha_minas(struct board d, int row, int column)

Funcionalidade: A função acha_minas foi criada para nos dar o número de minas quando a função uncover percorre o tabuleiro. Funciona de forma semelhante à função number.

Valor de retorno: Retorna o número de minas que rodeia a coordenada em causa

Nome da função: on_off

Declaração da mesma: int on_off(struct board d)

Funcionalidade: Esta função percorre todas as coordenadas do tabuleiro para verificar se existe alguma que não seja mina nem selecionada. Caso todas as coordenadas estejam selecionadas ou sejam minas é retornado o valor 0 e o ciclo da função game irá ser interrompido. Caso alguma coordenada esteja por destapar a variável dory toma o valor 1 e o ciclo continua.

Valor de retorno: variável dory com o valor referente ao fim do jogo ou não.

Nome da função: game_ended

Declaração da mesma: void game_ended(int its_on)

Funcionalidade: Esta função com base no valor da variável its_on mostra a mensagem de vitória (se for maior que 0) ou de derrota (se for 0).

Nome da função: receiver

Declaração da mesma: struct board receiver( )

Funcionalidade: Esta função pede ao utilizador a localização de um ficheiro no formato correto com as informações relativas ao tabuleiro de jogo. Se o programa não conseguir aceder ao ficheiro irá mostrar uma mensagem de erro, caso contrário irá proceder à recolha de informação e adicionar os dados à estrutura.

Valor de retorno: estrutura d com os valores referentes ao tabuleiro de jogo.

Nome da função: artificial_round

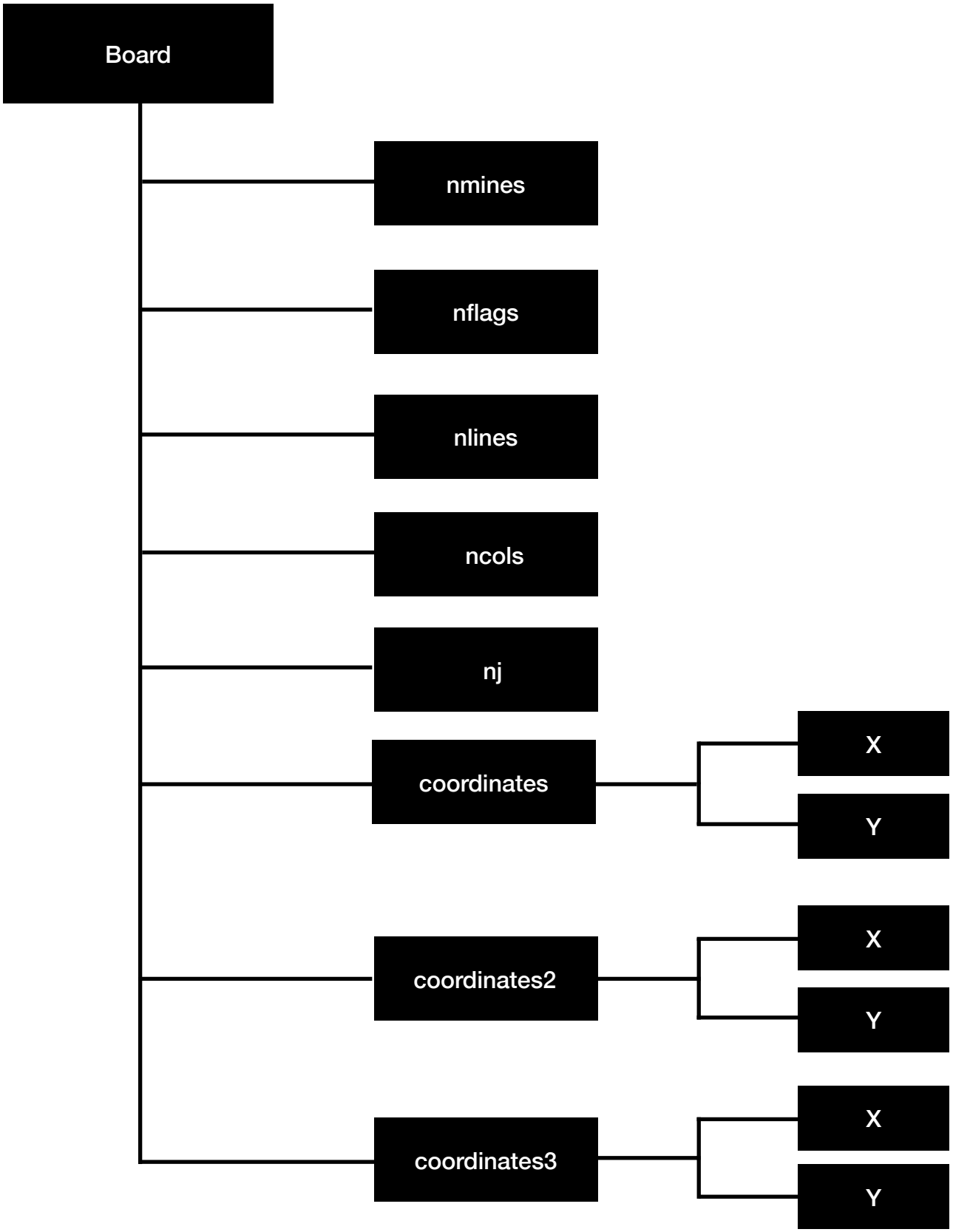
Declaração da mesma: int artificial_round(float num)

Funcionalidade: Esta função foi criada para desempenhar a tarefa da função já existente round. No entanto foi tomada a decisão de criar uma versão para este trabalho.

Valor de retorno: valor mínimo de minas que o tabuleiro tem de ter.

Estrutura

Para armazenar os dados relativos à estrutura foi criada uma nested struct com o seguinte formato:



Na estrutura board identificada no esquema acima são armazenadas:

- O número de minas em nmines;
- O número de flags em nflags;
- O número de linhas em nlines;
- O número de colunas em ncols;
- O número de jogadas em nj;
- As coordenadas das minas em coordinates;
- As coordenadas das flags em coordinates2;
- As coordenadas dos espaços selecionados em coordinates3.

Em cada uma das estruturas coordinates é assim armazenado os valores X e Y das coordenadas de cada tipo de elemento na mesma posição do array.

Nota

Se pretender aceder à primeira jogada efetuada e o nome atribuído à estrutura for d basta ver o valor das seguintes variáveis:
(x, y) <=> (d.coordinates3.x[0] , d.coordinates.y[0])

Inicialmente tinha sido pensado utilizar-se um array bidimensional para representar o tabuleiro no entanto após uma dúvida ser esclarecida pelo professor Barão foi decidida a implementação da estrutura da forma explicada anteriormente.

Complicações

- Inicialmente existiu dificuldades em gerar o número aleatoriamente através da função rand pois não havia conhecimento da necessidade de também usar a função srand no entanto após alguma pesquisa esse erro foi corrigido.
- A função uncover apresentava um erro pois ficava infinitamente a executar a recursividade. Após um ajuste das condições das estruturas de decisão foi possível corrigir esse erro.
- Inicialmente foi usada a função fgets para a leitura do ficheiro no entanto verificou-se uma maior eficácia com a função fscanf.
- Inicialmente o programa registava num ficheiro as informações do tabuleiro usado no entanto não se verificou utilidade na fase atual do projeto, talvez numa fase mais avançada implementássemos a opção do utilizador inserir o caminho para registar num ficheiro à sua escolha o tabuleiro gerado e aí sim valesse a pena o registo da informação.