Universidade Federal de Ouro Preto – UFOP
Departamento de Computação e Sistemas – DECSI
Disciplina: Algoritmos e Estruturas de Dados – CSI 428/488

Trabalho Prático 1:
Tipos Abstratos de Dados e Alocação Dinâmica de Memória

Valor: 0,75 pontos (7,5% da nota total) **Data de entrega: 07/04/2019**
Impressão frente-verso: +0.05 ponto (ecologicamente correto)

Objetivos

O objetivo desse trabalho é rever conceitos básicos de programação bem como explorar os conceitos de Tipos Abstratos de Dados (TADs) e análise de complexidade.

Descrição

Você já jogou Campo Minado? É um jogo adorável que vem com um certo Sistema Operacional cujo nome nós não conseguimos lembrar. Bom, o objetivo do jogo é encontrar onde estão todas as minas em um campo n x m. Para ajudar o jogador, o jogo mostra um número em um quadrado que lhe diz quantas minas existem adjacentes aquela posição. Por exemplo, suponha o campo 4 x 4 a seguir com 2 minas (que são representadas pelo caractere '`*`'):

```
*...
....
.*..
....
```

Se quisermos representar o mesmo campo posicionando os números-dica como descrito acima, então obteremos:

```
*100
2210
1*10
1110
```

Como você já deve ter reparado, cada quadrado pode possuir no máximo 8 quadrados adjacentes.

Entrada

A entrada consiste de um número arbitrário de campos. A primeira linha de cada campo contêm dois inteiros n e m ($0 < n, m \leq 100$) que significa o número de linhas e colunas do campo, respectivamente. As próximas n linhas contêm exatamente m caracteres e representam o campo.

Cada quadrado seguro é representado por um caractere '`.`'(sem as aspas) e cada quadrado contendo uma mina é representado por um caractere '`*`'(também sem as aspas). A primeira linha de campo onde n = m = 0 representa o fim da entrada e não deve ser processada.

Saída

Para cada campo, você deverá imprimir a seguinte mensagem em uma linha:

```
Field #x:
```

Onde x é o número do campo (começando de 1). As próximas n linhas devem conter o campo com os caracteres `'.'` substituídos pelo número de minas que são adjacentes aquela posição. Deve existir uma linha vazia entre cada campo

Exemplo de Entrada:

```
4 4
*...
....
.*..
....
3 5
**...
.....
.*...
0 0
```

Exemplo de Saída:

```
Field #1:
*100
2210
1*10
1110

Field #2:
**100
33200
1*100
```

Tpo Abstrato de Dados (TAD)

Você deverá implementar um tipo abstrato de dados **`TQuadrado`** para representar um quadrado do campo. Seu tipo abstrato deverá armazenar um caractere para representar uma bomba (`'*'`) ou um espaço seguro (`'.'`) e um campo lógico para visita (visitado ou não).

Você deverá implementar também um tipo abstrato de dados **`TCampo`** para represetar um campo do jogo Campo Minado. Seu tipo abstrato deverá armazenar os quadrados do campo e o número de linhas e colunas n e m. Considere que o tamanho máximo de um campo é limitado pela memória principal (você deverá usar alocação dinâmica). Use arranjos bidimensionais (matrizes) para implementar o TAD.

As operações que devem ser realizadas em seu TAD são:

1. Inicializar um campo vazio (já com n x m quadrados seguros).

   ```
   void inicializar(TCampo* pA);
   ```

2. Inserir uma bomba no campo, retornando 1 se possível e 0 se não.

   ```
   int inserir_bomba(TCampo* pA, int posx, int posy);
   ```

3. Excluir uma bomba do campo, retornando 1 se realizado e 0 se não.

   ```
   int excluir_bomba(TCampo* pA, int posx, int posy);
   ```

4. Calcular número de bombas adjacentes, retornando o número de bombas adjacentes ao quadrado em questão.

   ```
   int bombas_adjacentes(TCampo* pA, int posx, int posy).
   ```

5. Visitar um quadrado do campo, retornando 1 caso o quadrado seja segura e 0 caso uma bomba seja encontrada.

   ```
   int visitar_quadrado(TCampo* pA, int posx, int posy);
   ```

6. Gerar um campo a partir dos dados lidos pela entrada, como descrito anteriormente.

   ```
   TCampo carregar_campo(int n, int m);
   ```

7. Gerar um campo aleatoriamente com n x m quadrados e k bombas.

   ```
   TCampo gerar_campo_kbombas(int m, int n, int k);
   ```

8. Gerar um campo aleatoriamente com n x m quadrados e uma fração f ($0 \leq f \leq 1$) de bombas.

```
TCampo gerar_campo_fbombas(int m, int n, float f);
```

9. Imprimir os elementos do campo.

   ```
   void imprimir(TCampo A);
   ```

10. Solucionar o campo minado substituindo todos os quadrados seguros pela contagem de bombas adjacentes, como definido anteriormente.

    ```
    void solucionar_campo(TCampo A);
    ```

Implemente o seu TAD em arquivos separados do programa principal (`TCampo.h` e `TCampo.c`). Se necessário, você pode criar outras funções auxiliares em seu TAD. Suas funções devem executar testes de consistência (por exemplo, não se pode inserir uma bomba em um quadrado cujas posições extrapolam o tamanho definido para o campo).

Uma vez criado o seu TAD, você deverá utilizá-lo em dois programas diferentes. O primeiro é o programa que recebe um campo (pela entrada padrão) e imprime (utilizando a saída padrão) o campo correspondente trocando os quadrados seguros pelo número de bombas adjacentes (como mostrado anteriormente). Procure testar o programa com vários campos minados. Crie-os usando sua imaginação.

Campo Minado: O Jogo

O segundo programa no qual seu TAD deve ser usado é na implementação do Jogo Campo Minado. O Jogo Campo Minado gera um campo aleatório e deve permitir que o usuário percorra suas posições até que uma bomba seja encontrada (então ele perde) ou até que todas as posições seguras sejam visitadas (nesse caso o jogador é sagrado campeão!).

A cada escolha do usuário, o campo deverá ser atualizado mostrando ao usuário o que existia no quadrado escolhido. Caso encontre uma bomba, uma mensagem de fim-de-jogo deverá ser mostrada, caso contrário, um inteiro com o número de bombas adjacentes ao quadrado deverá ser mostrado na posição escolhida.

Ainda, o jogo deve permitir 4 níveis de dificuldade (fácil, médio, difícil e teste), o nível fácil usa campos de tamanho 4 x 4 e em 10% dos seus quadrados estão posicionadas bombas, o nível médio usa campos de tamanho 4 x 4 e em 20% dos seus quadrados estão posicionadas bombas, o nível difícil usa campos de tamanho 5 x 5 e em 40% dos seus quadrados estão posicionadas bombas, o nível teste é igual ao difícil, mas as bombas aparecem para o usuário. A interface do jogo pode ser completamente textual, mas você deve utilizar o TAD criado para implementar o jogo.

Note que o seu programa principal não poderá acessar diretamente a estrutura interna do TAD. Se necessário, acrescente novas funções ao seu TAD detalhando-as na documentação do trabalho

A sua aplicação deverá ser criada para ser "portátil", isto é, ela deverá funcionar com diversos Sistemas Operacionais (Windows, Linux, etc.). Desse modo, o aluno não poderá utilizar métodos da linguagem C que são dependentes do sistema como, por exemplo, a função:

```
int system(const char *command);
```

O que deve ser entregue

- Código fonte do programa em C (bem identado e comentado).
- Documentação do trabalho. Entre outras coisas, a documentação deve conter:

1. Introdução: descrição do problema a ser resolvido e visão geral sobre o funcionamento do programa.
2. Implementação: descrição sobre a implementação do programa. Deve ser detalhada a estrutura de dados utilizada (de preferência com diagramas ilustrativos), o funcionamento das principais funções e procedimentos utilizados, o formato de entrada e saída de dados, bem como decisões tomadas relativas aos casos e detalhes de especificação que porventura estejam omissos no enunciado. **Muito importante:** os códigos utilizados nas implementações devem ser inseridos na documentação.
3. Estudo de complexidade: estudo da complexidade do tempo de execução dos procedimentos implementados e do programa como um todo (notação O).
4. Listagem de testes executados: os testes executados devem ser apresentados, analisados e discutidos.
5. Conclusão: comentários gerais sobre o trabalho e as principais dificuldades encontradas em sua implementação.
6. Bibliografia: bibliografia utilizada para o desenvolvimento do trabalho, incluindo sites da Internet se for o caso. Uma referência bibliográfica deve ser citada no texto quando da sua utilização
7. Em Latex: Caso o trabalho seja elaborado/escrito em latex, ganha-se 0,1 ponto.
8. Formato: mandatoriamente em PDF.

Como deve ser feita a entrega:

A entrega DEVE ser feita pelo Moodle (moodlepresencial.ufop.br) na forma de um único arquivo zipado, contendo o código, os arquivos e a documentação. Também deve ser entregue a documentação impressa na próxima aula (teórica ou prática) após a data de entrega do trabalho.

Comentários Gerais:

- Comece a fazer este trabalho logo, enquanto o problema está fresco na memória e o prazo para terminá-lo está tão longe quanto jamais poderá estar;
- Clareza, identação e comentários no programa também serão avaliados;
- O trabalho é individual (grupo de UM aluno);
- Trabalhos copiados (e FONTE) terão nota zero; Devido a recorrentes problemas com cópias de trabalhos (plágios), os autores de trabalhos copiados também terão todos os demais trabalhos zerados, como forma de punição e coação ao plágio acadêmico;
- Trabalhos entregue em atraso serão aceitos, todavia a nota atribuída ao trabalho será zero;
- Evite discussões inócuas com o professor em tentar postergar a data de entrega do referido trabalho.