BCC202 – Estruturas de Dados I

Departamento de Computação - Universidade Federal de Ouro Preto - MG
Professor: **Pedro Silva** (`www.decom.ufop.br/`)

MANIPULAÇÃO DE MATRIZES

- **Submissão via *Moodle***.

- **Data e hora de entrega disponíveis no *Moodle*.**

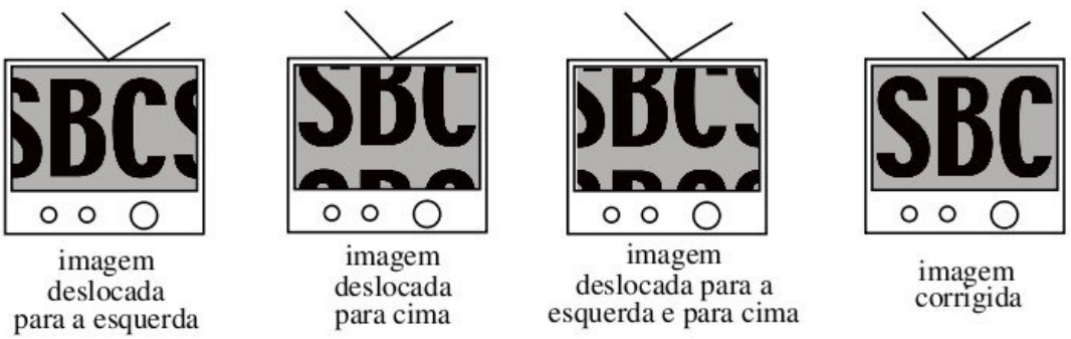
- Procedimento para a entrega:.

1. Os nomes dos arquivos e das funções devem ser especificados considerando boas práticas de programação.
2. Funções auxiliares, complementares aquelas definidas, podem ser especificadas e implementadas, se necessário.
3. A solução deve ser devidamente modularizada e separar a especificação da implementação em arquivos *.h* e *.c* sempre que cabível.
4. Os arquivos a serem entregues, incluindo aquele que contém *main()*, devem ser compactados (*.zip*), sendo o arquivo resultante submetido via ***Moodle***.
5. Caracteres como acento, cedilha e afins não devem ser utilizados para especificar nomes de arquivos ou comentários no código.
6. Siga atentamente quanto ao formato da entrada e saída de seu programa, exemplificados no enunciado.
7. Durante a correção, os programas serão submetidos a vários casos de testes, com características variadas.
8. A avaliação considerará o tempo de execução e o percentual de respostas corretas.
9. Eventualmente, serão realizadas entrevistas sobre os estudos dirigidos para complementar a avaliação.
10. Considere que os dados serão fornecidos pela entrada padrão. Não utilize abertura de arquivos pelo seu programa. Se necessário, utilize o redirecionamento de entrada.
11. Os códigos fonte serão submetidos a uma ferramenta de detecção de plágios em software.
12. Códigos cuja autoria não seja do aluno, com alto nível de similaridade em relação a outros trabalhos, ou que não puder ser explicado, acarretará na perda da nota.
13. Códigos ou funções prontas específicos de algoritmos para solução dos problemas elencados não são aceitos.
14. Não serão considerados algoritmos parcialmente implementados.

- Bom trabalho!

TV da Vovó

A vovó tem um televisor muito antigo, que ultimamente está exibindo um defeito incômodo: a imagem aparece 'deslocada' (para cima ou para baixo, para o lado direito ou para o lado esquerdo). Quando a imagem está deslocada para cima, a parte da imagem que deixa de ser vista na parte superior reaparece na parte de baixo da tela. Da mesma forma, quando a imagem está deslocada a direita, a parte da imagem que deixa de ser vista à direita reaparece na tela do lado esquerdo.



A imagem do televisor pode ser vista como uma matriz de pontos organizados em linhas e colunas. Para consertar o televisor da vovó, você pode ajustar a imagem introduzindo uma série de 'comandos de correção' em um painel de ajuste. Cada comando de correção desloca a imagem de um certo número de linhas (para cima ou para baixo) e um certo número de colunas (para a direita ou para a esquerda).

Considerações

O código-fonte deve ser modularizado corretamente conforme os arquivos de protótipo fornecidos. Um vetor dinâmico de inteiros deve ser alocado e posteriormente desalocado para armazenar os pontos. Cada caso de teste deve ser resolvido em até **1 segundo**!

- Não altere o nome dos arquivos.
- O arquivo `.zip` deve conter na sua raiz somente os arquivos-fonte.
- Há vários casos de teste. Você terá acesso (entrada e saída) de casos específicos para realizar os seus testes.

Especificação da Entrada e da saída

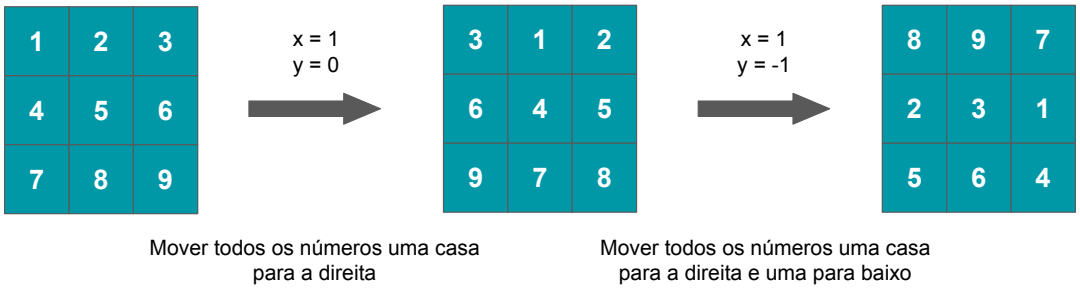
Dada uma matriz que representa uma imagem defeituosa e uma série de comandos de correção, seu programa deve calcular a matriz que representa a imagem resultante após todos os comandos terem sido aplicados sequencialmente.

A entrada possui vários conjuntos de teste. Cada conjunto de teste inicia com a descrição da matriz que representa a imagem do televisor. A primeira linha contém dois inteiros M e N representando o número de linhas e o número de colunas da matriz ($1 \leq M \leq 1000 e 1 \leq N \leq 1000$). As M linhas seguintes da entrada contém cada uma N inteiros, descrevendo o valor de cada ponto da imagem. Após a descrição da imagem, segue-se a descrição dos comandos de correção. Cada comando de correção é descrito em uma linha contendo dois inteiros X e Y. O valor de X representa o deslocamento na direção horizontal (valor positivo representa deslocamento para a direita, valor negativo para a esquerda), e o valor de Y representa o deslocamento da direção vertical (valor positivo para cima, valor negativo para baixo). O final da lista de comandos é indicado por $X = Y = 0$, e o final da entrada é indicado por $M = N = 0$.

Para cada conjunto de teste, o seu programa deve produzir uma imagem na saída. A primeira linha da saída deve conter um identificador do conjunto de teste, no formato "*Teste n*", onde n é numerado sequencialmente a partir de 1. A seguir deve aparecer a matriz que representa a imagem resultante, no mesmo formato da imagem de entrada. Ou seja, as N linhas seguintes devem conter cada uma M inteiros que representam os pixels da imagem. Após a imagem deixe uma linha em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Entrada	**Saída**
3 3 1 2 3 4 5 6 7 8 9 1 0 1 -1 0 0 3 4 6 7 8 5 10 11 12 9 2 3 4 1 -3 2 0 0 0 0	Teste 1 8 9 7 2 3 1 5 6 4 Teste 2 1 2 3 4 5 6 7 8 9 10 11 1

Exemplo de execução para o teste 1 apresentado.



Diretivas de Compilação

```
$ gcc pratica.c -o programa -Wall
```

Avaliação de *leaks* de memória

Uma forma de avaliar se não há *leaks* de memória é usando a ferramenta valgrind. Um exemplo de uso é:

```
gcc -g -o exe *.c -Wall; valgrind -leak-check=yes -s ./exe < casoteste.in
```

Espera-se uma saída com o fim semelhante a:

```
==38409== ERROR SUMMARY: 0 errors from 0 contexts (suppressed:  0 from 0)
```

Para instalar no Linux, basta usar: `sudo apt install valgrind`.