Principais erros encontrados

1. Utilizar pouca alocação dinâmica, ou seja, em alguns casos mesmo com matrizes e vetores grandes, muitos utilizavam ainda estruturas estáticas.
2. Leitura de Strings e escrita de strings, o ideal é fazer seu próprio readLine, no caso da leitura, o scanf e o printf com %s, pode possuem falhas de segurança. (Usar %s não é um erro, mas achei interessante comentar a falha de segurança para eles aprenderem).
3. No caso de liberação de memória, alguns alunos na liberaram a memória por completo. Principalmente no caso de matrizes, pois apenas liberavam o ponteiro pro vetor de ponteiros, e não cada vetor em si (e.g. matriz[i]), além disso structs com ponteiros como atributos da struct, muitas vezes não era liberada essa memória.
4. Uso de loops infinitos em situações desnecessárias.
5. Tentativa de desalocar vetores estáticos.
6. Códigos com pouca modularização (poucas funções).
7. (SUGESTÃO) Alguns códigos estavam muito amarrados com os problemas, as vezes, tentar uma solução mais genérica, facilita na reutilização de código.
8. Códigos inutilizáveis deixados no código, é sempre bom deixar apenas os trechos de códigos utilizados, isso facilita a leitura e a cara do código.
9. As indentações melhoraram, mas é sempre bom seguir o padrão C.
10. Comentários devem ser sucintos, ou seja, devem ser bem explicados e diretos.
11. (SUGESTÃO) Cabeçalho de funções são bem vindo em documentações de código.

    TEMPLATE:

    <tipo retorno> <nome função com parâmetros>

    Parâmetros:

<lista de parametros> (repetir para cada parametro)

<tipo do parâmetro> <nome do parâmetro> <descrição>

Retorno:

<tipo do retorno> <descrição>

EXEMPLO (função que printa uma matriz):

void printMatrix(int **matrix, int n, int m);

Parâmetros:

int **matrix -> Matriz que será printada.

int n -> Número de linhas da matriz.

int m -> Número de colunas da matriz.

Retorno:

Não há.

12. (SUGESTÃO) Não vejo problema em iniciar um ponteiro que será realocado com malloc de uma posição, porém iniciá-lo apenas com NULL é melhor.
13. Usar recursão de maneira errada, deve-se saber diferenciar onde recursão é necessária.
14. (SUGESTÃO) Quando usar vários números hard coded, recomenda-se fazer um define pois ajuda na modificação e entendimento do código.
15. (SUGESTÃO) No geral quando fazemos typedef de uma struct, o novo nome é dado todo em letra maiúscula, não é obrigado, é mais um padrão mesmo.

Considerações Finais

No geral os códigos estão bem melhor escritos que na primeira vez que fiz esse relatório, muitos erros foram corrigidos e os códigos estão mais legíveis, erros, como listado acima foram bem menos frequentes, por isso foquei em dizer mais sugestões de padronização e de documentação.