BCC202 – Estruturas de Dados I (2023-02)

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

AULA PRÁTICA 03

- **Data de entrega: Até 20 de outubro às 23:55.**

- Procedimento para a entrega:.

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

- Bom trabalho!

O estagiário

Você conseguiu um estágio para trabalhar como programador na secretaria da sua escola. Como primeira tarefa, Dona Vilma, a coordenadora, solicitou que você aprimore um programa que foi desenvolvido pelo estagiário anterior. Esse programa tem como entrada uma lista de nomes e de médias finais dos alunos de uma turma, e determina o aluno com a maior média na turma. Dona Vilma pretende utilizar o programa para premiar o melhor aluno de cada turma da escola.

Como você pode verificar, o programa na forma atual tem uma imperfeição: no caso de haver alunos empatados com a melhor média na turma, ele imprime apenas o primeiro aluno que aparece na lista.

Dona Vilma deseja que você altere o programa para que ele produza uma lista com todos os alunos da turma que obtiveram a maior média, e não apenas um deles. Você consegue ajudá-la nesta tarefa?

Considerações

O código-fonte deve ser modularizado corretamente conforme os arquivos de protótipo fornecidos. Um vetor dinâmico da TAD Aluno (disponível em aluno.h) deve ser alocado e posteriormente desalocado para armazenar os alunos. 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 arquivo-fonte (`aluno.h`, `aluno.h` e `pratica.c`).
- Há no total **cinco** casos de teste. Você terá acesso (entrada e saída) a somente **um** deles para realizar os seus testes.

Especificação da Entrada e da saída

A entrada é constituída de vários conjuntos de teste, representando várias turmas. A primeira linha de um conjunto de testes contém um número inteiro N ($1 <= N <= 1000$) que indica o total de alunos na turma. As N linhas seguintes contêm, cada uma, um par de números inteiros C ($1 <= C <= 20000$) e M ($0 <= M <= 100$), indicando respectivamente o código e a média de um aluno. O final da entrada é indicado por uma turma com $N = 0$.

Para cada turma da entrada seu programa deve produzir três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "`Turma n`", onde n é numerado a partir de 1. A segunda linha deve conter os códigos dos alunos que obtiveram a maior média da turma. Os códigos dos alunos devem aparecer na mesma ordem da entrada, e cada um deve ser seguido de um espaço em branco. A terceira linha deve ser deixada em branco. O formato mostrado no exemplo de saída abaixo deve ser seguido rigorosamente.

Entrada	Saída
3 1 85 2 91 3 73 5 12300 81 12601 99 15023 76 10111 99 212 99 0	Turma 1 2 Turma 2 12601 10111 212

Diretivas de Compilação

```
$ gcc -c aluno.c -Wall
$ gcc -c pratica.c -Wall
$ gcc aluno.o pratica.o -o exe
```

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`.