OBI

OLIMPÍADA BRASILEIRA
DE INFORMÁTICA

OBI2003

CADERNO DE TAREFAS - MODALIDADE PROGRAMAÇÀO

31/5/2003 • 13:00 às 18:00

Leia atentamente estas instruções antes de iniciar a prova. Este caderno é composto de 15 páginas, numeradas de 1 a 15. Verifique se o caderno está completo.

1. É proibido consultar livros, anotações ou qualquer outro material durante a prova. É permitido a consulta ao *help* do ambiente de programação se este estiver disponível.
2. **A correção é automatizada**, portanto siga atentamente as exigências da tarefa quanto ao formato da entrada e saída de seu programa.
3. **Não implemente nenhum recurso gráfico nas suas soluções** (janelas, menus, etc), nem utilize qualquer rotina para limpar a tela ou posicionar o cursor.
4. Todas as tarefas têm o mesmo valor na correção.
5. As tarefas **não** estão ordenadas, neste caderno, por ordem de dificuldade; procure resolver primeiro as questões mais fáceis.
6. Preste muita atenção no nome dos arquivos fonte indicados nas tarefas. Soluções na linguagem C devem ser arquivos com sufixo .c; soluções na linguagem C++ devem ser arquivos com sufixo .cc ou .cpp; soluções na linguagem Pascal devem ser arquivos com sufixo .pas.
7. Para problemas diferentes você pode escolher trabalhar com linguagens diferentes, mas apenas uma solução, em uma única linguagem, deve ser submetida para cada problema.
8. Ao final da prova, para cada solução que você queira submeter para correção, copie o **arquivo fonte** para o seu diretório de trabalho ou disquete, conforme especificado pelo seu professor.
9. **Não utilize arquivos para entrada ou saída**. Todos os dados devem ser lidos da entrada padrão (normalmente é o teclado) e escritos na saída padrão (normalmente é a tela). Utilize as funções padrão para entrada e saída de dados:
   - em Pascal: *readln, read, writeln, write*;
   - em C: *scanf, getchar, printf, putchar*;
   - em C++: as mesmas de C ou os objetos *cout* e *cin*.
10. Procure resolver o problema de maneira eficiente. Na correção, eficiência também será levada em conta. As soluções serão testadas com outras entradas além das apresentadas como exemplo nas tarefas.

Sociedade Brasileira de Computação
http://www.sbc.org.br Email: sbc@sbc.org.br

Cofrinhos da Vó Vitória

Arquivo fonte: cofre.c, cofre.cc, cofre.cpp ou cofre.pas

Vó Vitória mantém, desde o nascimento dos netos Joãozinho e Zezinho, um ritual que faz a alegria dos meninos. Ela guarda todas as moedas recebidas como troco em dois pequenos cofrinhos, um para cada neto. Quando um dos cofrinhos fica cheio, ela chama os dois netos para um alegre almoço, ao final do qual entrega aos garotos as moedas guardadas nos cofrinhos de cada um.

Ela sempre foi muito zelosa quanto à distribuição igualitária do troco arrecadado. Quando, por força do valor das moedas, ela não consegue depositar a mesma quantia nos dois cofrinhos, ela memoriza a diferença de forma a compensá-la no próximo depósito.

1. Tarefa

Vó Vitória está ficando velha e tem medo que deslizes de memória a façam cometer injustiças com os netos, deixando de compensar as diferenças entre os cofrinhos. Sua tarefa é ajudar Vó Vitória, escrevendo um programa de computador que indique as diferenças entre os depósitos, de forma que ela não tenha que preocupar-se em memorizá-las.

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de teste contém um número inteiro N, que indica o número de depósitos nos cofrinhos. As N linhas seguintes descrevem cada uma um depósito nos cofrinhos; o depósito é indicado por dois valores inteiros J e Z, separados por um espaço em branco, representando respectivamente os valores, em centavos, depositados nos cofres de Joãozinho e Zezinho. O final da entrada é indicado por $N = 0$.

Exemplo de Entrada

```
3
20 25
10 5
10 10
4
0 5
12 0
0 20
17 1
0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir um conjunto de linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste n", onde n é numerado seqüencialmente a partir de 1. A seguir seu programa deve escrever uma linha para cada depósito do conjunto de testes. Cada linha deve conter um inteiro que representa a diferença (em centavos) entre o valor depositado nos cofrinhos do Joãozinho e do Zezinho. Deixe uma linha em branco ao final de cada conjunto de teste. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
-5
0
0

Teste 2
-5
7
-13
3
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 100$ ($N = 0$ apenas para indicar o fim da entrada)
$0 \leq J \leq 100$ (valor de cada depósito no cofre de Joãozinho)
$0 \leq Z \leq 100$ (valor de cada depósito no cofre de Zezinho)

Estágio

Arquivo fonte: estagio.c, estagio.cc, estagio.cpp ou estagio.pas

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. O programa desenvolvido pelo estagiário anterior encontra-se nas páginas a seguir (programa Pascal na página 5, programa C na página 6, programa C++ na página 7).

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.

1. Tarefa

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?

2. Entrada

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 ($1 \leq N \leq 1000$) que indica o total de alunos na turma. As N linhas seguintes contêm, cada uma, um par de números inteiros C ($1 \leq C \leq 20000$) e M ($0 \leq M \leq 100$), indicando respectivamente o código e a média de um aluno. O final da entrada é indicado por uma turma com $N = 0$.

Exemplo de Entrada

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

3. Saída

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.

Exemplo de Saída

```
Turma 1
2

Turma 2
12601 10111 212
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 1000$ ($N = 0$ apenas para indicar o fim da entrada)
$1 \leq C \leq 20000$
$0 \leq M \leq 100$

Programa em Pascal:

```
program estagio;

const MAX_ALUNOS = 1000;
type
   registro_aluno = record
                       codigo, media : integer;
                    end;
var
   alunos : array[1..MAX_ALUNOS] of registro_aluno;
   n, i, indice_melhor, turma :  integer;
begin
   readln(n); { le numero de alunos da primeira turma }
   turma := 1;
   while n > 0 do begin
      for i := 1 to n do { le dados dos alunos }
         readln(alunos[i].codigo, alunos[i].media);
      indice_melhor := 1;
      for i := 2 to n do { procura aluno de maior media }
         if alunos[i].media > alunos[indice_melhor].media then
            indice_melhor := i;
      { escreve resposta }
      writeln('Turma ', turma);
      turma := turma + 1;
      writeln(alunos[indice_melhor].codigo);
      writeln;
      { le numero de alunos da proxima turma }
      readln(n);
   end;
end.
```

Programa em C:

```
#include <stdio.h>

#define MAX_ALUNOS 1000

int main()
{
  int i, indice_melhor, n;
  int turma=1;
  struct
  {
    int codigo, media;
  } alunos[MAX_ALUNOS];
  /* le numero de alunos da primeira turma */
  scanf("%d", &n);
  while (n > 0)
    {
      /* le dados dos alunos */
      for (i = 0; i < n; i++)
        scanf("%d %d", &alunos[i].codigo, &alunos[i].media);
      /* procura aluno de maior media */
      indice_melhor = 0;
      for (i = 1; i < n; i++)
        if (alunos[i].media > alunos[indice_melhor].media)
            indice_melhor = i;
      /* escreve resposta */
      printf("Turma %d\n%d\n\n", turma++, alunos[indice_melhor].codigo);
      /* le numero de alunos da proxima turma */
      scanf("%d", &n);
    }
  return 0;
}
```

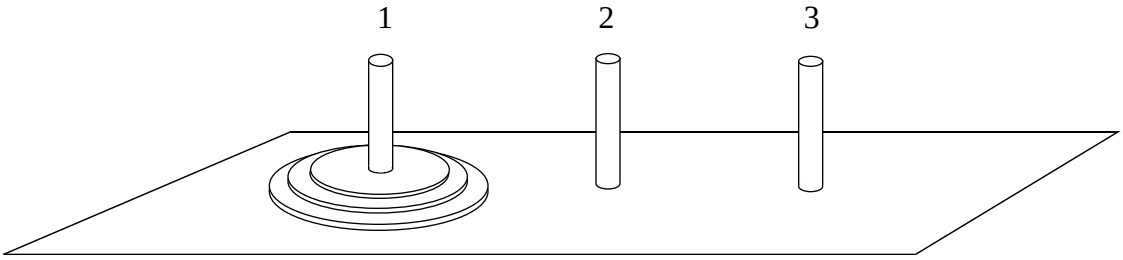
Programa em C++:

```
#include <iostream>

const int MAX_ALUNOS = 1000;

int main()
{
  int i, indice_melhor, n;
  int turma=1;
  struct
  {
    int codigo, media;
  } alunos[MAX_ALUNOS];

  // le numero de alunos da primeira turma
  cin >> n;
  while (n > 0)
    {
      // le dados dos alunos
      for (i = 0; i < n; i++)
        cin >> alunos[i].codigo >> alunos[i].media;
      // procura aluno de maior media
      indice_melhor = 0;
      for (i = 1; i < n; i++)
        if (alunos[i].media > alunos[indice_melhor].media)
            indice_melhor = i;
      // escreve resposta
      cout << "Turma " << turma++ << "\n";
      cout << alunos[indice_melhor].codigo << "\n\n";

      // le numero de alunos da proxima turma
      cin >> n;
    }
  return 0;
}
```

Torres de Hanói

Arquivo fonte: hanoi.c, hanoi.cc, hanoi.cpp ou hanoi.pas

O quebra-cabeças Torres de Hanoi é muito antigo e conhecido, sendo constituído de um conjunto de *N* discos de tamanhos diferentes e três pinos verticais, nos quais os discos podem ser encaixados.



Cada pino pode conter uma pilha com qualquer número de discos, desde que cada disco não seja colocado acima de outro disco de menor tamanho. A configuração inicial consiste de todos os discos no pino 1. O objetivo do quebra-cabeças é mover todos os discos para um dos outros pinos, sempre obedecendo à restrição de não colocar um disco sobre outro menor.

Um algoritmo para resolver este problema é o seguinte.

```
procedimento Hanoi(N, Orig, Dest, Temp)
  se N = 1 então
    mover o menor disco do pino Orig para o pino Dest;
  senão
    Hanoi(N-1, Orig, Temp, Dest);
    mover o N-ésimo menor disco do pino Orig para o pino Dest;
    Hanoi(N-1, Temp, Dest, Orig);
  fim-se
fim
```

1. Tarefa

Sua tarefa é escrever um programa que determine quantos movimentos de trocar um disco de um pino para outro serão executados pelo algoritmo acima para resolver o quebra-cabeça.

2. Entrada

A entrada possui vários conjuntos de teste. Cada conjunto de teste é composto por uma única linha, que contém um único número inteiro N ($0 \leq N \leq 30$), indicando o número de discos. O final da entrada é indicado por $N = 0$.

Exemplo de Entrada

```
1
2
0
```

3. Saída

Para cada conjunto de teste, o seu programa deve escrever três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado seqüencialmente a partir de 1. A segunda linha deve conter o número de movimentos que são executados pelo algoritmo dado para resolver o problema das Torres de Hanói com *N* discos. A terceira linha deve ser deixada em branco. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
1

Teste 2
3
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq N \leq 30$ ($N = 0$ apenas para indicar o final da entrada)

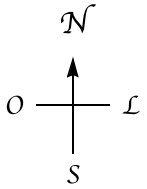
Supermercado

Arquivo fonte: super.c, super.cc, super.cpp ou super.pas

A rede de supermercados BemBom, da cidade de Planalto, decidiu reformular o armazenamento de seus estoques. No sistema atual, cada uma das lojas da rede possui espaço para armazenar um pequeno estoque, sendo freqüentemente necessário transportar mercadorias de uma loja para outra. Para racionalizar o transporte e aumentar a capacidade de estoque, a direção da rede BemBom decidiu instalar um depósito central. De forma a diminuir os custos com transporte, ficou definido que o novo depósito deve ser localizado em um quarteirão que minimize a soma das distâncias dele até todas as lojas da rede.

Por ser uma cidade planejada, Planalto possui uma característica muito peculiar. Todas as suas ruas são orientadas na direção leste-oeste ou norte-sul, e todos os quarteirões são do mesmo tamanho. Veja uma parte do mapa de Planalto na figura abaixo. Os quarteirões em Planalto são identificados pelo número de quadras, em cada direção, que os separam da localização da prefeitura (0,0). Localizações a leste e a norte da prefeitura são identificadas por coordenadas positivas, e localizações a oeste e a sul por coordenadas negativas.

-2, 2	-1, 2	0, 2	1, 2	2, 2
-2, 1	-1, 1	0, 1	1, 1	2, 1
-2, 0	-1, 0	0, 0	1, 0	2, 0
-2, -1	-1, -1	0, -1	1, -1	2, -1
-2, -2	-1, -2	0, -2	1, -2	2, -2



Parte do mapa de Planalto

1. Tarefa

A sua tarefa é, dadas as coordenadas dos quarteirões onde estão localizados todos os supermercados da rede, determinar o quarteirão onde deve ser instalado o novo depósito. A localização deste depósito deve ser tal que a soma das distâncias entre o depósito e as lojas, em número de quarteirões em ambas as direções, seja a menor possível. A distância entre dois quarteirões é dada pela distância entre eles na direção leste-oeste mais a distância na direção norte-sul. Por exemplo, a distância entre os quarteirões (2,-1) e (4, 3) é 2 + 4 = 6.

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de cada conjunto de teste contém um número inteiro *S* que é o número de supermercados da rede. A seguir, são dadas *S* linhas, cada uma contendo dois números inteiros *X* e *Y*, representando as coordenadas do quarteirão onde se situa um dos supermercados. *X* representa a coordenada na direção leste-oeste e *Y* represetna a coordenada na direção norte-sul. O final da entrada é dado por um conjunto de teste com *S* = 0.

Exemplo de Entrada

```
4
1 2
-3 -3
4 -1
-5 0
5
1 3
7 13
25 13
15 14
25 1
0
```

3. Saída

Para cada conjunto de teste, o seu programa deve escrever três linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado seqüencialmente a partir de 1. A segunda linha deve conter as coordenadas *X* e *Y* do quarteirão onde deve ser instalado o novo depósito, separadas por um espaço em branco. Se mais de um quarteirão puder ser escolhido como localização do depósito, seu programa pode imprimir qualquer um deles. A terceira linha deve ser deixada em branco. O formato do exemplo de saída abaixo deve ser seguido rigorosamente.

Exemplo de Saída

```
Teste 1
-2 0

Teste 2
15 13
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq S \leq 1000$ ($S = 0$ apenas para indicar o final da entrada)
$-1000 \leq X \leq 1000$
$-1000 \leq Y \leq 1000$

Número de Erdos

arquivo fonte: erdos.pas, erdos.c, erdos.cc ou erdos.cpp

O matemático húngaro Paul Erdos (1913-1996), um dos mais brilhantes do século XX, é considerado o mais prolífico matemático da história. Erdos publicou mais de 1500 artigos, em colaboração com cerca de outros 450 matemáticos. Em homenagem a este gênio húngaro, os matemáticos criaram um número, denominado "número de Erdos". Toda pessoa que escreveu um artigo com Erdos tem o número 1. Todos que não possuem número 1, mas escreveram algum artigo juntamente com alguém que possui número 1, possuem número 2. E assim por diante. Quando nenhuma ligação pode ser estabelecida entre Erdos e uma pessoa, diz-se que esta possui número de Erdos infinito. Por exemplo, o número de Erdos de Albert Einstein é 2. E, talvez surpreendentemente, o número de Erdos de Bill Gates é 4.

1. Tarefa

Sua tarefa é escrever um programa que, a partir de uma lista de autores de artigos, determine o número de Erdos dos autores.

2. Entrada

A entrada é constituída por vários conjuntos de teste. A primeira linha de um conjunto de teste contém um número inteiro A ($1 \leq A \leq 100$), que indica o número de artigos. Cada uma das A linhas seguintes contém a lista de autores de um artigo. Cada autor é identificado pela inicial de seu nome (em maiúscula), seguida de um ponto e de um espaço em branco (indicando que o nome está abreviado), seguida de seu último sobrenome (‘`P. Erdos`’, por exemplo). O sobrenome de um autor possui, no máximo, 15 letras, e apenas a letra inicial aparece em maiúscula. Os autores são separados por vírgulas, e a lista de autores de um artigo termina com um ponto (veja os exemplos abaixo). Um único espaço em branco separa a abreviatura do nome do sobrenome, bem como o nome de um autor do anterior. Espaços em branco não são usados em outros locais. Um artigo possui, no máximo, 10 autores, e o total de autores não excede 100. O final da entrada é indicado por $A = 0$.

Exemplo de Entrada

```
5
P. Erdos, A. Selberg.
P. Erdos, J. Silva, M. Souza.
M. Souza, A. Selberg, A. Oliveira.
J. Ninguem, M. Ninguem.
P. Duarte, A. Oliveira.
2
Z. Silva, P. Erdos.
Z. Souza.
0
```

3. Saída

Para cada conjunto de teste da entrada seu programa deve produzir um conjunto de linhas na saída. A primeira linha deve conter um identificador do conjunto de teste, no formato “Teste *n*”, onde *n* é numerado seqüencialmente a partir de 1. A seguir devem aparecer uma linha para cada

autor do conjunto de testes (exceto o próprio P. Erdos). Cada linha deve conter o nome do autor seguido pelo caractere ':', um espaço em branco e o seu número de Erdos. Caso o número de Erdos de um determinado autor seja infinito, escreva '`infinito`'. A saída deve ser ordenada alfabeticamente pelo sobrenome do autor, e, em caso de mesmo sobrenome, o desempate deve ser feito pela inicial do primeiro nome. Imprima uma linha em branco ao final de cada conjunto de teste. A grafia mostrada no Exemplo de Saída, abaixo, deve ser seguida rigorosamente.

Exemplo de Saída

```
Teste 1
P. Duarte: 3
J. Ninguem: infinito
M. Ninguem: infinito
A. Oliveira: 2
A. Selberg: 1
J. Silva: 1
M. Souza: 1

Teste 2
Z. Silva: 1
Z. Souza: infinito
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq A \leq 100$ (número de artigos de um caso de teste; $A = 0$ apenas para indicar final da entrada)
$1 \leq$ *tamanho, em número de letras, do sobrenome de um autor* ≤ 15
$1 \leq$ *número de autores de um artigo* ≤ 10
$1 \leq$ *número total de autores em um conjunto de teste* ≤ 100

Tetris

Arquivo fonte: tetris.c, tetris.cc, tetris.cpp ou tetris.pas

A sua turma do colégio resolveu organizar um campeonato de tetris. Após discussão sobre as regras, ficou definido que cada aluno jogaria um total de 12 partidas. Das 12 pontuações obtidas por um aluno, a maior e a menor são descartadas, e as demais são somadas, resultando na pontuação final do aluno.

1. Tarefa

Como você possui conhecimentos de programação, acabou sendo designado pela turma para escrever um programa para imprimir a classificação final do campeonato, a partir das pontuações de cada jogador.

2. Entrada

A entrada é composta de vários conjuntos de teste. A primeira linha de um conjunto de testes contém um número inteiro *J*, que indica o número de jogadores que participaram do campeonato. A seguir, para cada jogador há duas linhas na entrada: a primeira possui o nome do jogador (formado apenas por letras, sendo apenas a inicial em maiúscula, e com no máximo 15 letras), e a segunda possui as 12 pontuações que o jogador obteve, separadas por espaço. As pontuações são inteiros entre 0 e 1000. O final da entrada é indicado por um conjunto de teste com $J = 0$.

Exemplo de Entrada

```
4
Zezinho
100 123 133 333 400 300 129 200 360 340 200 600
Luizinho
60 50 120 250 170 190 190 220 260 270 290 300
Carlinhos
10 10 20 10 10 10 10 20 20 20 20 20
Joaozinho
200 300 400 400 500 500 500 600 650 650 700 810
3
Pedrinho
100 100 200 200 300 300 400 400 500 500 600 600
Huguinho
50 100 200 200 300 300 500 500 400 400 600 700
Zezinho
100 100 100 100 100 100 100 100 100 100 100 100
0
```

3. Saída

Para cada conjunto de teste, o seu programa deve escrever uma linha contendo o identificador do conjunto de teste, no formato "Teste *n*", onde *n* é numerado seqüencialmente a partir de 1. A seguir, o seu programa deve escrever a classificação final no campeonato, utilizando uma linha

para cada participante. Cada linha deve conter três informações, separadas por um espaço em branco: a classificação do jogador, a sua pontuação final, e o seu nome. A classificação de um jogador é igual a 1 mais o número de jogadores que obtiveram pontuação maior do que a sua. Em caso de empate, os jogadores devem ser ordenados em ordem alfabética. Depois de toda a classificação, deve ser deixada uma linha em branco. O formato do exemplo de saída abaixo deve ser seguido rigorosamente.

Exemplo de Saída

```
Teste 1
1 5200 Joaozinho
2 2518 Zezinho
3 2020 Luizinho
4 150 Carlinhos

Teste 2
1 3500 Huguinho
1 3500 Pedrinho
3 1000 Zezinho
```

(esta saída corresponde ao exemplo de entrada acima)

4. Restrições

$0 \leq J \leq 1000$ ($J = 0$ apenas para indicar final da entrada)
$0 \leq$ *pontuação em uma partida* ≤ 1000
$1 \leq$ *tamanho dos nomes, em número de letras* ≤ 15