Departamento de Informática – PUC-Rio
INF1620 - Estruturas de Dados

Lista 2 Completa – 2006.1 (Gerada em 8 de maio de 2006)

1. Considerando a estrutura

```
struct Ponto {
   int x;
   int y;
};
```

   para representar um ponto em uma grade 2D, implemente uma função que indique se um ponto p está localizado dentro ou fora de um retângulo. O retângulo é definido por seus vértices inferior esquerdo v_1 e superior direito v_2. A função deve retornar 1 caso o ponto esteja localizado dentro do retângulo e 0 caso contrário. Essa função deve obedecer o protótipo:

```
int dentroRet (struct Ponto* v1, struct Ponto* v2, struct Ponto* p);
```

2. Considerando a estrutura

```
struct Ponto {
   int x;
   int y;
};
```

   para representar um ponto em uma grade 2D, implemente uma função que indique se um ponto p está localizado dentro ou fora de um círculo. O círculo é definido por seu centro c e seu raio r. A função deve retornar 1 caso o ponto esteja localizado dentro do círculo e 0 caso contrário. Essa função deve obedecer o protótipo:

```
int dentroCirculo (struct Ponto* c, int raio, struct Ponto* p);
```

3. Considerando a estrutura

```
struct Vetor {
   float x;
   float y;
   float z;
};
```

   para representar um vetor no R^3, implemente uma função que calcule a soma de dois vetores. Essa função deve obedecer o protótipo:

```
void soma (struct Vetor* v1, struct Vetor* v2, struct Vetor* res);
```

   onde os parâmetros $v1$ e $v2$ são ponteiros para os vetores a serem somados, e o parâmetro res é um ponteiro para uma estrutura vetor onde o resultado da operação deve ser armazenado.

4. Considerando a estrutura

```
struct Vetor {
   float x;
   float y;
   float z;
};
```

para representar um vetor no R^3, implemente uma função que calcule a subtração de dois vetores. Essa função deve obedecer o protótipo:

```
void sub (struct Vetor* v1, struct Vetor* v2, struct Vetor* res);
```

onde os parâmetros $v1$ e $v2$ são ponteiros para os vetores a serem subtraídos, e o parâmetro *res* é um ponteiro para uma estrutura vetor onde o resultado da operação $v1 - v2$ deve ser armazenado.

5. Considerando a estrutura

```
struct Vetor {
   float x;
   float y;
   float z;
};
```

para representar um vetor no R^3, implemente uma função que calcule o produto escalar de dois vetores. Essa função deve obedecer o protótipo:

```
float dot (struct Vetor* v1, struct Vetor* v2);
```

onde os parâmetros $v1$ e $v2$ são ponteiros para os vetores a serem multiplicados, e o resultado da operação $v1 \cdot v2$ deve ser retornado.

6. Considerando a estrutura

```
struct Vetor {
   float x;
   float y;
   float z;
};
```

para representar um vetor no R^3, implemente uma função que multiplique os componentes de um vetor por um número escalar. Essa função deve obedecer o protótipo:

```
void mult (struct Vetor* v, float escalar);
```

onde o parâmetros v é um ponteiro para o vetor a ser multiplicado e *escalar* é o valor pelo qual os componentes do vetor devem ser multiplicados. Observe que o vetor apontado pelo ponteiro v tem seu conteúdo alterado por essa função.

7. Considere um arquivo texto que descreve um conjunto de retângulos, triângulos e círculos. Cada linha do arquivo contém a descrição de uma figura. O primeiro caractere não branco da linha indica o tipo da figura: 'r' para retângulo, 't' para triângulo e 'c' para círculo. Esse caractere é seguido por valores reais: valores da base e da altura, no caso de retângulos e triângulos, e valor do raio, no caso de círculo. O arquivo pode conter eventuais linhas em branco. Um exemplo desse formato é mostrado abaixo.

```
r  2.0  4.3
t  4.0  5.0
c  2.0

r  1.0  1.5
r  6.0  0.5
c  3.0
t  1.0  1.02
```

   Escreva um programa completo que leia o arquivo "entrada.txt", que contém as descrições de figuras no formato descrito acima, e imprima na tela, com duas casas decimais, o valor da maior área das figuras listadas no arquivo. Se não for possível abrir o arquivo, o programa deve ter como saída a mensagem "ERRO". Se não existir nenhuma figura no arquivo (arquivo existente, mas vazio), deve-se imprimir a mensagem "VAZIO". Para cálculo da área do círculo use o valor de PI igual a 3.14159.

8. Considere um arquivo texto que descreve um conjunto de retângulos, triângulos e círculos. Cada linha do arquivo contém a descrição de uma figura. O primeiro caractere não branco da linha indica o tipo da figura: 'r' para retângulo, 't' para triângulo e 'c' para círculo. Esse caractere é seguido por valores reais: valores da base e da altura, no caso de retângulos e triângulos, e valor do raio, no caso de círculo. O arquivo pode conter eventuais linhas em branco. Um exemplo deste formato é mostrado abaixo.

```
r  2.0  4.3
t  4.0  5.0
c  2.0
r  1.0  1.5

r  6.0  0.5
c  3.0
t  1.0  1.02
```

   Escreva um programa completo que leia o arquivo "entrada.txt", que contém as descrições de figuras no formato descrito acima, e imprima na tela, com duas casas decimais, o valor da média das áreas das figuras listadas no arquivo. Se não for possível abrir o arquivo, o programa deve ter como saída a mensagem "ERRO". Se não existir nenhuma figura no arquivo (arquivo existente, mas vazio), deve-se imprimir a mensagem "VAZIO". Para cálculo da área do círculo use o valor de PI igual a 3.14159.

9. Considere um arquivo de texto que contém o cadastro dos alunos de uma turma . Cada linha do arquivo contém o nome do aluno, delimitado por aspas simples, e três notas obtidas pelo aluno, representadas por números reais. Considere que os nomes dos alunos têm no máximo 80 caracteres. O arquivo pode conter eventuais linhas em branco. Um exemplo deste formato é mostrado abaixo:

```
'Fulano de Tal'    5.6  7.6  8.1
'Sicrano Silva'    2.0  4.6  4.2

'Beltrano Alves'   10.0 8.8  9.0
```

   Escreva um programa completo que leia o arquivo "entrada.txt", que contém um cadastro de alunos no formato descrito acima, e imprima na tela, com uma casa decimal, a maior média obtida por um aluno considerando as três notas com pesos iguais. Se não for possível abrir o arquivo, o programa deve ter como saída a mensagem "ERRO". Se não existir nenhum aluno descrito no arquivo (arquivo existente, mas vazio), deve-se imprimir a mensagem "VAZIO".

10. Considere um arquivo de texto que contém o cadastro dos alunos de uma turma. Cada linha do arquivo contém o nome do aluno, delimitado por aspas simples, e três notas obtidas pelo aluno, representadas por números reais. Considere que os nomes dos alunos têm no máximo 80 caracteres. O arquivo pode conter eventuais linhas em branco. Um exemplo deste formato é mostrado abaixo:

```
'Fulano de Tal'    5.6  7.6  8.1
'Sicrano Silva'    2.0  4.6  4.2

'Beltrano Alves'   10.0 8.8  9.0
```

   Escreva um programa completo que leia o arquivo "entrada.txt", que contém um cadastro de alunos no formato descrito acima, e imprima na tela o número de alunos com média maior ou igual a 5.0, considerando as três notas com pesos iguais. Se não for possível abrir o arquivo, o programa deve ter como saída a mensagem "ERRO". Se não existir nenhum aluno descrito no arquivo (arquivo existente, mas vazio), deve-se imprimir a mensagem "VAZIO".

11. Considere um tipo que representa os dados de um aluno definido pela estrutura a seguir:

```
struct aluno {
   int mat;            /* número de matrícula */
   char turma;         /* turma do aluno: a, b, c, d, ou e */
   char nome[81];      /* nome do aluno */
   float p1, p2, p3;   /* notas */
};
typedef struct aluno Aluno;
```

   Considere um vetor que armazena, em ordem crescente do número de matrícula, ponteiros para estruturas Aluno. Usando a técnica de busca binária vista no curso, escreva

uma função para buscar um aluno no vetor dado um número de matrícula. A função deve receber como parâmetros o número de alunos, o vetor e o número de matrícula a ser buscado. A função deve retornar o índice no vetor onde o elemento está armazenado. Se o número de matrícula não for encontrado no vetor, a função deve retornar -1. A função deve obedecer ao protótipo definido a seguir:

```
int busca (int n, Aluno** v, int mat);
```

12. Considere um tipo que representa os dados de um aluno definido pela estrutura a seguir:

```
struct aluno {
   int matricula;      /* número de matrícula */
   char turma;         /* turma do aluno: a, b, c, d, ou e */
   char nome[81];      /* nome do aluno */
   float p1, p2, p3;   /* notas */
};
typedef struct aluno Aluno;
```

Considere um vetor que armazena, em ordem crescente do número de matrícula, estruturas Aluno. Usando a técnica de busca binária vista no curso, escreva uma função para buscar um aluno no vetor dado um número de matrícula. A função deve receber como parâmetros o número de alunos, o vetor e o número de matrícula a ser buscado. A função deve retornar o índice no vetor onde o elemento está armazenado. Se o número de matrícula não for encontrado no vetor, a função deve retornar -1. A função deve obedecer ao protótipo definido a seguir:

```
int busca (int n, Aluno* v, int mat);
```

13. Considere um tipo que representa os dados de um aluno definido pela estrutura a seguir:

```
struct aluno {
   int matricula;      /* número de matrícula */
   char turma;         /* turma do aluno: a, b, c, d, ou e */
   char nome[81];      /* nome do aluno */
   float p1, p2, p3;   /* notas */
};
typedef struct aluno Aluno;
```

Considere um vetor que armazena, em ordem alfabética de nomes, ponteiros para estruturas Aluno. Considere que os nomes armazenados contêm apenas letras minúsculas. Usando a técnica de busca binária vista no curso, escreva uma função para buscar um aluno no vetor dado um nome. A função deve receber como parâmetros o número de alunos, o vetor e o nome a ser buscado. A função deve retornar o índice no vetor onde o elemento está armazenado. Se o número de matrícula não for encontrado no vetor, a função deve retornar -1. A função deve obedecer ao protótipo definido a seguir:

```
int busca (int n, Aluno** v, char* nome);
```

14. Considere um tipo que representa os dados de um aluno definido pela estrutura a seguir:

```
struct aluno {
   int matricula;      /* número de matrícula */
   char turma;         /* turma do aluno: a, b, c, d, ou e */
   char nome[81];      /* nome do aluno */
   float p1, p2, p3;   /* notas */
};
typedef struct aluno Aluno;
```

Considere um vetor que armazena, em ordem alfabética de nomes, estruturas Aluno. Considere que os nomes armazenados contêm apenas letras minúsculas. Usando a técnica de busca binária vista no curso, escreva uma função para buscar um aluno no vetor dado um nome. A função deve receber como parâmetros o número de alunos, o vetor e o nome a ser buscado. A função deve retornar o índice no vetor onde o elemento está armazenado. Se o número de matrícula não for encontrado no vetor, a função deve retornar -1. A função deve obedecer ao protótipo definido a seguir:

```
int busca (int n, Aluno* v, char* nome);
```

15. Considere um tipo que representa os dados de um aluno definido pela estrutura a seguir:

```
struct aluno {
   int mat;            /* número de matrícula */
   char turma;         /* turma do aluno: a, b, c, d, ou e */
   char nome[81];      /* nome do aluno */
   float p1, p2, p3;   /* notas */
};
typedef struct aluno Aluno;
```

Usando alguma das técnicas de ordenação vistas no curso, escreva uma função para ordenar um vetor de alunos, em ordem decrescente da média obtida considerando-se as três notas com pesos iguais, usando como critério de desempate o número de matrícula, em ordem crescente. A função deve receber como parâmetros o número de alunos e um vetor que armazena ponteiros para estruturas do tipo Aluno, de acordo com o protótipo definido a seguir:

```
void ordena (int n, Aluno** v);
```

16. Considere um tipo que representa os dados de um aluno definido pela estrutura a seguir:

```
struct aluno {
   int mat;            /* número de matrícula */
   char turma;         /* turma do aluno: a, b, c, d, ou e */
   char nome[81];      /* nome do aluno */
   float p1, p2, p3;   /* notas */
};
typedef struct aluno Aluno;
```

Usando alguma das técnicas de ordenação vistas no curso, escreva uma função para ordenar um vetor de alunos, em ordem decrescente da média obtida considerando-se as três notas com pesos iguais, usando como critério de desempate os nomes dos alunos em ordem alfabética. Considere que todos os nomes são representado por letras minúsculas. A função deve receber como parâmetros o número de alunos e um vetor que armazena estruturas do tipo Aluno, de acordo com o protótipo definido a seguir:

```
void ordena (int n, Aluno* v);
```

17. Considere um tipo que representa os dados de um aluno definido pela estrutura a seguir:

```
struct aluno {
   int mat;            /* número de matrícula */
   char turma;         /* turma do aluno: a, b, c, d, ou e */
   char nome[81];      /* nome do aluno */
   float p1, p2, p3;   /* notas */
};
typedef struct aluno Aluno;
```

Usando alguma das técnicas de ordenação vistas no curso, escreva uma função para ordenar um vetor de alunos, em ordem crescente da turma, usando como critério de desempate os nomes dos alunos em ordem alfabética. Considere que todos os nomes são representado por letras minúsculas. A função deve receber como parâmetros o número de alunos e um vetor que armazena ponteiros para estruturas do tipo Aluno, de acordo com o protótipo definido a seguir:

```
void ordena (int n, Aluno** v);
```

18. Considere um tipo que representa os dados de um aluno definido pela estrutura a seguir:

```
struct aluno {
   int mat;            /* número de matrícula */
   char turma;         /* turma do aluno: a, b, c, d, ou e */
   char nome[81];      /* nome do aluno */
   float p1, p2, p3;   /* notas */
};
typedef struct aluno Aluno;
```

Usando alguma das técnicas de ordenação vistas no curso, escreva uma função para ordenar um vetor de alunos, em ordem crescente da turma, usando como critério de desempate o número de matrícula, em ordem crescente. A função deve receber como parâmetros o número de alunos e um vetor que armazena estruturas do tipo Aluno, de acordo com o protótipo definido a seguir:

```
void ordena (int n, Aluno* v);
```