Universidade Federal de Goiás
Instituto de Informática
Introdução à Programação
Lista - Structs e Alocação Dinâmica de Memória

Prof. Msc. Elias Batista Ferreira
Prof. Msc. Gilmar Ferreira Arantes
Prof. Dr. Gustavo Teodoro Laureano
Profa. Dra. Luciana Berretta
Prof. Dr. Thierson Rosa Couto

12/11/2020

Instruções para a Resolução dos Problemas

Esta lista de exercícios sobre Structs e Alocação Dinâmica de Memória é composta por 9 questões, variando do nível de dificuldade 1 a 4. Você pode resolvê-la seguindo as seguintes orientações:

1. Para garantir sua presença na aula assíncrona dos dias 12 e 17/11/2020, é necessária a resolução de no mínimos 6 questões.
2. Para garantir a pontuação total (10), atribuída à lista é necessária a resolução todas as questões questões.
3. Ao utilizar valores de ponto flutuante, usar o tipo double, mesmo que a especificação mencione que o tipo é float.

Sumário

1 Cursos

 (+)

Uma universidade particular possui uma tabela de valores de créditos por curso com os campos:

- código do curso (int),
- valor por crédito (float).
- nome do curso (cadeia com no máximo 100 caracteres),

Faça um programa que primeiramente carregue a tabela de cursos acima e depois leia registros de alunos com as seguintes informações:

- nome do aluno (cadeia com no máximo 500 caracteres),
- o código do curso onde o mesmo está matriculado (int), e
- o número de créditos que ele está cursando (int).

Calcule a mensalidade a ser paga pelo aluno e imprima as informações em um boleto de pagamento com as seguintes informações: nome do aluno, nome do curso, total de créditos que o aluno cursa, valor do crédito do curso e o valor final da mensalidade a pagar.

Entrada

A primeira linha da entrada contém o número n ($5 \leq n \leq 30$) de cursos a universidade. Em seguida há $3n$ linhas contendo as três informações dos n cursos. A próxima linha contém um valor inteiro que corresponde ao número m ($1 \leq m \leq 1000$) de alunos da universidade. Em seguida, há $3m$ linhas, com os três dados dos m alunos.

Saída

A saída é formada por m linhas, cada uma no seguinte formato: Aluno(a): a Curso: b Num. Creditos: c Valor Credito: d Mensalidade: e, onde a e b são cadeias de caracteres, c é um número inteiro e d e e são números em ponto flutuante com duas casas decimais.

Exemplo

Entrada
4 298 234.5 Direito 123 150.00 Eng. Eletrica 452 132.00 Eng. Civil 341 120.00 Farmacia 5 Joao Luiz 298 5 Paula Lima 452 4 Maria 123 6 Luiz Andre 341 3 Antonio Luiz 341 8
Saída
Aluno(a): Joao Luiz Curso: Direito Num. Creditos: 5 Valor Credito: 234.50 Mensalidade: 1172.50 Aluno(a): Paula Lima Curso: Eng. Civil Num. Creditos: 4 Valor Credito: 132.00 Mensalidade: 528.00 Aluno(a): Maria Curso: Eng. Eletrica Num. Creditos: 6 Valor Credito: 150.00 Mensalidade: 900.00 Aluno(a): Luiz Andre Curso: Farmacia Num. Creditos: 3 Valor Credito: 120.00 Mensalidade: 360.00 Aluno(a): Antonio Luiz Curso: Farmacia Num. Creditos: 8 Valor Credito: 120.00 Mensalidade: 960.00

2 Frações Equivalentes

 (+)

Tia Zuleika está ensinado seus alunos sobre frações equivalentes. Ela vai gerar uma lista de frações e vai pedir aos seus alunos para indicarem quais pares de frações são equivalentes entre si, percorrendo a lista da esquerda para direita. Ela pretende gerar uma lista grande, para que todos os alunos da turma tenham oportunidade de responder. Tia Zuleika está muito ocupada e sabendo que você é "fera" em programação, está pedindo sua ajuda para que faça um programa capaz de ler sequências de frações e para cada sequência indicar quais pares são equivalentes entre si. O programa deve comparar a primeira fração com todas as outras seguintes, depois deve comparar a segunda com todas as outras seguintes e assim por diante, de modo que a posição na lista da primeira fração de um par de frações seja sempre menor que a posição da segunda fração do par.

Entrada

A primeira linha da entrada contém o número $n \leq 30$ de casos de teste. Para cada caso de teste há duas linhas na entrada. A primeira linha, contém o tamanho $m(m \leq 50)$ da sequência de frações. A segunda linha contém a sequência de m frações separadas entre si por um espaço. Cada fração tem o seguinte formato: x/y.

Saída

Para cada caso de teste a saída é formada por uma frase no formato "Caso de Teste t", onde t corresponde ao número de um caso de teste. Em seguida, o programa deve imprimir tantas linhas no formato "x/y equivalente a w/u" quantos forem os pares de frações equivalentes na sequência. Se não houver pares equivalentes o programa deve emitir uma linha com a mensagem: "Nao ha fracoes equivalentes na sequencia".

Exemplo

Entrada
2 5 1/2 1/3 1/4 1/5 1/6 5 1/2 2/4 3/5 10/20 3/6
Saída
Caso de teste 1 Nao ha fracoes equivalentes na sequencia Caso de teste 2 1/2 equivalente a 2/4 1/2 equivalente a 10/20 1/2 equivalente a 3/6 2/4 equivalente a 10/20 2/4 equivalente a 3/6 10/20 equivalente a 3/6

3 Vetores Ordenados

 (++)

Faça um programa que leia vários pares de pontos no espaço de quatro dimensões e calcule a norma do vetor correspondente a cada vetor e imprima as normas dos vetores em ordem crescente. A norma de um vetor $A(a_u, a_x, a_y, a_z)$ no espaço tetradimensional corresponde a sua distância e o ponto de origem e $O(0,0,0,0)$ e é calculada como:

$$||A|| = \sqrt{a_u^2 + a_x^2 + a_y^2 + a_z^2} \tag{1}$$

Voce deve usar um vetor de structs para armazenar as coordenadas e a norma de cada ponto.

Entrada

A entrada consiste de várias linhas. A primeira linha apresenta um número de pontos N, com $2 \leq N \leq 1000$. As N linhas seguintes apresentam pontos no espaço na forma $u,x,y.z$ com u,x,yez números reais tais que $-1000 \leq u,x,y,z \leq 1000$.

Saída

A saída consiste de N linhas, cada uma no formato: "Vetor: (a, b, c, d) Norma: x", onde a,b,c,d correspondem à coordenadas de um vetor lido, com duas casas decimais cada e x o valor de sua norma com duas casas decimais. As linhas devem conter os vetores em ordem crescente de norma.

Exemplo

Entrada
4 1 1 5 2 2 -1 3 0.2 4 2 -1 0.9 -3 4 2 34.2
Saída
Vetor: (2.00, -1.00, 3.00, 0.20) Norma: 3.75 Vetor: (4.00, 2.00, -1.00, 0.90) Norma: 4.67 Vetor: (1.00, 1.00, 5.00, 2.00) Norma: 5.57 Vetor: (-3.00, 4.00, 2.00, 34.20) Norma: 34.62

4 Cursos - Versão 2

 (++)

Uma universidade particular possui uma tabela de valores de créditos por curso com os campos:

- código do curso (int),
- valor por crédito (float).
- nome do curso (cadeia com no máximo 100 caracteres),

Faça um programa que primeiramente carregue a tabela de cursos acima e depois leia registros de alunos com as seguintes informações:

- nome do aluno (cadeia com no máximo 500 caracteres),
- o código do curso onde o mesmo está matriculado (int), e
- o número de créditos que ele está cursando (int).

Os nomes dos cursos e dos alunos devem ter espaço suficiente apenas para armazenar os caracteres que formam os nomes e o caractere delimitador da string. Para isso, seu programa deve ler inicialmente os nomes (de cursos e de alunos) em um buffer que é uma string com 1000 caracteres. Para cada nome lido deve ser alocado um espaço para ele mais o '\0' no elemento do vetor onde esse nome será armazenado. Calcule a mensalidade a ser paga pelo aluno e imprima as informações em um boleto de pagamento com as seguintes informações: nome do aluno, nome do curso, total de créditos que o aluno cursa, valor do crédito do curso e o valor final da mensalidade a pagar. **Ao final, seu programa deve liberar todos os espaços alocados para armazenar os nomes de cursos e de alunos.**

Entrada

A primeira linha da entrada contém o número n ($5 \leq n \leq 30$) de cursos a universidade. Em seguida há $3n$ linhas contendo as três informações dos n cursos. A próxima linha contém um valor inteiro que corresponde ao número m ($1 \leq m \leq 1000$) de alunos da universidade. Em seguida, há $3m$ linhas, com os três dados dos m alunos.

Saída

A saída é formada por m linhas, cada uma no seguinte formato: Aluno(a): a Curso: b Num. Creditos: c Valor Credito: d Mensalidade: e, onde a e b são cadeias de caracteres, c é um número inteiro e d e e são números em ponto flutuante com duas casas decimais.

Exemplo

Entrada
4 298 234.5 Direito 123 150.00 Eng. Eletrica 452 132.00 Eng. Civil 341 120.00 Farmacia 5 Joao Luiz 298 5 Paula Lima 452 4 Maria 123 6 Luiz Andre 341 3 Antonio Luiz 341 8
Saída
Aluno(a): Joao Luiz Curso: Direito Num. Creditos: 5 Valor Credito: 234.50 Mensalidade: 1172.50 Aluno(a): Paula Lima Curso: Eng. Civil Num. Creditos: 4 Valor Credito: 132.00 Mensalidade: 528.00 Aluno(a): Maria Curso: Eng. Eletrica Num. Creditos: 6 Valor Credito: 150.00 Mensalidade: 900.00 Aluno(a): Luiz Andre Curso: Farmacia Num. Creditos: 3 Valor Credito: 120.00 Mensalidade: 360.00 Aluno(a): Antonio Luiz Curso: Farmacia Num. Creditos: 8 Valor Credito: 120.00 Mensalidade: 960.00

5 Distância entre pontos

 (++)

Faça um programa que leia vários pares de pontos no espaço de quatro dimensões e calcule a distância entre eles. Considere que a distância entre dois pontos no espaço tetradimensional $A(a_u, a_x, a_y, a_z)$ e $B(b_u, b_x, b_y, b_z)$ é calculada como:

$$d(A,B) = |BA| = \sqrt{(a_u - b_u)^2 + (a_x - b_x)^2 + (a_y - b_y)^2 + (a_z - b_z)^2} \tag{2}$$

Voce deve usar uma struct para representar cada ponto.

Entrada

A entrada consiste de várias linhas. A primeira linha apresenta um número de pontos N, com $2 \leq N \leq 1000$. As N linhas seguintes apresentam pontos no espaço na forma u, x, y, z, com u, x, y e z números reais tais que $-1000 \leq x, y, z \leq 1000$. Faça um programa que calcule a distância entre dois pontos consecutivos nesta lista. Note que, com exceção do primeiro e último valor de entrada, todos os pontos serão utilizados duas vezes, uma para o cálculo de distância com o ponto que veio antes na lista e outra para o ponto que veio depois.

Saída

A saída consiste de $(N-1)$ linhas, cada uma contendo a distância entre os pontos com 2 casas decimais após a vírgula. Após a impressão do valor de uma distância, o programa deve imprimir o caractere de quebra de linha.

Exemplo

Entrada
2 4 1 0 1 -1 2 1 2
Saída
5.29

Entrada
4 1 1 5 1 2 -1 3 0 4 2 -1 2 -3 4 2 2
Saída
3.16 5.74 7.87

Entrada
4 15.89 0.7 0.53 0.33 0.45 0.38 0.22 0.11 0 0 0 1 0 0 1 0.5
Saída
15.45 1.09 1.12

6 Ordenação por Data

 (++)

Uma determinada professora quer ordenar seus alunos em ordem crescente de idade. Escreva um programa em C que leia os dados dos alunos, entre eles a data de nascimento, ordene os alunos em ordem crescente de idade. Para isso seu programa deve ter uma função *ComparaDataNasc()* que recebe dois parâmetros. O primeiro corresponde a uma struct (ou um ponteiro para uma struct) que contém o dia, o mês e o ano de nascimento de uma aluno. O segundo parâmetro tem o memo tipo de dado do primeiro e contém a data de nascimento do segundo aluno. Essa função retorna 1 se o primeiro aluno é mais novo ou tem a mesma idade do segundo aluno e retorna zero em caso contrário. Essa função deve ser chamada pela função que ordena os alunos em ordem crescente de idade.

Entrada

A entrada contém apenas um caso de teste. A primeira linha da entrada contém um número inteiro n, ($1 \leq n \leq 30$) que corresponde ao número de alunos da turma. Em seguida há n linhas, contendo cada uma:

- a matrícula de um aluno (int);
- o dia de nascimento de um aluno (int);
- o mês de nascimento de um aluno (int);
- o ano de nascimento de um aluno (int);
- o nome de um aluno (no máximo 200 caracteres);

Saída

A saída é formada por n linhas, cada uma correspondendo ao um aluno, ordenadas em ordem crescente de idade dos alunos. Cada linha deve ter o seguinte formato: Matric.: *m* Nome: *n* Data Nasc.: *dd/mm/aa*, onde *m* é a matricula de um aluno, *n*, o seu nome, e *dd*, *mm* e *aa*, são respectivamente o dia, o mês e o ano do seu nascimento.

Exemplo

Entrada
5 12345 12 07 1978 Felizbina Freitas 23489 11 03 2009 Joao Feliz da Tristeza 98762 05 12 1976 Maria Batista de Souza 34561 11 07 1978 Roberto de Assis 34599 07 05 1976 Luiz Alberto Ferreira
Saída
Matric.: 23489 Nome: Joao Feliz da Tristeza Data Nasc: 11/3/2009 Matric.: 12345 Nome: Felizbina Freitas Data Nasc: 12/7/1978 Matric.: 34561 Nome: Roberto de Assis Data Nasc: 11/7/1978 Matric.: 98762 Nome: Maria Batista de Souza Data Nasc: 5/12/1976 Matric.: 34599 Nome: Luiz Alberto Ferreira Data Nasc: 7/5/1976

7 Soma e Subtração de Polinômios

 (+++)

Faça um programa para somar ou subtrair polinômios

Entrada

A primeira linha da entrada corresponde ao número de casos de teste. Cada caso de teste é formado por várias linhas. A primeira linha contém o caractere '+' ou o caractere '-', indicando se a operação é, respectivamente de soma ou de subtração de polinômios. A próxima linha contém um número inteiro n_1 que indica o número de termos do primeiro polinômio. Em seguida, há n_1 ($n_1 \leq 50$) linhas cada uma contendo um par de números c e e separados entre si por um espaço. O número c é um valor float, com sinal e corresponde ao coeficiente de um termo do polinômio. O valor e é um número inteiro positivo e corresponde ao expoente do termo do polinômio. Após o par n_1. Há um outro número inteiro n_2 ($n_2 \leq 50$) que corresponde ao número de termos do segundo polinômio, seguido, de $n-2$ linhas contendo pares de coeficientes e expoentes, como explicando para o caso do primeiro polinômio. Os pares de cada polinômio estão ordenados em ordem decrescente de expoente.

Saída

Para cada caso de teste o programa deve imprimir uma linha contendo o polinômio resultante escrito no seguinte formato:

$$sc_1X \wedge e_n sc_2X \wedge e_{n-1} \cdot sc_nX \wedge e_1$$

onde $s \in \{'+','-'\}$ é o sinal do coeficiente do polinômio, c_i é o valor do deficiente e e_i é o valor do expoente do termo i. Se durante a operação se soma ou subtração de polinômios um dos termos do polinômio resultante ficar com coeficiente igual a zero, esse termo não deve aparecer no polinômio resultante.

Sugestão

Represente cada termo do polinômio como uma struct com dois campos: *coeficiente* e *expoente*. Represente cada polinômio de entrada e também o polinômio resultante como vetores de structs. Escreva uma função para cada uma das seguintes ações:

- Leitura de um polinômio
- Impressão de um polinômio
- Soma de dois polinômios
- Subtração de dois polinômios

Exemplo

Entrada
2 + 5 -0.5 7 +4.2 4 -3 2 +1 1 -4 0 4 +3 5 +2 4 -1 2 +3 0 + 2 -5 7 -3 0 6 +2 6 +2 5 +3 4 -2.5 3 +3 2 -1.2 1
Saída
$-0.50X \wedge 7 + 3.00X \wedge 5 + 6.20X \wedge 4 - 4.00X \wedge 2 + 1.00X \wedge 1 - 1.00$ $-5.00X \wedge 7 + 2.00X \wedge 6 + 2.00X \wedge 5 + 3.00X \wedge 4 - 2.50X \wedge 3 + 3.00X \wedge 2 - 1.20X \wedge 1 - 3.00$

8 Números Racionais (++++)

 (++++)

Número racional é todo o número que pode ser representado por uma razão (ou fração) entre dois números inteiros. O conjunto dos números racionais (representado por $\mathbb{Q}$) é definido por:

$$\mathbb{Q} = \{\frac{a}{b} \mid a \in \mathbb{Z}; b \in \mathbb{Z}^*\} \tag{3}$$

Em outras palavras, o conjunto dos números racionais é formado por todos os quocientes de números inteiros a e b, em que b é não nulo. O uso da letra "Q"é derivado da palavra inglesa *quotient*, cujo significado é quociente, já que a forma de escrever um número racional é o quociente de dois números inteiros. São exemplos de números racionais:

$$\frac{5}{8}; 1; 2; \frac{1}{3}; \text{-}8; -\frac{2}{5};$$

Faça um programa que defina um novo tipo de dado através de uma estrutura chamada `tRacional`, com os componentes inteiros a e b, conforme a definição anterior. Escreva as seguintes funções para operar sobre o novo tipo:

```
1
2  /**
3   * Calcula o MDC de x e y
4   * @param x
5   * @param y
6   * @return
7   */
8  int MDC(int x, int y);
9
10
11 /**
12  * Recebe dois inteiros a e b e retorna o racional
13  * @param a numerador
14  * @param b denominador
15  * @return
16  */
17 struct tRacional racional(int a, int b);
18
19 /**
20  * Recebe um racional e retorna o seu negativo (-r).
21  * @param r numero racional
22  * @return
23  */
24 struct tRacional negativo(struct tRacional r);
25
26 /**
27  * Recebe dois racionais e retorna a adição de ambos (r1 + r2).
28  * @param r1 fator esquerdo da soma
29  * @param r2 fator direito da soma
30  * @return
31  */
32 struct tRacional soma(struct tRacional r1, struct tRacional r2);
33
34 /**
35  * Recebe dois racionais e retorna o produto de ambos (r1 * r2).
36  * @param r1 primeiro fator do produto
37  * @param r2 segundo fator do produto
38  * @return
```

```
 */
struct tRacional mult(struct tRacional r1, struct tRacional r2);

/**
 * Recebe dois racionais e retorna o quociente de ambos (r1/r2).
 * @param r1 numerador
 * @param r2 denominador
 * @return
 */
struct tRacional div(struct tRacional r1, struct tRacional r2);


/**
 * Recebe um racional e reduz a fração ao máximo.
 * @param r o número racional a ser reduzido
 */
void reduzFracao( struct tRacional * r);
```

Entrada

A entrada consiste de várias linhas no seguinte formato: a b operação a b, onde: $-10000 \leq a \leq 10000$, $0 < b \leq 10000$. E operação será um dos seguintes caracteres: +, -, * ou /. A entrada termina com `EOF`.

Saída

A saída consiste de várias linhas com o resultado da operação sobre os racionais. É necessário reduzir a fração ao máximo. Após a impressão do último resultado quebre uma linha.

Exemplo

Entrada	Saída
1 5 + 2 10 2 3 + 5 7 17 24 - 5 6 8 3 * 4 3 -5 2 * 4 3 8 3 / 4 3 2 3 / 2 5	2 5 29 21 -1 8 32 9 -10 3 2 1 5 3

9 Raízes de equações quadradas (++++)

 (++++)

Uma equação de grau 2 tem o formato $ax^2 + bx + c = 0$, onde a, b, c são os coeficientes da equação e x a variável independente. Faça uma função que calcule, imprima e retorne as as raízes de uma equação de segundo grau. Para isso, crie as estruturas `Complex` e `RaizEqu2`, sendo Complex formada por duas variáveis do tipo **`float`**, representando a parte real e imaginária; e `RaizEqu2`, contendo duas variáveis `Complex`, representando as duas raízes da equação. Você deve implementar as funções:

```
/**
 * Função que calula as raízes de uma equação de segundo grau.
 *
 * @param a coeficiente quadrado
 * @param b coeficiente linear
 * @param c constante
 * @return retorna uma estrutura RaizEqu2 com dois números complexos
 */
struct RaizEqu2 calcula_raiz_equ_2( float a, float b, float c);
```

```
/**
 * Imprime numeros complexos na saída padrão do sistema. A impressão segue o formato
 * a + bi, onde a é a parte real e b a imaginária. Os valores são apresentados
 * somente se forem diferente de zero. No caso de a e b forem zero, o valor 0 é
 * apresentado. Se o valor de b for 1 ou -1, somente o caracter i ou -i é
   apresentado.
 *
 * @param c Numero complexo a ser impresso.
 */
void complex_print(struct Complex c);
```

As soluções das raízes desse tipo de equação são dadas pela fórmula de Bhaskara 4.

$$x_1 = \frac{-b + \sqrt{b^2 - 4ac}}{2a} \quad x_2 = \frac{-b - \sqrt{b^2 - 4ac}}{2a} \tag{4}$$

Entrada

O programa deve ler 3 valores reais, correspondentes aos coeficientes a, b, c.

Saída

O programa deve apresentar as raízes x_1 ex_2 em linhas separadas. Para os valores positivos deve-se omitir o sinal $+$. A parte imaginária deve ser apresentada seguida do caracter i. Os valores iguais a zero devem ser omitidos. Apresenta-se o valor 0 quando a raíz possui a parte real e imaginária zeradas. Todos os valores devem ser apresentados com 2 casas decimais.

Exemplo

Entrada	Saída
1 0 0	x1 = 0.00 x2 = 0.00

Entrada	Saída
1 2 1	x1 = -1.00 x2 = -1.00

Entrada	Saída
1 0 1	x1 = i x2 = -i

Entrada	Saída
1 5 4	x1 = -1.00 x2 = -4.00

Entrada	Saída
1 -2 2	x1 = 1.00+i x2 = 1.00-i