Universidade Federal de Pernambuco

CIn – Centro de Informática

Introdução a Programação – EC

Prof° Adriano Sarmento

Data: 08/03/2013

Data entrega: 21/03/2013

Quinta Lista - IP

Considerações:

- É proibido usar a biblioteca conio.h;
- Leia a lista toda o quanto antes, para evitar más interpretações e muitas dúvidas em cima da hora;
- Envie uma prévia das questões, pelo menos um dia antes da data final da entrega, para o caso de acontecer algum imprevisto;
- A lista é para ser feita individualmente. Qualquer tentativa de cópia acarretará o zeramento da lista de todos os envolvidos;
- Em caso de dúvida, envie email para duvidasip@googlegroups.com.
- Atenção para a liberação da memória no final dos programas, será cobrado que o espaço alocado no decorrer do programa seja totalmente liberado no final do mesmo.

1- Alunos de Engenharia, depois de anos convertendo manualmente vetores de coordenadas cartesianas para coordenadas cilíndricas e para coordenadas esféricas, decidiram pedir ajudas aos seus colegas de EC. Faça um programa que capture do usuário vetores em uma das três representações, e converta para outra representação.

Devem ser criados um tipo estruturado para cada representação de vetor, e devem ser criadas funções de conversão de qualquer representação para qualquer outra representação.

Represente ângulos (em radianos) por **float** e numeros reais por **double**.

Representação cartesiana: (x, y, z)

Representação cilíndrica: (angulo[θ], altura, raio)

Representação esférica: (angulo1 [θ], angulo2[ρ], raio)

Formulas de conversão de cartesiano para cilíndrica:

Θ = $\arcsin(\frac{y}{\sqrt{x^2+y^2}})$

Altura = z

Raio = $\sqrt{x^2+y^2}$

Formulas de conversão de cartesiano para esferica:

Θ = $\arcsin(\frac{y}{\sqrt{x^2+y^2}})$

ρ = $\arccos(\frac{\sqrt{x^2+y^2}}{\sqrt{x^2+y^2+z^2}})$

Altura = $\sqrt{x^2+y^2+z^2}$

Exemplo:

Qual a representação do vetor?

1 – Cartesiana.

2 – Cilíndrica.

3 – Esférica.

2

Qual o vetor?

(3.1415926 , 1 , 1)

Qual a representação desejada?

1 – Cartesiana.

2 – Cilíndrica.

3 – Esférica.

1

Vetor guardado na representação cartesiana. Seu valor é: (-1, 0, 1)

O que deve ser feito?

1 – Novo vetor

2 – Nova representação do vetor atual

3 – Sair

3

Fim do programa.

Obs: Considere pi = 3.1415926 //provavelmente nao será preciso.

Obs2: a biblioteca math.h pode ser usada.

2 - Nessa questão será feita uma playlist e cada música terá nome de no máximo 25 caracteres (para essa questão o nome da música tem que ser alocada dinamicamente) seguido de seu respectivo tempo de execução (sempre será um número inteiro), onde inicialmente terá um menu que será assim:

1- Inserir musica

2-Remover musica

3- Tocar musica

4-Sair

Na função inserir música será adicionado uma música na playlist e quando inserido deve ser impresso “Musica inserida”. Na função remover será removido a última música e caso essa função tenha êxito deve ser impresso “Musica removida”, caso contrário deve ser impresso “Musica nao consta na playlist”. Na função tocar música caso a música seja encontrada deve ser impresso o nome e o tempo de execução da música, caso contrário deve ser impresso “Musica nao consta na playlist”. Quando for selecionada a opção de sair deve ser impresso todas as músicas seguidas de seus tempos de execução, por exemplo,

Madri – 5

DoLadodeCa – 7

Nesse exemplo só tinham duas músicas na playlist.

3 - Sua tarefa é simular um terminal UNIX, em particular as operações com arquivos e pastas.

assumindo que uma pasta é uma estrutura composta de:

nome(string)
vetor de até 50 pastas
vetor de até 50 arquivos
ponteiro para a pasta acima

toda pasta tem duas pastas padrão: "..", que significa a pasta acima dela, e ".", que significa a própria pasta. Ambas nao devem ser incluidas na lista de pastas.
Na pasta raíz, o ponteiro para a pasta acima é NULL.

Um arquivo é uma estrutura composta de:
nome(string)
conteúdo(string)

Seu programa irá receber vários comandos em sequência.

ao esperar um novo comando, deve ser impresso a pasta atual que o usuário se encontra, seguida de '$' a pasta inicial é a /home

ls -> lista todos os arquivos e pastas da pasta atual, diferenciando-os com D (diretório) e F(arquivo)
ex:

```
/home$ ls
..          D
.           D
pasta1      D
pasta2      D
arquivo1    F
arquivo2    F
/home$
```

cd PASTA -> navega para a pasta PASTA, se PASTA existir, ou avisa que não foi possível entrar na pasta. ao navegar para "..", o usuário ira navegar para a pasta acima. navegar para "." muda nada.
ex:

```
/home$ cd pasta2
/home/pasta2$
```

mkdir PASTA -> cria a pasta PASTA na pasta atual, se possível. se existir uma pasta ou arquivo de mesmo nome, o usuário deverá ser avisado disso.
ex:

```
/home$ ls
..          D
.           D
pasta1      D
```

```
pasta2      D
arquivo1    F
arquivo2    F
/home$ mkdir pasta3
/home$ ls
..          D
.           D
pasta1      D
pasta2      D
pasta3      D
arquivo1    F
arquivo2    F
/home$
```

touch ARQUIVO TEXTO -> cria o arquivo ARQUIVO no diretório atual, com conteudo TEXTO. se ja existir arquivo ou pasta com esse nome, isso deve ser avisado
ex:

```
/home$ ls
..          D
.           D
pasta1      D
pasta2      D
arquivo1    F
arquivo2    F
/home$ touch arquivo3 abcdef
/home$ ls
..          D
.           D
pasta1      D
pasta2      D
arquivo1    F
arquivo2    F
arquivo3    F
/home$
```

cat arquivo -> imprime na tela o conteúdo de arquivo. se o arquivo nao existir, isso deve ser avisado.

```
/home$ cat arquivo3
abcdef
/home$
```

locate NOME -> procura recursivamente algum arquivo ou pasta de nome NOME, e diz onde está. se nada for encontrado, nada é impresso.

```
/home$ ls
..          D
.           D
pasta1      D
```

```
pasta2      D
arquivo1    F
arquivo2    F
/home$ locate pasta1
/home/pasta1          D
/home$
```

rm NOME -> deleta a pasta ou arquivo NOME, e se for pasta também deleta tudo dentro dela.

todos os comandos podem usar tanto o nome do arquivo dentro da pasta, ou usando o nome completo do arquivo.
ex:

```
/home$ ls
..              D
.               D
pasta1      D
pasta2      D
arquivo1    F
arquivo2    F
/home$ rm pasta1
/home$ ls
..              D
.               D
pasta2      D
arquivo1    F
arquivo2    F
/home$ rm /home/pasta2
/home$ ls
..              D
.               D
arquivo1    F
arquivo2    F
```

Para simplificar a implementação, é permitido guardar globalmente um ponteiro para a pasta raíz e um ponteiro para a pasta atual.

4 - Olá. Você está no ano de 2112. Nestes tempos precários os programadores não querem mais saber de programar com qualidade. Com o domínio da Nuvem, todos agora usam softwares prontos. Pior ainda, que rodam na JVM. Porém nós, da UNATCO, não podemos confiar nossos dados a servidores na rede, dos quais não teremos controle total. Por isto, nós necessitamos de um programador jovem e capaz de escrever programas de forma criativa e confiável, para que possamos guardar nossos

dados com segurança.
Escreva um programa que cria um banco de dados e é capaz de guardar, obter, e apagar elementos dentro dele. Todo o conteúdo pode e deve ser guardado em memória volátil, pois temos garantias que nossos computadores não serão desligados.

É parte de nossa especificação que seu programa deva ter implementado as seguintes funções internas:
create(**password**) → Retorna um novo banco de dados. O **password** será uma senha de caracteres alfa-numéricos dada no momento da criação do banco de dados, passada como argumento nesta função. Confiamos no programador o suficiente para que ele utilize o algorítmo que quiser para a encriptação da senha, desde que o valor encriptado seja um valor inteiro (por exemplo, a senha IP2012 virasse 33), e que toda vez que esta senha sofre encriptação, retorne o mesmo valor.

destroy(database, password) → Deverá existir uma função que irá deletar o banco de dados, liberando toda a memória necessária para o funcionamento deste. Retorna o valor booleano 1, caso o banco de dados tenha sido apagado com sucesso, ou 0, caso tenha ocorrido algum erro. Nesta e nas outras funções deve se verificar se a senha é correta.

write(key, element, database, password) → Escreve dentro do banco de dados uma nova chave com o elemento dado. Estas chaves devem ser guardadas por referência em um vetore dinâmico que deve ser ordeado por ordem crescente de **key**. Estes vetores devem ser terminados com uma key NULL. Atenção: *É proibido o uso de algorítmos de ordenação.* A ordenação deve ser feita na inserção, ou seja, o elemento deve ser inserido já no local correto em cada vetor para manter a ordem. Não se preocupe. Não haverão **keys** repetidas.

erase(key, database, password) → Deleta a chave dada, incluindo o seu conteúdo (ou seja, o elemento). Fica a critério do programador a reorganização do vetor, visando o mínimo de disperdício de memória.

read(key, database, password) → Deve imprimir na tela todas as informações do elemento correspondente à chave guardada.

printDb(database, password) → Deve imprimir na ordem inversa das chaves.
Nota: como você não tem guardado em lugar nenhum o tamanho deste vetor, as impressões inversas devem acontecer através de recursão.
Deverá ser impresso na tela todas as informações guardadas na estrutura **element**. Incluindo suas estruturas internas.

Todas as estruturas usadas no programa *deverão ser passadas por referência*. Como todo o banco de dados ficará armazenado na memória, não queremos disperdício desta devido a cópias lentas e duplicações de objetos na pilha.

Especificação das estruturas e nomenclaturas:
Os elementos aqui mostrados são obrigatórios. Mas fica a critério do programador acrescentar novos elementos que ele julgue necessário.

database – Deve ser implementada guardando um ponteiro para um vetor dinâmico, e um inteiro, responsável pelo *valor encriptado* da senha (**password**). Como dito anteriormente, o vetor deverá ser responsável por guardar ponteiros para as estruturas de dados. No caso, ponteiros para **key**.

element – Deverá ser uma estrutura contendo as seguintes informações:
subject_name: Nome do registrado. Seja ele **máquina** ou **humano**. Ciborgues se enquadram em **máquina**. É com este elemento que deve se levar em conta na ordenação por nome dos vetores.
subject_type_enum: Enum do tipo do registrado. 0 para humano ou 1 para máquina.
subject_characteristics_union: Como o nome diz, deverá ser um union de dois ponteiros para estruturas. Uma é a estrutura de características intrínsecas de humanos, e a outra é a estrutura de características intrínsecas de máquinas.

key – Estrutura simples que irá guardar um número identificador interno, chamado criativamente de **key_num**, e um ponteiro para uma estrutura do tipo **element**.

human – Estrutura de características intrínsecas de humanos. Nela deve possuir a idade, seu sexo(string de 9 caracteres) e a quantidade de tempo que ele demora para fazer macarrão instantâneo, com precisão da ordem de 10^(-3)seg.

machine – Estrutura de características intrínsecas a máquinas. Nela deve possuir o número de série, data de fabricação (ou data de nascimento, no caso de ciborgues) usando uma string no formato dd/mm/aaaa, e sua função na empresa (uma string de 20 caracteres).

Deverá, além destas características, ser fornecido uma interface de menu, com as seguintes opções:
1 – Create: Cria novo banco de dados. Deve ser permitido ao usuário criar mais de um banco de dados, ficando a critério do programador uma forma de identificar cada um.
2 – Insert: Insere um novo elemento no banco de dados escolhido.
3 – Delete: Remove um elemento do banco de dados escolhido, através da **key** dada.
4 – Read: Lê um elemento do banco de dados escolhido, seja por nome, ou por chave.
5 – Print: Deve ser fornecido um sub-menu, com uma das 2 formas de impressão a ser escolhida.
6 – Destroy: Destrói o banco de dados escolhido, junto com todas as suas informações.

Exemplo:
#*menu de opções*
>>2
#Digite o nome do banco de dados:
>>teste
#Banco de dados inexistente.

#*menu de opções*
>>1
#Digite o nome do novo banco de dados:
>>teste
#Digite a senha do novo banco de dados:
>>teste
#Banco de dados criado com sucesso!

#*menu de opções*
>>2
#Digite o nome do banco de dados:

>>teste
#Digite a senha do banco de dados:
>>teste
#Digite o nome do registrado:
>>JC Denton
#Digite a key do registrado:
>>155
#Digite o tipo do registrado:
>>1
#Digite a idade do registrado:
>>83
#Digite o sexo do registrado:
>>masculino
#Digite a quantidade de tempo que ele demora para fazer macarrao instantaneo:
>>58.322
#Inserção de elemento no Banco de Dados completa!

#*menu de opções*
>>4
#Digite 1 para busca através de key ou 2 para busca através de element:
>>2
#Digite o nome do banco de dados:
>>teste
#Digite a senha do banco de dados:
>>teste
#Digite o nome do element:
>>JC Denton
#115

#*menu de opções*
>>4
#Digite 1 para busca através de key ou 2 para busca através de element:
>>1
#Digite o nome do banco de dados:
>>teste
#Digite a senha do banco de dados:
>>teste
#Digite o nome do element:
>>115
#Nome: JC Denton
#Tipo: Humano
#Idade: 83
#Sexo: masculino
#Velocidade com que faz macarrao instantaneo: 58.322 sec

#*menu de opções*
>>6
#Digite o nome do banco de dados:

>>teste
#Digite a senha do banco de dados:
>>teste
#Queima de arquivo concluida com sucesso!

OBS: A inserção e a remoção sempre ocorrerão na última posição.

5 -Professores do glorioso departamento de Física da UFPE precisam de ajuda com simulações cinematicas, pois eles são muito bons em física, porém nem tanto em programação e você (dedicado aluno de IP) pode ajuda-los. Lembre-se sempre que eles são tão exigentes com os softwares que usam quanto nas correções de provas. Pois bem, vamos ao programa pedido por eles.

Sendo um vetor3d uma estrutura representado por 3 floats, um para cada coordenada de um sistema cartesiano tridimensional. E sendo uma flecha pontual (não possui massa, comprimento ou espessura) outra estrutura representado por dois vetores (vetor3d), um de posição e outro de velocidade, e uma string de até 20 caracteres.

Faça um sistema cinemático que movimenta no tempo pedido até 10 flechas pontuais com ou sem a influencia da gravidade e checando colisões. A distancia é medidas em metros, o tempo em segundos, a velocidade em metros por segundo, e a aceleração (gravidade) em metros por segundo ao quadrado. Guarde as flechas em um vetor de ponteiros para flecha, inicializando com todos os campos como NULL, representando assim que nenhuma daquelas flechas já foi alocad

Seu sistema deve iniciar em 0 segundo e ter quatro opções de menu. Antes de cada menu diga o status atual do sistema: O tempo atual, se possui gravidade e qual seu vetor e modulo, a atual posição de cada flecha no sistema e seu vetore velocidade seguido do modulo de sua velocidade. As opções do menu são :

1) Adicixonar Flecha
2) Remover Flecha
3) Percorrer tempo
4) Sair

Na primeira opção peça ao usuário em forma de vetor, sua posição inicial, sua velocidade inicial e o nome a ser atribuido aquela fecha. Depois a adiciona ao sistema.

A segunda remove uma flechas já adicionadas do sistema, sem modificar a posição das demais flechas no vetor.

A terceira pergunta quanto tempo se quer que o sistema ande, e se calcula onde cada flecha estará depois desse tempo e a velocidade que cada flecha terá depois desse tempo. Colisões de flecha com flecha devem ser analisadas ao longo do trajeto, em caso de colisão a velocidade das flechas envolvidas deve ser trocada entre as duas flechas colididas, sendo assim um choque perfeitamente elastico com massas iguais. Não precisa considerar o caso de 3 ou mais flechas colidirem ao mesmo tempo entre si.

Cada coordenada de espaço e velocidade deve ser calculada como um movimento retilíneo uniformemente variado independente das demais coordenadas, usando se assim das formulas do espaço pelo tempo percorrido, e da velocidade pelo tempo percorrido. Outras formulas de cinemática vetorial podem ser usadas, desde que explicadas nos comentários.

Espaço $S(c) = So(c) + t * Vo(c) + t ^2 * g(c) / 2$

Velocidade $V(c) = Vo(c) + g(c) * t$

Sendo:
S(c) o espaço na coordenada c
Vo(c) a velocidade inicial na coordenada c
So(c) o espaço inicial na coordenada c
g(C) a gravidade na coordenada c
t o tempo percorrido

Cuidado com colisões, o seu sistema deve a partir dessas formulas, ou outras de cinemática, checar se durante o trajeto não houve um colisão entre qualquer par de flechas, colidir é ter no mesmo instante todas as três coordenadas exatamente iguais.

Não resolva usando laços que incrementem os vetores posição e velocidade e vão checando colisões, isso é falho e custoso por estar trabalhando com floats e apesar de não dar valores idênticos, pode na realidade ser uma colisão que seu laço não foi ínfimo o suficiente para reparar. Pense em formulas matemáticas que definam se houve ou não colisão, e em qual local, veja todas as combinações entre as flechas do sistema e lembre-se que cada colisão modifica seu sistema.

Após um choque entre flechas, as mesmas podem ainda ter velocidades e devem continuar se movimentando seguindo as regras já ditas.

Após calcular todos os deslocamentos e colisões acontecidas no intervalo, retorne ao menu exibindo o atual status do sistema.

A quarta simplesmente fecha ao programa. Ao fim de cada item que não seja a quinta retorne ao menu.

No terceiro item do menu se houver colisões diga quais colisões ocorreram, os instantes, e as flechas envolvidas. Siga a ordem cronologica das colisões.

Se não houve diga que não houve.

Exemplo:

Gravidade (S/N)?
>> S
Eixo-x da Gravidade em m/s^2 ?
>> 0
Eixo-y da Gravidade em m/s^2 ?
>> 0
Eixo-y da Gravidade em m/s^2 ?
>> -10

Tempo: 0.00s
Gravidade: (0.00 , 0.00 , - 10.00) | 10.00 | m/s^2
Não há flechas

Menu :
1) Adicionar Flechas
2) Remover Flechas
3) Percorrer tempo
4) Sair
>>1

Coordenada X de localização?
>> 1
Coordenada Y de localização?
>> 2
Coordenada Z de localização?
>> 40
Coordenada X de velocidade inicial?
>> 1
Coordenada Y de velocidade inicial?
>> 1
Coordenada Z de velocidade inicial?
>> 0
Nome?
>> Flecha A

Tempo: 0s
Gravidade: (0.00 , 0.00 , - 10.00) | 10.00 | m/s^2

Flecha A: (1.00 , 2.00 , 40.00)m a (1.00 , 1.00 , 0.00) | 1.14 | m/s

Menu:
1) Adicionar Flechas
2) Remover Flechas
3) Percorrer tempo
4) Sair
>> 3

Quanto tempo será percorrido em segundos?
>>2.0

Gravidade: (0.00 , 0.00 , - 10.00) | 10.00 | m/s^2

Tempo: 2.00s
Flecha A: (3.00 , 4.00 , 20.00)m a (1.00 , 1.00 , -20.00) |20,05| m/s

Não houve colisão entre flechas

Menu:
1) Adicionar Flecha
2) Remover Flechas
3) Percorrer tempo
4) Sair
>> 1

Coordenada X de localização?
>> 5
Coordenada Y de localização?
>> 6
Coordenada Z de localização?
>> -20
Coordenada X de velocidade inicial?
>> -1
Coordenada Y de velocidade inicial?
>> -1
Coordenada Z de velocidade inicial?
>> 20
Nome?
>> Flecha B

Tempo: 2.00s

Gravidade: (0.00 , 0.00 , - 10.00) | 10.00 | m/s^2

Flecha A: (3.00 , 4.00 , 20.00)m a (1.00 , 1.00 , -20.00) |20,05| m/s
Flecha B: (5.00 , 6.00 ,-20.00)m a (-1.00, -1.00 , 20.00) | 20.00 | m/s

Menu:
1) Adicionar Flecha
2) Remover Flechas
3) Percorrer tempo
4) Sair
>>3

Quanto tempo será percorrido ao todo em segundos?
>>2.0

/*Não sai na tela , é apenas um comentário do que esta ocorrendo no sistema.
No instante 3.0s, haveu uma colisão. Isso modifica o sistema.

Gravidade: (0.00 , 0.00 , - 10.00) | 10.00 | m/s^2

Tempo: 3.00s

Flecha A: (4.00 , 5.00 , -5.00)m a (1.00 ,1.00 , -30.00) |--| m/s
Flecha B: (4.00 , 5.00 , -5.00)m a (1.00, 1.00 , 10.00) | -- | m/s

colisão, tratando a velocidade:

Flecha A: (4.00 , 5.00 , -5.00)m a (0.00 , 0.00 , 10.00) | 10.00 | m/s
Flecha B: (4.00 , 5.00 , -5.00)m a (0.00, 0.00 , -30.00) | 10.00 | m/s
****/***

Tempo: 4.00s
Gravidade: (0.00 , 0.00 , - 10.00) | 10.00 | m/s^2

Flecha A: (4.00 , 5.00 , 0.00)m a (0.00 , 0.00 , 0.00) |20.00 | m/s
Flecha B: (4.00 , 5.00 , -40.00)m a (0.00 , 0.00 , -40.00) |20.00 | m/s

No instante 3.00 houve colisão entre as flechas : A , B

Menu:
1) Adicionar Flecha

2) Remover Flechas
3) Percorrer tempo
4) Sair
>>4
//Programa encerrado
//Se flechas forem removidas, mantenha os nomes das que restaram.

Lembre-se de desalocar tudo o que foi alocado.

As estruturas citadas no texto devem ser usadas com citadas, vetor3d é a mesma estrutura em gravidade assim como nos dois subcampos da estrutura fecha.

Cuide da interatividade com úsuario.

Dica 1 : Trate a colisão como um problema entre 2 flechas, fazendo uma função que diz se haverá ou não colisão entre as 2 flechas e em que tempo. Chame essa função para todos os pares de flechas (não repetidamente) e veja sempre a primeira colisão, preste atenção que uma colisão depois do tempo pedido de deslocamento de fato não ocorre.

Dica 2 : A gravidade influência todas as flechas igualmente, logo para colisões se você colocar o referencial em uma das flechas entre elas não haverá aceleração, ou seja, a formula da colisão deste problema não depende da gravidade.