

Programação 2022/23

LEI, LEI-PL, LEI-CE

Guião laboratorial n.º 3

Estruturas Dinâmicas

Tópicos da matéria:

- Listas ligadas e outras estruturas dinâmicas
- Operações com estruturas dinâmicas

Bibliografia:

K. N. King, *C Programming: A Modern Approach*: Capítulo 17.

A – Listas Ligadas Simples

1. Pretende-se utilizar uma lista ligada simples para armazenar informação sobre o índice de massa corporal de um conjunto de indivíduos. A estrutura a utilizar para armazenar a informação de cada pessoa é a seguinte:

```
typedef struct pessoa no, *pno;
struct pessoa{
    char nome[100];
    int id;
    float peso, altura;
    pno prox;
};
```

O índice de massa corporal é obtido através da seguinte expressão: $imc = peso/altura^2$, em que o peso é especificado em kg e a altura em metros.

a) Crie um ponteiro de lista na função *main()*.

b) Escreva uma função que adicione os dados de uma nova pessoa à lista. O nome, identificador numérico, peso e altura dessa pessoa devem ser obtidos do utilizador. A função deve garantir que o id indicado é único e não se encontra atribuído a outra pessoa que já esteja na lista. A nova pessoa deve ser adicionada ao final da lista ligada.

c) Escreva uma função que mostre na consola a informação completa de todas as pessoas que se encontram na lista ligada, incluindo o seu imc.

d) Escreva uma função que atualize o peso de uma das pessoas que se encontra na lista. O id da pessoa a considerar e o seu novo peso são 2 dos parâmetros da função.

Programação 2022/23

LEI, LEI-PL, LEI-CE

e) Escreva uma função que elimine uma pessoa da lista ligada. O id da pessoa a eliminar é um dos parâmetros da função. O nó retirado da lista deve ser libertado.

f) Escreva uma função que elimine todos os nós da lista ligada.

g) Escreva uma função que elimine da lista todas as pessoas com um imc superior a um determinado limite. O valor limite a considerar é um dos parâmetros da função. Todos os nós retirados da lista devem ser libertados.

h) Escreva uma função que elimine os K últimos nós da lista ligada. O valor K é um dos parâmetros da função. Se a lista tiver menos do que K elementos, a função não efetua nenhuma alteração.

2. Pretende-se alterar a gestão da lista ligada definida na questão 1, por forma a que as pessoas na lista passem a estar ordenadas por imc crescente. Altere o que for necessário nas funções implementadas na questão 1, para garantir que a informação passa a estar organizada desta forma.

3. Um consultório médico pretende controlar o atendimento de sucessivos pacientes de forma a respeitar a ordem de chegada. Pretende-se implementar um programa que faça a gestão de uma lista ligada em que cada nó corresponde a um dos pacientes que se encontra à espera. A lista ligada funcionará como uma fila de espera em que o primeiro a chegar será o primeiro a ser atendido (estrutura de dados do tipo FIFO, i.e., *First In First Out*).

a) Considerando que cada paciente é completamente identificado pelo seu nome, defina a(s) estrutura(s) que considerar necessária(s) para a resolução do problema.

b) Desenvolva uma função que **adicione um novo paciente ao final da fila de espera**. O nome do paciente que acabou de chegar deve ser introduzido pelo utilizador.

c) Desenvolva uma função que permita **visualizar toda a fila de espera**.

d) Desenvolva uma função que implemente o atendimento de um novo paciente. A função deve **retirar o elemento que está no início da fila de espera** e escrever o seu nome no monitor.

e) Desenvolva uma função que permita retirar da fila um paciente que tenha desistido de esperar. A função deve obter o nome do paciente através do utilizador. Esta operação corresponde a **eliminar um dos nós da lista**.

4. Considerando o exercício anterior, nem todos os pacientes necessitam de ser atendidos com a mesma urgência. Considere que existem três graus de urgência: grávidas (nível 1), idosos e crianças (nível 2) e restantes pacientes (nível 3). Só serão atendidos pacientes do nível 2 se não

Programação 2022/23

LEI, LEI-PL, LEI-CE

existirem pacientes do nível 1 e só serão atendidos pacientes do nível 3 se não existirem pacientes dos níveis 1 e 2. Pretende-se que modifique a função de inserção de um novo paciente de modo a ter em conta estes três níveis de prioridade. A fila de espera deve continuar a ser única e a função de atendimento não deve ser alterada (o próximo paciente a ser atendido é o que se encontra no início da fila). Para responder a esta questão deve efetuar as alterações necessárias na definição da(s) estrutura(s) de dados.

Sugestão: Para encontrar o ponto de inserção de um novo paciente, a função deve procurar o primeiro nó da lista que tenha urgência menor (por exemplo, se o novo paciente for de nível 1, então a função deve procurar o primeiro nó que tenha nível 2 ou superior). A inserção do novo paciente é feita imediatamente antes deste ponto.

5. Considere as seguintes definições:

```
typedef struct linha no, *pno;
struct linha{
    char st[80];
    pno prox;
};
```

Considere que existem duas listas ligadas distintas constituídas por elementos do tipo `no`. Em cada uma das listas os elementos estão ordenados alfabeticamente (considerando o campo `st`). Pretende-se agrupar os elementos das duas listas numa única lista que deve permanecer ordenada. Escreva uma função em C que receba os dois ponteiros para o início de cada uma das listas e devolva o ponteiro para o início da lista com todos os elementos. Não deve ser requisitado espaço para os elementos da nova lista (o espaço ocupado pelos nós das duas listas iniciais deve ser aproveitado).

Programação 2022/23

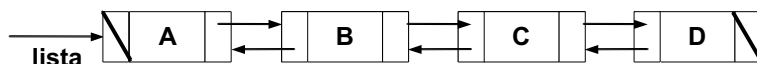
LEI, LEI-PL, LEI-CE

B – Outros Tipos de Listas Ligadas

6. Considere as seguintes definições:

```
typedef struct dados no, *pno;  
struct dados{  
    int val;  
    pno prev, prox;  
};
```

A estrutura `struct dados` permite criar uma lista duplamente ligada. Numa estrutura dinâmica deste tipo, cada nó tem dois ponteiros: um (`prev`) aponta para o elemento que está imediatamente antes e o outro (`prox`) aponta para o elemento que está imediatamente a seguir.



O ponteiro `lista` (variável local da função `main()`) aponta para o primeiro elemento da lista duplamente ligada. Considerando que os nós da lista devem estar ordenados pelo valor armazenado no campo `val`, escreva funções em C que efetuem as seguintes operações:

- a) Adicionar um novo elemento à lista. A função recebe como argumentos um ponteiro para o início da lista e o novo valor a adicionar. Devolve um ponteiro para o início da lista modificada.
- b) Apresentar uma listagem completa dos valores armazenados na lista. A função recebe como argumento um ponteiro para o início da lista
- c) Eliminar um elemento da lista. A função recebe como argumentos um ponteiro para o início da lista e o valor a eliminar. Devolve um ponteiro para o início da lista modificada.
- d) Apresentar os elementos que sejam iguais à média dos seus vizinhos (os vizinhos de um nó são os elementos que se encontram imediatamente antes e depois dele). A função recebe como argumento um ponteiro para o início da lista.

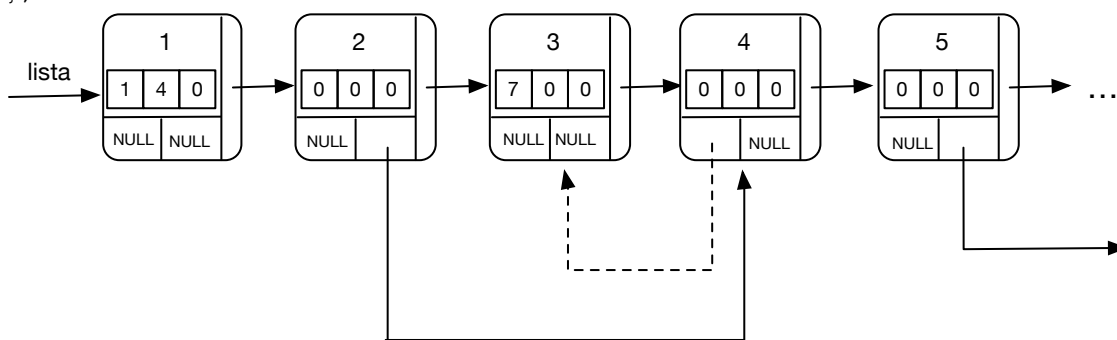
7. Uma estrutura dinâmica armazena um jogo de tabuleiro do tipo jogo da glória. Neste jogo, vários jogadores (3, no máximo) lançam alternadamente um dado para ir avançando nas casas de um tabuleiro. Ganha quem atingir primeiro a última posição. Em algumas casas existem escadas: um jogador que atinja uma escada durante uma jogada, avança para uma casa mais à frente. Em outras posições existem cobras que fazem um jogador recuar algumas casas. Nenhuma posição tem simultaneamente uma escada e uma cobra e não existem cobras nem escadas nas primeira e

Programação 2022/23

LEI, LEI-PL, LEI-CE

última posições do tabuleiro. A estrutura seguinte define uma posição, i.e., uma casa do tabuleiro, e permite criar a estrutura dinâmica completa para jogar o jogo:

```
typedef struct posicao no, *pno;
struct posicao{
    int index;      // Índice da posição do tabuleiro (a 1ª tem índice 1)
    int joga[3];    // Ids dos jogadores que estão nesta posição
    pno cobra, escada; // cobras e escadas existentes nesta posição
    pno prox;
};
```



A estrutura dinâmica exemplificada em cima mostra as primeiras 5 posições de um tabuleiro. São casas sequenciais identificadas pelo seu índice e, em cada casa, existem 2 ponteiros que assinalam a existência de eventuais cobras ou escadas. Se algum destes ponteiros estiver a NULL, não existe ligação deste tipo. Caso contrário apontam para a posição para onde a cobra ou escada levam. No exemplo da figura, a posição 2 tem uma escada para a posição 4. Por sua vez a posição 4 tem uma cobra para a posição 3. Em cada uma das posições do tabuleiro, a tabela *joga* assinala que jogadores se encontram nesse local. Cada jogador tem um identificador inteiro positivo para esse efeito. No exemplo, os jogadores 1 e 4 estão na posição 1 e o jogador 7 está na posição 3. A tabela *joga* serve apenas para identificar quais os jogadores que se encontram numa determinada posição, pelo que a ordem pela qual os identificadores estão armazenados não é relevante ($\{1, 4, 0\}$ é equivalente a $\{4, 0, 1\}$).

a) Escreva uma função em C que mostre em que posição se encontram os jogadores no tabuleiro. A função recebe como argumento um ponteiro para o início da estrutura dinâmica

b) Escreva uma função em C que efetue uma jogada. A função tem o seguinte protótipo:

```
int jogada(pno lista, int totPos, int idJog, int dado);
```

Recebe um ponteiro para o início da estrutura dinâmica, o número de casas do tabuleiro (o número de nós da lista), a identificação do jogador que está a fazer a jogada e o valor que saiu no dado. Deve atualizar a estrutura dinâmica movendo o jogador para a nova posição. Devolve a casa em que ele ficar depois da jogada. Se o jogador indicado não existir, a função devolve 0.

Programação 2022/23

LEI, LEI-PL, LEI-CE

Caso a jogada o faça ultrapassar o limite do tabuleiro, deve ficar colocado na última casa desse mesmo tabuleiro. Por exemplo: se o jogador 4 tiver 1 no dado, deve sair da posição 1 e terminar a jogada na posição 3.

c) Escreva uma função em C que faça uma cópia de uma estrutura dinâmica deste tipo. A função recebe um ponteiro para o início da estrutura dinâmica e o número de casas do tabuleiro. Devolve um ponteiro para uma nova estrutura dinâmica que seja uma cópia da recebida por parâmetro. A estrutura dinâmica original não deve ser modificada.

8. Uma lista circular pode ser vista como uma lista ligada em que o último elemento aponta para o primeiro (embora neste tipo de listas não façam muito sentido as referências ao primeiro e ao último elemento).

Considere que se pretende representar através de uma lista circular um conjunto de pessoas que estão à volta de uma mesa redonda. Cada nó da lista terá um campo para armazenar o nome da pessoa e um ponteiro para o elemento relativo à pessoa que está à sua esquerda.

a) Efetue as definições necessárias de modo a ser possível implementar a estrutura de dados descrita.

b) Escreva uma função em C que devolva o número de pessoas que estão à volta da mesa. A função recebe como argumento um ponteiro para um dos elementos da lista.

c) Escreva uma função em C que devolva o número de lugares que existem entre duas pessoas. A função recebe como argumentos 2 ponteiros (um para cada uma das pessoas a considerar).

d) Escreva uma função em C que permita sentar a primeira pessoa à mesa. A função recebe como argumento o nome dessa pessoa e, no final, devolve um ponteiro para o nó da lista que acabou de ser inserido.

e) Escreva uma função em C que permita inserir uma nova pessoa na lista circular. A função recebe como argumentos, um ponteiro para um dos elementos da lista, o nome da nova pessoa e o nome da pessoa que vai ficar sentada à sua esquerda (alguém que já deve fazer parte da lista). Se essa pessoa não existir, a inserção fica sem efeito.

f) Escreva uma função em C que permita mudar alguém de lugar. A função recebe como argumentos, um ponteiro para um dos elementos da lista, o nome da pessoa que deseja mudar de lugar e o seu deslocamento (número de posições que essa pessoa deseja ir para a esquerda).

Programação 2022/23

LEI, LEI-PL, LEI-CE

9. Considere as seguintes definições:

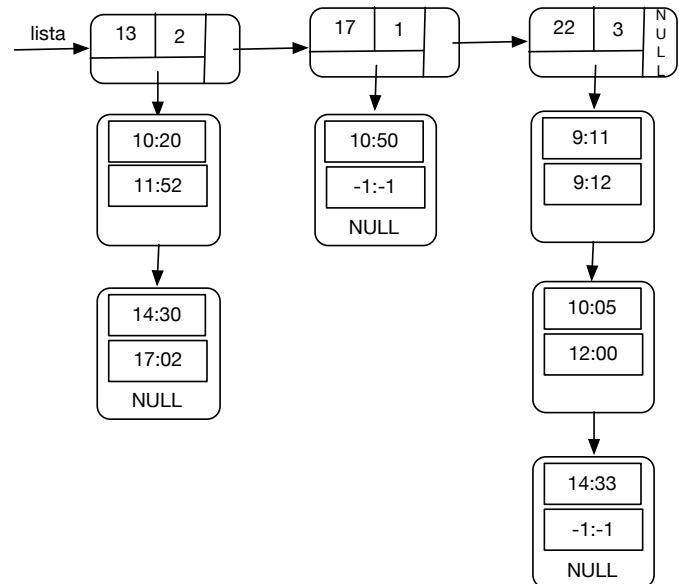
```
typedef struct tipoA cliente, *pCliente;
typedef struct tipoB acesso, *pAcesso;
typedef struct {int h, m;} hora;

struct tipoA{
    int id;                // Identificador único
    int contador;          // Número de utilizações nesse dia
    pAcesso lista;         // Ponteiro para a lista de acessos
    pCliente prox;         // Ponteiro para o próximo cliente
};

struct tipoB{
    hora in, out;          // Horas de entrada e saída
    pAcesso prox;
};
```

Um parque de estacionamento armazena numa estrutura dinâmica informação sobre a utilização dos seus serviços por parte dos clientes registados. **Esta informação diz respeito a um único dia.**

Existe uma lista ligada principal com estruturas do tipo *cliente* contendo informação sobre os clientes registados no parque. De cada nó da lista principal sai uma lista secundária com elementos do tipo *acesso* contendo informação sobre os acessos desse cliente ao longo do dia. Cada nó de acesso regista a hora em que se processou a entrada e a posterior saída do parque. As listas secundárias estão ordenadas por hora de utilização, ou seja, novas utilizações estão sempre no final desta lista. Caso o cliente ainda se encontre no parque, a hora de saída do último nó tem o valor -1:-1. Estas são consideradas utilizações em aberto.



a) Escreva uma função em C que indique quantos clientes estão nesse momento dentro do parque. A função recebe como parâmetro um ponteiro para o início da estrutura dinâmica e devolve o valor contabilizado

Programação 2022/23

LEI, LEI-PL, LEI-CE

b) Escreva uma função em C que indique qual o cliente que já passou mais minutos no parque no dia atual. Na contabilização não devem ser consideradas utilizações em aberto, ou seja, não devem ser consideradas as utilizações em que o cliente ainda se encontre no parque. A função recebe como parâmetro um ponteiro para o início da estrutura dinâmica e devolve o identificador numérico do cliente. Caso não exista nenhum cliente com utilizações completas, a função devolve -1.

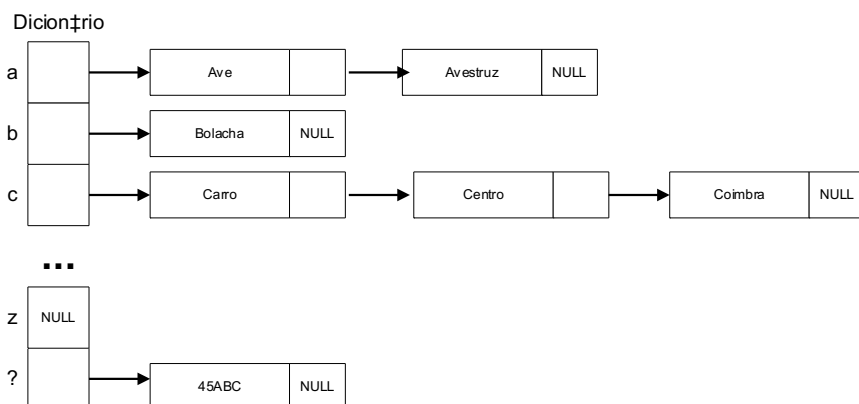
c) Escreva uma função em C que elimine todas as utilizações em aberto que se encontrem nas listas secundárias da estrutura dinâmica. Os espaços retirados da estrutura dinâmica devem ser eliminados e devem ser atualizados os contadores da lista principal. A função recebe como parâmetro um ponteiro para o início da estrutura dinâmica.

d) A função seguinte processa uma ativação de cancela de acesso ao parque:

```
pcliente acessoParque(pcliente lista, int id, hora x);
```

Indica que o cliente com identificador *id* ativou a cancela na hora *x*. Pode corresponder a uma entrada ou a uma saída e pode ser a primeira vez (ou não) que esse cliente usa o parque nesse dia. A função recebe o ponteiro para o início da estrutura dinâmica como um dos seus parâmetros e deve fazer a atualização necessária: coloca a hora de saída se este cliente tiver a última utilização em aberto ou adiciona uma nova utilização ao final da lista especificando a sua hora de entrada. Devolve a estrutura dinâmica atualizada.

10. Pretende-se encontrar uma representação dinâmica para um dicionário de palavras que possam surgir num texto.



Uma possibilidade é ter um conjunto de 27 listas ligadas (uma para cada letra do alfabeto mais uma para palavras que não tenham início em caracteres do alfabeto) em que vamos armazenando as palavras que vão surgindo. Cada palavra pertencente ao texto tem um nó numa das listas (a lista correspondente ao primeiro carácter da palavra). Além disso, cada palavra tem associado um contador inteiro que indica quantas vezes ela já surgiu no texto.

Programação 2022/23

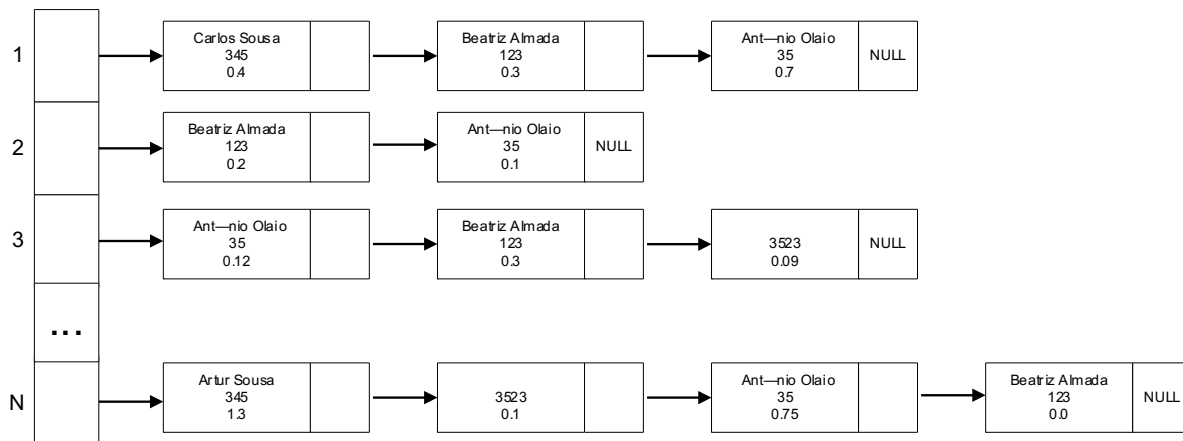
LEI, LEI-PL, LEI-CE

Se agruparmos os ponteiros iniciais de cada uma das listas num array de ponteiros o processamento do dicionário torna-se bastante mais simples. A estrutura genérica terá então o formato indicado na figura. As listas parciais também estão ordenadas alfabeticamente. Não existe diferença entre caracteres minúsculos e maiúsculos.

- Defina as estruturas necessárias de modo a poder implementar o dicionário.
- Escreva uma função em C que receba uma palavra como argumento e devolva como resultado o número de vezes que essa palavra já surgiu no texto.
- Escreva uma função em C que receba uma palavra como argumento e atualize o dicionário. Se essa palavra já fizer parte do dicionário o contador respetivo deve ser atualizado. Caso contrário deve ser criado um novo nó e inserido no sítio respetivo.

11. Vários alunos do ISEC participaram ao longo do ano nas famosas provas de ciclismo organizadas pela Associação de Estudantes. Um programa para fazer a gestão das classificações em cada uma das provas utiliza uma estrutura dinâmica com o formato exemplificado na figura.

Corridas



Para cada uma das provas realizadas, a informação sobre os concorrentes que a terminaram é armazenada numa lista ligada (existe uma lista para cada prova). A ordem pela qual os alunos aparecem nas listas indica a posição em que eles terminaram cada uma das corridas.

Ao longo do ano foram realizadas N corridas (logo, existem N listas). Este identificador foi definido através da diretiva `#define` no início do módulo (pode assumir que está definido e que pode ser utilizado pelas suas funções).

Os N ponteiros que apontam para o início de cada uma das listas com as classificações, são elementos de um array de ponteiros, de modo a facilitar o processamento da informação sobre as várias corridas.

Programação 2022/23

LEI, LEI-PL, LEI-CE

A informação relativa a cada um dos concorrentes é armazenada numa estrutura com a seguinte definição:

```
typedef struct concorrente no, *pno;
struct concorrente{
    char nome[200];          /*nome do concorrente*/
    int id;                  /*n.º de aluno: identificador único*/
    float analise;           /*nível de álcool no sangue*/
    pno prox;
};
```

a) Escreva uma função em C que apresente na consola a classificação verificada em cada uma das N corridas.

b) Escreva uma função em C que crie uma nova lista ligada com informação sobre os atletas que terminaram todas as corridas realizadas. Os elementos dessa nova lista devem ser do tipo `struct concorrente`. O valor do campo `analise` nas estruturas da nova lista é irrelevante. A função recebe como argumento o endereço inicial do array de ponteiros e devolve um ponteiro para o início da lista criada.

Notas:

- as listas originais não devem ser alteradas;
- a ordem dos elementos na nova lista não é relevante.

c) O campo `analise` indica a quantidade de álcool no sangue que existe nos atletas no final da cada corrida. Verifica-se que alguns dos concorrentes ultrapassam o limite máximo admitido. Estes concorrentes devem ser punidos e expulsos das provas de ciclismo organizadas no ISEC. Escreva uma função em C que elimine de todas as listas com as classificações, os concorrentes que ultrapassem em qualquer uma das provas o limite máximo estabelecido. O espaço ocupado pelos nós retirados das listas deve ser libertado. A função recebe como argumentos o endereço inicial do array de ponteiros e o limite máximo admitido.

Exemplo: considerando a figura anterior, se o máximo estabelecido for de 0.5, os atletas António Olaio e Artur Sousa deveriam desaparecer de todas as listas com as classificações das provas.

Programação 2022/23

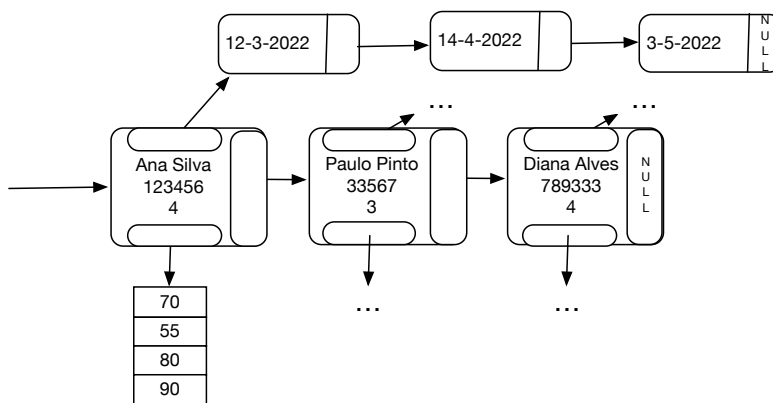
LEI, LEI-PL, LEI-CE

12. Considere as seguintes definições:

```
typedef struct data{
    int dia, mes, ano;
    struct data *prox;} *pdata;

typedef struct pessoa aluno, *paluno;
struct pessoa{
    char nome[100];    // Nome do aluno
    int num;           // Número do aluno
    int nprovas;        // Provas realizadas
    int* notas;         // Ponteiro para a tabela com as notas
    pdata faltas;       // Ponteiro para a lista com registo de faltas
    paluno prox;
};
```

Uma escola mantém uma estrutura dinâmica com informação sobre os seus alunos. A informação consiste em dados pessoais, notas obtidas nos testes já realizados e registo de faltas. Ao lado é mostrado um exemplo de uma estrutura dinâmica deste tipo. Existe uma lista ligada principal constituída por nós do tipo *aluno*. Cada nó tem informação pessoal de um aluno: nome, número e número de provas realizadas.



A partir de cada nó desta lista, é possível aceder a informação adicional sobre o aluno. Um vetor dinâmico de inteiros, referenciado pelo campo *notas*, contém as notas obtidas nas provas realizadas por esse aluno. Além disso, o campo *faltas* é um ponteiro para uma lista ligada de nós do tipo *struct data* onde estão registados os dias em que esse aluno faltou.

a) Escreva uma função em C que calcule e apresente a nota média de todos os alunos presentes na estrutura dinâmica. A função recebe como parâmetro um ponteiro para o início da lista de alunos. Deve escrever na consola o nome, o número de provas e a média obtida por cada aluno (1 linha por aluno). O valor da média deve surgir com uma casa decimal. Se o aluno ainda não tiver realizado nenhuma prova deve surgir o valor 0.0.

```
Ana Silva, 4 provas: 73.7
```

```
...
```

b) Escreva uma função em C que elimine da estrutura dinâmica o aluno que tiver mais faltas. Toda a memória dinâmica relativa a esse aluno deve ser eliminada (nó da lista principal, vetor com notas e lista de faltas). Caso existam vários alunos empatados nesta situação (maior número de faltas), a função não faz nenhuma alteração na estrutura dinâmica. A função recebe como parâmetro um ponteiro para o início da lista de alunos e devolve esse ponteiro depois da atualização.