Programação 2023/24

LEI, LEI-PL, LEI-CE

Aula Laboratorial 10

Bibliografia:

K. N. King. *C programming: A Modern Approach* (2nd Edition). W. W. Norton: capítulo 17.

Código de apoio para a aula: https://gist.github.com/FranciscoBPereira

Estruturas Dinâmicas

Exercícios Obrigatórios

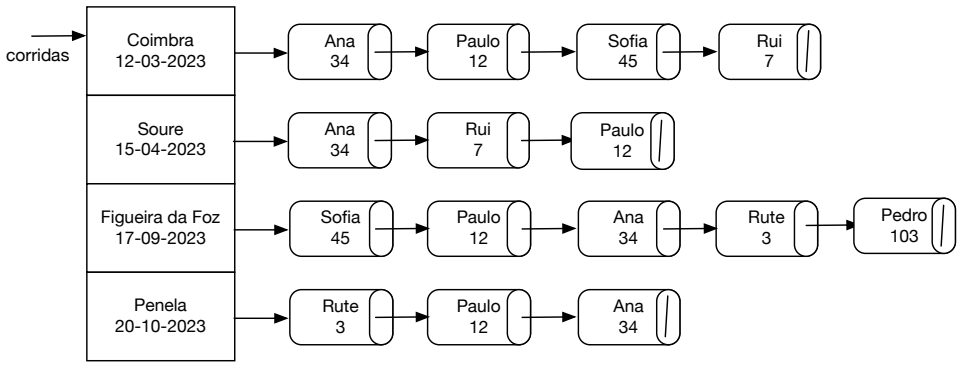
1. Considere as seguintes definições:

```
typedef struct {int d, m, a;} calendario;
typedef struct pessoa atleta, *pAtleta;

struct pessoa{
    char nome[100];     // Nome
    int id;             // Identificador numérico único
    pAtleta prox;
};

struct prova{
    char local[50];     // Local da prova
    calendario data;    // Data de realização da prova
    pAtleta p;
};
```

As informações sobre as classificações obtidas nas provas de atletismo da região de Coimbra são armazenadas numa estrutura dinâmica. Existe um vetor dinâmico constituído por estruturas do tipo *struct prova*. Cada uma destas estruturas armazena informação sobre uma prova. De cada prova sai uma lista ligada simples, constituída por nós do tipo *atleta*, contendo a classificação obtida nessa corrida. No exemplo da figura pode confirmar-se que já foram efetuadas 4 provas e que na primeira, realizada em Coimbra, terminaram 4 atletas: a Ana venceu e o Paulo ficou em segundo.



O código disponibilizado cria uma estrutura dinâmica igual à que surge na figura. A partir desse ponto deve implementar e testar as seguintes funções:

a) Escreva uma função em C mostre a classificação de uma prova. A função recebe como parâmetros o endereço e a dimensão do array de corridas e o nome da prova a imprimir.

b) Escreva uma função em C determine o número de provas terminadas por um atleta. A função recebe como parâmetros o endereço e a dimensão do array de corridas e o identificador do atleta a considerar. Devolve o número de provas que este atleta terminou.

c) Escreva uma função em C descubra quais os atletas que terminaram todas as provas. A função recebe como parâmetros o endereço e a dimensão do array de corridas. Escreve na consola o nome e identificador dos atletas que terminaram todas as corridas.

d) Escreva uma função em C que descubra qual o atleta campeão. Para este efeito considera-se que, em cada corrida, é atribuída a seguinte pontuação: 5 pontos para o vencedor, 3 para o segundo classificado e 1 para o terceiro. O atleta campeão é o que acumular mais pontos ao longo de todas as provas. A função recebe como parâmetros o endereço e a dimensão do array de corridas e escreve na consola o nome e identificador do atleta campeão. Em caso de empate, escreve a informação de todos os atletas que terminaram com pontuação máxima.

e) A função seguinte desclassifica um atleta de todas as corridas:

```
int desclassifica(struct prova *a, int totC, int id);
```

Recebe como parâmetros o endereço e a dimensão do array de corridas e o identificador do atleta a desclassificar. Deve eliminar este atleta de todas as corridas em que surja. O espaço ocupado pelos nós deve ser libertado. A função devolve o número de provas em que o atleta se encontrava.

f) A função seguinte adiciona uma nova prova à estrutura dinâmica:

```
struct prova* novaProva(struct prova *a, int *totC, char *n,
    calendario d, int nAt, char nomes[][100], int ids[]);
```

Recebe como parâmetros o endereço do array de corridas, o endereço de uma variável inteira onde se encontra a dimensão do array e os dados da nova prova: nome, data, número de atletas e os respetivos nomes e identificadores. A ordem pela qual os atletas surgem nos arrays dos 2 últimos parâmetros correspondem à sua classificação nesta prova. Esta função deve adicionar esta nova corrida à estrutura dinâmica, mantendo o ordenamento do array de corridas (data do calendário). Devolve o endereço inicial do array de corridas depois da inserção da prova.

Programação 2023/24

LEI, LEI-PL, LEI-CE

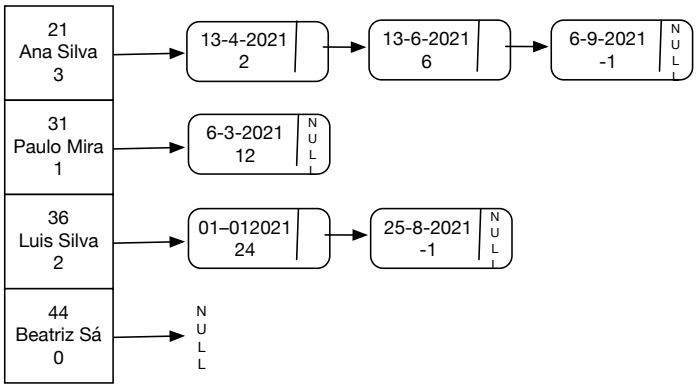
2. No contexto da gestão dos internamentos numa clínica, considere as seguintes definições:

```
typedef struct paciente Paciente, *pPaciente;
typedef struct internamento Inter, *pInter;
typedef struct {int dia, mes;} data;

struct paciente{
    int id;              // Identificador único
    char nome[100];      // Nome do paciente
    int ninternamentos;  // Número de internamentos nesse ano
    pInter lista;        // Ponteiro para a lista de internamentos
};

struct internamento{
    data din;             // Data de entrada
    int nDias;           // Número de dias internado
    pInter prox;
};
```

A informação completa dos pacientes registados e dos respetivos internamentos está armazenada numa estrutura dinâmica com as seguintes características: existe uma tabela dinâmica constituída por estruturas do tipo *Paciente*. Cada paciente registado na clínica tem os seus dados armazenados numa posição da tabela (Id numérico único, nome e número de internamentos registados no ano atual).



Os pacientes na tabela estão ordenados pelo valor do identificador. A cada paciente está associada uma lista ligada simples (através do campo *lista*), constituída por nós do tipo *Inter*, contendo informação dos seus internamentos. Cada nó da lista contém a data de internamento e o número de dias passado na clínica. Caso o paciente ainda esteja internado, o campo nDias tem o valor -1. Cada uma destas listas está ordenada por data. A figura em cima exemplifica um exemplo com 4 pacientes.

a) Escreva uma função que apresente na consola toda a informação armazenada na estrutura dinâmica. A função recebe, como parâmetros, o endereço do vetor de clientes e a sua dimensão.

b) Escreva uma função que descubra qual o paciente que já teve mais internamentos no ano atual. A função recebe, como parâmetros, o endereço do vetor de clientes e a sua dimensão. Devolve o identificador número do paciente com mais internamentos. Em caso de empate, a função devolve o identificador numérico mais baixo.

c) Escreva uma função em C que indique qual o paciente que já passou mais dias na clínica no ano atual. Na contabilização não devem ser considerados internamentos que ainda estejam a decorrer. A função recebe, como parâmetros, o endereço do vetor de clientes e a sua dimensão. Devolve o identificador do paciente com mais dias de internamento. Caso não existam pacientes com internamentos completos, a função devolve -1. Em caso de empate, a função devolve o identificador numérico mais baixo.

d) Escreva uma função em C que elimine da estrutura dinâmica um paciente com um determinado identificador numérico. Deve eliminar a estrutura do paciente do vetor dinâmico e a lista ligada de internamentos. Todo o espaço deve ser libertado. O cabeçalho da função é o seguinte:

```
pPaciente elimina(pPaciente v, int *nPaciente, int id);
```

A função recebe, como parâmetros, o endereço do vetor de clientes, o endereço de uma variável inteira onde está armazenada a dimensão do vetor dinâmico e o identificador do paciente a eliminar. Devolve o endereço do vetor dinâmico depois da atualização. Caso não exista nenhum paciente com o identificador indicado, a função não efetua nenhuma alteração.

Exercícios Complementares

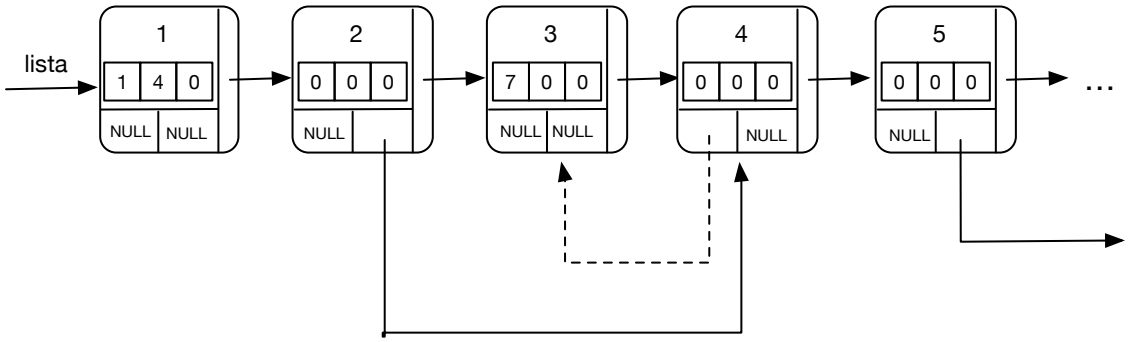
3. Uma estrutura dinâmica armazena um jogo de tabuleiro do tipo jogo da glória. Neste jogo, vários jogadores (3, no máximo) lançam alternadamente um dado para ir avançando nas casas de um tabuleiro. Ganha quem atingir primeiro a última posição. Em algumas casas existem escadas: um jogador que atinja 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 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;
};
```

Programação 2023/24

LEI, LEI-PL, LEI-CE



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 apresente uma listagem na consola com todas as cobras e escadas existentes no tabuleiro, indicando, para cada uma delas, em que posição se encontram e para onde levam. A função recebe como argumento um ponteiro para o início da estrutura dinâmica.

c) 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. 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.

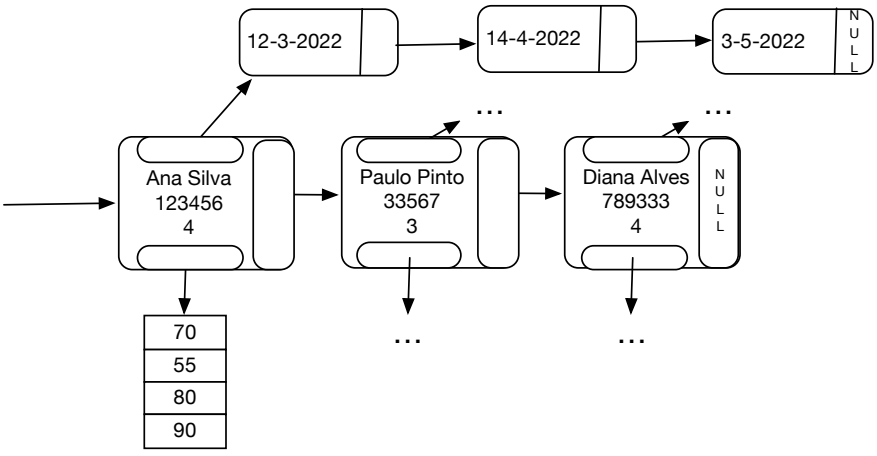
d) Escreva uma função em C que faça uma cópia de uma estrutura dinâmica deste tipo. A função recebe um ponteiro para o início da estrutura dinâmica e o número de casas do tabuleiro. Devolve um ponteiro para uma nova estrutura dinâmica que seja uma cópia da recebida por parâmetro. A estrutura dinâmica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

4. 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. A figura ilustra 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 apresente na consola o nome de todos os alunos que ainda não tenham nenhuma falta registada. A função recebe como parâmetro um ponteiro para o início da lista de alunos.

b) 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.

c) Escreva uma função em C que adicione uma nova prova a um aluno. Esta operação implica adicionar um novo valor ao vetor de provas do aluno. A função tem o seguinte protótipo:

```
paluno adicionaTeste(paluno p, char *nome, int num, int nota);
```

Recebe um ponteiro para o início da lista de alunos, o nome e número do aluno ao qual deverá ser adicionado o teste e o valor obtido na prova. Caso o aluno indicado ainda não se encontre na estrutura dinâmica, a função deve igualmente adicionar um novo nó à lista ligada de alunos. Ao responder a esta questão pode assumir que a lista não está ordenada por nenhum critério em particular. A função devolve um ponteiro para o início da lista de alunos atualizada.

d) Escreva uma função em C que registe uma nova falta a um aluno. Esta operação implica adicionar um novo nó à lista de faltas do aluno. Ao responder a esta questão deve considerar que as listas de faltas estão ordenadas por data. A função tem o seguinte protótipo:

```
int adicionaFalta(paluno p, int num, struct data d);
```

Recebe um ponteiro para o início da lista de alunos, o número do aluno ao qual deverá ser adicionada a falta e a data em que esta ocorreu. Caso o aluno indicado não se encontre na estrutura dinâmica, a função não efetua nenhuma alteração. A função devolve 1 no caso da adição ser efetuada com sucesso, ou 0, caso contrário.

e) 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.

Trabalho Autónomo

5. Em C, é possível alocar dinamicamente memória para armazenar arrays de 2 dimensões (matrizes). No entanto, isso não pode ser feito com uma única chamada à função *malloc()*, uma vez que ela efetua a alocação de memória de forma linear, o que dificulta o acesso e processamento de uma estrutura bidimensional. Considere que se pretende alocar dinamicamente memória para a matriz de inteiros 3×4 seguinte:

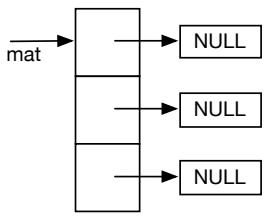
12	5	6	9
8	9	11	5
3	6	4	78

Em termos genéricos, uma estrutura que permite armazenar a matriz de inteiros M×N terá o seguinte formato:

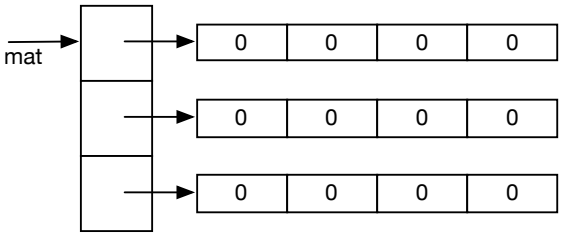
i. Declarar uma variável *mat* do tipo *int*** (ponteiro para ponteiro para inteiro):

```
int **mat = NULL;
```

ii. Alocar um vetor dinâmico com dimensão igual ao número de linhas (M). Cada um dos elementos deste vetor é um ponteiro, neste caso um ponteiro para um inteiro (*int**). A variável *mat* passa a armazenar o endereço inicial deste vetor.



iii. Cada uma das linhas da matriz é alocada individualmente. Neste caso serão M linhas, cada uma delas com N valores inteiros. Na prática, corresponde a alocar M vetores de N inteiros. Cada um destes vetores será referenciado por um dos ponteiros da alocação do ponto ii.



Esta implementação tem a vantagem de permitir um acesso simplificado aos elementos da matriz. Depois da alocação, a notação *mat[i][j]* é valida e refere-se ao elemento que está na posição *(i, j)* da matriz.

Implemente funções que permitam alocar e efetuem algumas das operações básicas de manipulação de uma matriz dinâmica de inteiros:

i. Criar matriz de inteiros: Recebe as dimensões e devolve a alocação completa de uma matriz de inteiros. Todas as posições são inicializadas com o valor 0.
ii. Libertar matriz de inteiros: Recebe endereço inicial da estrutura dinâmica e dimensões da matriz. Liberta todo o espaço alocado.
iii. Mostrar matriz: Recebe endereço inicial da estrutura dinâmica e dimensões da matriz. Apresenta uma listagem completa dos valores armazenados.
iv. Coloca valor (setPos): Recebe endereço inicial da estrutura dinâmica, dimensões da matriz, coordenadas de uma posição (x,y) e valor v. Coloca o valor v na posição (x, y). Deve garantir que é uma posição válida.
v. Obtém valor (getPos): Recebe endereço inicial da estrutura dinâmica, dimensões da matriz e coordenadas de uma posição (x,y). Devolve o valor armazenado na posição (x, y). Deve garantir que é uma posição válida.