Programação

Licenciatura em Engenharia Informática: 1º ano - 2º semestre 2019/2020

Guião laboratorial n.º 2

Estruturas – Arrays Dinâmicos de Estruturas

Tópicos da matéria:

- Criação de tipos estruturados.
- Ficheiros de cabeçalho (*Header files*)
- Arrays de estruturas.
- Arrays dinâmicos.
- Unions.

Bibliografia:

K. N. King, *C Programming: A Modern Approach*: Capítulos 15, 16 e 17.

A – Estruturas

1. Pretende-se representar um ponto num espaço com duas dimensões.

a) Crie um tipo estruturado que lhe permita armazenar as coordenadas inteiras (x, y) de um ponto.
b) Escreva uma função que imprima as coordenadas de um ponto recebido como parâmetro.
c) Escreva uma função que inicialize uma estrutura do tipo ponto com coordenadas indicadas pelo utilizador. A função recebe o endereço do ponto a inicializar.
d) Escreva uma função que desloque um ponto no espaço 2D. A função recebe, como parâmetros, o endereço do ponto a mover e o valor dos deslocamentos ao longo dos eixos *xx* e *yy*.
e) Escreva uma função que verifique se 3 pontos recebidos por argumento estão alinhados na mesma reta. A função devolve 1 se pertencerem à mesma reta, ou 0, caso contrário

Nota: O código deste exercício deve ser distribuído por 3 ficheiros: *main.c, ponto.c, ponto.h*. Deve utilizar os ficheiros que estão disponíveis no InforEstudante e completar o código em falta.

2. Pretende-se representar um retângulo num espaço com duas dimensões. O retângulo deve ser definido através do seu canto inferior esquerdo, altura e largura. A altura e a largura são valores inteiros positivos. Ao responder a esta questão, deve utilizar a estrutura definida no exercício 1.

a) Crie um tipo estruturado que lhe permita representar um retângulo.
b) Escreva uma função que imprima as coordenadas dos 4 cantos de um retângulo recebido como parâmetro.
c) Escreva uma função que inicialize uma estrutura do tipo retângulo. Toda a informação necessária é indicada pelo utilizador. A função recebe o endereço do retângulo a inicializar.
d) Escreva uma função que devolva a área de um retângulo passado como parâmetro;
e) Escreva uma função que verifique se um ponto está dentro de um retângulo (excluindo as arestas que definem a sua fronteira). A função recebe como parâmetros o retângulo e o ponto. Devolve 1 se o ponto estiver dentro do retângulo, ou 0, caso contrário.
f) Escreva uma função que desloque um retângulo no plano. A função recebe três parâmetros: o endereço do retângulo a deslocar, o deslocamento ao longo do eixo *xx* e o deslocamento ao longo do eixo *yy*.

B – Arrays de Estruturas

3. Pretende-se criar um programa que permita armazenar um conjunto de retângulos e efetuar algumas operações sobre eles. Sempre que possível, as operações sobre retângulos individuais devem ser feitas através de chamadas às funções implementadas na questão anterior.
a) Declare uma tabela que tenha capacidade para armazenar 10 retângulos. A tabela deve ser uma variável local da função `main()`.
b) Escreva uma função que adicione um novo retângulo à tabela. Os dados necessários são indicados pelo utilizador. A função deve verificar se existe espaço na tabela para adicionar o novo retângulo.
c) Escreva uma função que imprima a informação relativa a todos os retângulos armazenados na tabela. Para cada retângulo devem ser indicadas as coordenadas dos seus 4 cantos.
d) Escreva uma função que elimine da tabela o retângulo mais pequeno que aí estiver armazenado.
e) Escreva uma função que elimine da tabela todos os retângulos com área inferior a um determinado limite. Este limite é um dos parâmetros da função.
f) Escreva uma função que ordene os retângulos da tabela pela distância Euclidiana do seu centro à origem das coordenadas (do mais próximo para o mais afastado).

4. O aeródromo de Coimbra pretende um programa para gerir a informação relativa às partidas de voos. Cada voo é identificado por: número do voo, nome da companhia, cidade destino e hora de partida (horas e minutos).
a) Crie um tipo estruturado chamado *struct tempo* que permita armazenar as componentes de uma determinada hora de um dia (horas e minutos).
b) Crie um tipo estruturado chamado *struct voo* que permita armazenar a informação relativa a um voo. O campo que armazena a hora de partida deve ser do tipo *struct tempo.*
c) Desenvolva uma função que escreva o conteúdo dos campos de uma variável estruturada do tipo *struct voo*. A função recebe como argumento a variável já preenchida.

d) Desenvolva uma função que permita introduzir informação relativa a um novo voo. É passado como argumento um ponteiro para a estrutura a inicializar.
e) Desenvolva uma função que altere a hora de partida de um voo. Recebe como argumento um ponteiro para uma estrutura já preenchida e solicita ao utilizador as novas componentes da hora para efetuar a alteração.
f) Desenvolva uma função que verifique se um determinado voo já partiu. Recebe como argumentos uma estrutura do tipo *struct voo* onde está armazenada a informação do voo e uma estrutura do tipo *struct tempo* onde está armazenada a hora atual. Devolve 1 se o voo já tiver partido, ou 0, se isso ainda não tiver acontecido.
g) Declare uma tabela que lhe permita armazenar informação relativa a 300 voos. A tabela deve ser uma variável local da função `main()`.
h) Desenvolva uma função que permita adicionar novos voos à tabela. A função recebe como argumentos um ponteiro para o início da tabela e um ponteiro para o inteiro que indica quantos voos existem. É o utilizador que indica quantas novas entradas quer inserir e que especifica a informação relativa a cada um dos novos voos. A função deve atualizar a tabela e o inteiro referenciado pelo segundo argumento, indicando quantos voos passam a existir.
i) Desenvolva uma função que liste a informação completa de todos os voos armazenados na tabela. A função recebe como argumentos um ponteiro para o início da tabela e o número de voos que esta contém.
j) Desenvolva uma função que verifique quais os voos que partem nos próximos 30 minutos. Deve ser escrito o número, o destino e o tempo que falta para a partida de cada um desses voos. Um ponteiro para o início da tabela de voos, o número de elementos que esta contém e a hora atual são passados como argumentos.
k) Desenvolva uma função que retire da tabela todos os voos que já partiram. São passados como argumentos um ponteiro para o início da tabela, um ponteiro para um inteiro que indica quantos voos esta contém e a hora atual. A função deve atualizar a tabela e o contador de voos.
l) Crie um programa que construa um menu com o seguinte formato:

1. Introduzir novos voos
2. Listar todos os voos
3. Listar proximos voos
4. Atualizar tabela de voos
5. Terminar

O programa deve fazer a gestão das seleções que o utilizador for efetuando. Termina a execução quando for selecionada a opção 5.

Sugestão: Utilize a seguinte função para obter a hora atual do sistema. O resultado é devolvido numa estrutura do tipo `struct tempo`.

```
#include <stdio.h>
#include <time.h>

/* Estrutura auxiliar para guardar as componentes da hora */
struct tempo{ int h, m;};

struct tempo hora_atual(){
      time_t a;
      struct tm* b;
      struct tempo atual;

      time(&a);
      b = localtime(&a);
      atual.h = b->tm_hour;
      atual.m = b->tm_min;
      return atual;
}
```

```
/* Exemplo de utilização da função hora_atual() */
int main(){
      struct tempo t = hora_atual();
      printf(" São %2.2d:%2.2d\n", t.h, t.m);
      return 0;
}
```

Proposta de trabalho: Altere o programa de gestão do aeródromo de modo a que os voos fiquem ordenados na tabela por hora de partida. Efetue as alterações necessárias nas funções que implementou.

5. A pizzaria Pepe Verde decidiu informatizar a sua cozinha. A informação que deve ser armazenada é a seguinte:

- existem 6 ingredientes básicos necessários para a confeção das pizzas: queijo, tomate, fiambre, cebola, pimentos, cogumelos. É necessário saber, em cada momento, qual a quantidade em stock de cada ingrediente;
- existem 5 tipos de pizzas. Cada pizza é identificada pelo nome, tempo de cozedura, preço e quantidade necessária de cada um dos ingredientes;

a) Crie as estruturas de dados necessárias para armazenar toda a informação;
b) Inicialize as estruturas com informação à sua escolha;
c) Desenvolva uma função que verifique se existem ingredientes necessários para fabricar uma determinada pizza. A função deve receber como argumento a informação sobre qual a pizza a fabricar e uma referência que lhe permita aceder à estrutura de dados onde estão armazenadas as quantidades em stock dos ingredientes. A função devolve 1 se existirem ingredientes suficientes ou 0 no caso contrário;

d) Desenvolva uma função que atualize o preço de uma determinada pizza. A função recebe como argumento o novo preço e uma referência que lhe permita aceder à informação sobre a pizza que vai ser modificada.
e) Desenvolva uma função que escreva no monitor os nomes de todas as pizzas que demorem menos tempo a cozinhar do que um determinado valor. A função recebe como argumento o tempo máximo de cozedura permitido e uma referência para a estrutura de dados onde está armazenada a informação sobre todas as pizzas.

Nota: ao resolver as alíneas c) a e) deve assumir que as estruturas onde está armazenada a informação são locais (para ter acesso a elas deve utilizar os argumentos das diferentes funções);

C – Arrays Dinâmicos de Estruturas

6. Pretende-se criar uma agenda para armazenar contactos de telemóvel (nome e número). A agenda não deve ter um tamanho máximo pré-definido, ou seja, terá a capacidade necessária e suficiente para armazenar os contactos existentes em cada momento.
a) Crie um tipo estruturado que lhe permita armazenar a informação de um contacto.
b) Declare as variáveis na função *main()* que lhe irão permitir gerir um array dinâmico de contactos telefónicos.
c) Escreva uma função que adicione um novo contacto à agenda. Toda a informação necessária é indicada pelo utilizador. Deve garantir que os nomes são únicos, ou seja, não podem existir contactos com o mesmo nome na agenda.
d) Escreva uma função que liste todos os contactos existentes na agenda
e) Escreva uma função que receba um nome como parâmetro e indique qual o número de telemóvel dessa pessoa.
f) Escreva uma função que atualize o número de telemóvel de um determinado contacto armazenado na agenda.
g) Escreva uma pessoa que elimine da agenda o contacto de uma pessoa. O nome da pessoa a eliminar é um dos parâmetros da função.
h) Altere o que for necessário nas alíneas anteriores para garantir que os contactos na agenda se encontram ordenados pelo nome da pessoa.

7. Considere a seguinte estrutura:

```
typedef struct movimento mov;
typedef struct {int d, m, a;} data;

struct movimento{
      data dMov;
      char nconta[15];
      int val;
};
```

Cada estrutura do tipo *mov* consegue armazenar um movimento realizado por um determinado cliente de um banco. O campo *dMov* indica a data em que foi realizada a operação, o campo *nconta* contém o identificador da conta que realizou o movimento e o campo *val* indica qual o valor movimentado (positivo para depósito ou negativo para levantamento).
Todos os movimentos realizados numa agência bancária ao longo de um período vão sendo armazenados num array dinâmico. De cada vez que um novo movimento é efetuado, uma nova estrutura é adicionada ao final do array.

a) Declare as variáveis na função *main()* que lhe irão permitir gerir um array dinâmico de movimentos.
b) Escreva uma função que adicione um novo movimento ao array. Toda a informação necessária é indicada pelo utilizador. O novo movimento deve ser colocado no final do array.
c) Escreva uma função que liste todos os movimentos existentes no array.
d) Escreva uma função que liste todos os movimentos do array que foram realizados num determinado período. As datas limite do período são dois dos parâmetros da função.
e) Escreva uma função que, dado um número de conta, devolva a seguinte informação: número total de movimentos realizados, número de levantamentos, número de depósitos, saldo acumulado dos movimentos realizados.
f) Escreva uma pessoa que elimine do array todos os movimentos efetuados numa determinada data. A data a considerar é um dos parâmetros da função.

D – Unions e Enumerations

8. Considere as seguintes definições:

```
# define MAXP   100
enum categoria {LIVRO, REVISTA, JORNAL};

struct data {int dia, mes, ano;};

struct pub {
      char nome[50];
      struct data d_entrada;
      char cota[10];
      enum categoria tipo;
      union {
            char autor[50];
            int numero;
            struct data d_publicacao;
      } u;
};

int main(){
      struct pub catalogo[MAXP];
      int n_pub = 0;
      ....
}
```

O array `catalogo` serve para armazenar informação relativa a um conjunto de publicações de uma biblioteca. Considere que existem três tipos de publicações: livros, revistas e jornais.

Além da informação geral: `nome`, `data de entrada`, `cota` e `tipo de publicação`, cada publicação tem uma informação específica: `autor do livro`, `número da revista` ou `data de publicacao do jornal`.

A variável inteira `n_pub` indica quantas publicações existem no array `catalogo`. Varia entre 0 (não existe informação sobre nenhuma publicação) e MAXP (o array está completamente cheio). Tanto o `catalogo` como `n_pub` são variáveis locais da função `main()`.

A informação no array `catalogo` está ordenada por tipo de publicação: primeiro devem surgir todos os livros, em segundo lugar surgem as revistas e, no final, aparecem os jornais. Os elementos de cada categoria estão organizados por ordem de inserção no array.

a) Desenvolva uma função que permita inserir uma nova publicação no array `catalogo`. Os dados para inicialização são fornecidos pelo utilizador.
b) Desenvolva uma função que mostre toda a informação armazenada no array.
c) Desenvolva uma função que escreva no monitor os títulos de todos os livros que tenham sido escritos por um autor cujo nome é passado como argumento;
d) Desenvolva uma função que elimine do catálogo todos os jornais que tenham sido publicados antes de 1995;

9. Considere que, num determinado jardim zoológico, existem animais de três tipos: mamíferos, répteis e peixes (cada um dos animais só pertence a um dos grupos). Cada animal é identificado pelas seguintes características:

- Nome, país de origem e idade
- caso seja mamífero -> período de amamentação (inteiro)
- caso seja réptil -> o seu habitat natural (máximo 10 caracteres)
- caso seja peixe -> indicação se é de água doce ou salgada (caracter)

Estabeleça as estruturas de dados que possibilitem representar:

a) um animal;
b) um conjunto de 100 animais;
c) Desenvolva uma função que permita inicializar a estrutura de dados criada na alínea b) com N animais. O valor N é passado como argumento para a função.
d) Descobriu-se que o utilizador cometeu um erro ao inicializar a estrutura de dados relativa a um golfinho, uma vez que o considerou como sendo um peixe de água salgada. Desenvolva uma função que localize o erro na estrutura de dados e o corrija.

Nota: A estrutura de dados criada na alínea b) deve ser uma variável local da função `main()`, logo é necessário utilizar os mecanismos habituais de comunicação entre funções (argumentos e valor devolvido) para resolver as alíneas c) e d).