Programação

Folha Prática 8

Lab. 8

Departamento de Informática
Universidade da Beira Interior
Portugal
2017

LAB. 8

8ª semana

SUBPROGRAMAS+STRUCTS+ARRAYS DE STRUCTS

1. Revisão.
2. Objectivos.
3. Exercícios

Lab. 8

SUBPROGRAMAS, STRUCTS E ARRAYS DE STRUCTS

1. Revisão

1. Arrays:
   1.1. Tipos de dados compostos por apenas um tipo de dados (simples ou composto) guardados numa zona contígua de memória => elementos do mesmo tipo.
   1.2. Acesso directo a determinado elemento usando a sua posição: nomeArray [posicao] (os elementos são indexados).
2. Structs (estruturas):
   2.3. Tipos de dados compostos por vários tipos de dados (simples ou composto) guardados numa zona contígua de memória => elementos não necessariamente do mesmo tipo.
   2.4. Os elementos também chamados de campos ou componentes não são indexados.
   2.5. Definição de um novo tipo de dados usando uma struct:

```
typedef struct {
        tipo_campo1     var1;
        tipo_campo2     var2;
        ...
        tipo_campoN     varN;
} nome_tipo_dados;
```

   2.6. Definição de uma variável do novo tipo de dados criado:

```
nome_tipo_dados nome_variavel;
```

3. Subprogramas
   3.1. Passagem de arrays para funções.
   3.2. Passagem de estruturas para funções.
   3.3. Retorno de estruturas a partir de funções.

2. Objectivos

No final deste módulo prático, o aluno deve ser capaz de:

1. Programar em C usando structs e arrays de structs recorrendo à utilização de subprogramas, ou seja, deve ser capaz de construir aplicações simples de software.
2. Programar pequenas aplicações de software em C, embora sem recorrer ainda à utilização de ficheiros.

3. Exercícios

Exercício 8.1 (Soma de números complexos)

Usando o tipo **COMPLEXO** abaixo transcrito, escreva um programa que retorne a soma de dois números complexos. Lembre-se que um número complexo é um número que pode ser escrito na forma **a+b*i***, em que a e b são números reais, sendo a designado por parte real e b por parte imaginária dum número complexo.

```
typedef struct
{
        float   a;      /* parte real */
        float   b;      /* parte imaginaria */
} COMPLEXO;
```

Notas de análise do problema:

Utilize funções para:

- Ler um número complexo
- Escrever um número complexo.
- Calcular a soma de dois números complexos.

Nota: a soma de dois números complexos é um número complexo.

Exercício 8.2 (Cadastro de alunos nas residências universitárias)

Tendo em conta a seguinte estrutura, bem como o respectivo tipo de dados, que permite cadastrar os registos de alunos em residências universitárias:

```
#define MAX 20

typedef struct
{
        char nome[MAX];
        char apelido[MAX];
        char residencia[MAX];
        int telefone;
} ESTUDANTE;
```

1) Escreva a função **equalEntry** que toma dois registos de alunos como argumentos, devolvendo depois o valor 1 (verdade) se os dois registos são idênticos ou o valor 0 (falso) no caso contrário.
2) Escreva uma função **comesFirst** que toma dois registos de alunos como argumentos, devolvendo depois o valor 1 (verdade) se o primeiro registo tem precedência alfabética sobre o segundo registo ou o valor 0 (falso) no caso contrário. A comparação alfabética decorre pela comparação inicial dos nomes e só depois se compara os apelidos.
3) Escreva um programa que leia do teclado dois registos, que escreva no ecrã se são idênticos (usando a função **equalEntry**), e que escreva ainda no ecrã qual deles vem em primeiro lugar (usando a função **comesFirst**).

Exercício 8.3 (Gestão do tempo)

Tendo em conta a seguinte estrutura, bem como o respectivo tipo de dados, que permite representar o tempo no fromato de horas (hh), minutos (mm) e segundos (ss.sss). Um exemplo de tempo neste formato é, por exemplo, 12:34:56.123.

```
typedef struct
```

```
{
        int hh;
        int mm;
        double seconds;
} TIME;
```

1) Escreva a função **convertTime** que converta o tempo que lhe é passado em segundos no formato ss.sss para o tempo no formato hh:mm.ss.sss. O tempo neste formato deverá ser devolvido pela função como uma struct do tipo TIME. Escreva depois um pequeno programa para testar a conversão. O protótipo da função anterior é o seguinte:

```
TIME convertTime(double secs);
```

2) Escreva a função **addTime** que toma dois tempos do tipo TIME como argumentos, e que depois devolve a soma dos tempos numa estrutura do tipo TIME. Escreva depois um pequeno programa para testar a conversão. O protótipo da função anterior é o seguinte:

```
TIME addTime(TIME a, TIME b);
```

3) Como é sabido, a passagem e a devolução de estruturas para e de uma função não é muito eficiente porque a cópia de todos os seus campos ou variáveis leva algum tempo. A eficiência aumenta quando na passagem de parâmetros se usa apontadores, pois só se copia ou transfere o endereço de cada estrutura. Reformule a função **addTime** de tal modo que os seus argumentos sejam agora variáveis apontadoras ou endereçadoras.

Exercício 8.4 (Aplicação geométrica interactiva)

Desenhe e implemente uma aplicação geométrica interactiva com o seguinte menu:

```
1. Distância entre dois pontos em R3.
2. Ponto médio entre dois pontos em R3.
3. Baricentro de N pontos em R3.
4. Exit
```

Após o cálculo de cada item do menu, o controlo deve regressar ao menu. A aplicação só termina quando o utilizador selecionar a opção 4. A aplicação deve ignorar outras opções que não as de 1, 2, 3 e 4.

Notas de análise do problema:

Obviamente, teremos de definir um tipo de dados POINT3D através de uma struct, bem como uma função

para cada um dos problemas associados às três primeiras opções do menu. Será também necessário ter uma função para input e outra de output de pontos.

Exercício 8.5 (Rectângulos)

A partir do tipo de dados POINT3D do Exercício 8.4, defina o tipo de dados RECTANGLE para rectângulos alinhados com os eixos, os quais são definidos pelos dois pontos da diagonal principal, ou seja o ponto do canto inferior esquerdo e o ponto do canto superior direito.

Após a definição destes tipos de dados, escreva as seguintes funções:

1) A função **rectangleArea** que determine a área de um rectângulo, assumindo que lhe são passados os dois vértices da diagonal principal. O protótipo desta função é o seguinte:

```
float rectangleArea (POINT3D p, POINT3D q);
```

2) A função **fallsWithin** que determine se ou não um quadrado está completamente dentro de outro. O protótipo desta função é o seguinte:

```
int fallsWithin (RECTANGLE r, RECTANGLE s);
```

Exercício 8.6 (Pontos mais próximos e mais afastados)

A partir do tipo de dados POINT3D do Exercício 8.4, escreva um programa que leia uma sequência de pontos, guardando-os num array de pontos.

Após a leitura dos pontos a partir do teclado, as seguintes funções deverão ser invocadas para calcular os dois pontos mais próximos e os dois pontos mais afastados:

1) A função **nearestPoints** que determina os dois pontos mais próximos. O protótipo desta função é o seguinte:

```
void nearestPoints(POINT3D *arrayOfPoints, POINT3D *p, POINT3D *q);
```

2) A função **farthestPoints** que determina os dois pontos mais afastados. O protótipo desta função é o seguinte:

```
void farthestPoints(POINT3D *arrayOfPoints, POINT3D *p, POINT3D *q);
```

Exercício 8.7 (Espécies com pernas e braços)

Defina o tipo de dados SPECIES através de uma estrutura que represente as espécies animais e materiais com pernas e braços. A estrutura tem 3 campos: nome, pernas e braços. Defina ainda um array de SPECIES onde guardará as representações de umas quantas espécies. Preencha os campos de cada espécie via teclado. Depois, faça a seguinte saída de dados para o ecrã:

```
Um humano tem 2 pernas e 2 braços
Um cão tem 4 pernas e 0 braços
Uma televisão tem 4 pernas e 0 braços
Uma cadeira tem 4 pernas e 2 braços
etc...
```

Exercício 8.8 (Aplicação para gestão de pessoal duma empresa)

Desenhe e construa uma aplicação interactiva que permita fazer a gestão do pessoal duma empresa. A aplicação deve, através de um menu, executar as seguintes tarefas:

1. Cadastrar os funcionários, sendo que a informação do registo de cada funcionário deve incluir pelo menos o nome, a categoria, o salário, o seu número de entrada na empresa e a data de entrada na empresa;
2. Apresentar os registos dos funcionários pelo número de entrada na empresa;
3. Procurar um funcionário pelo seu número de entrada;
4. Apresentar por ordem crescente os registos dos funcionários que recebem salários superiores a um dado valor em euros;
5. Apresentar por ordem crescente os registos dos funcionários que recebem salários inferiores a um dado valor em euros;
6. Sair da aplicação de cadastro.

Exercício 8.9 (Aplicação de cálculo da média final de licenciatura)

Escreva uma aplicação em C que permita a um aluno de licenciatura calcular a sua média final de curso. A aplicação deve incluir um menu a partir o qual de pode inserir e eliminar disciplinas do plano de estudos, bem como introduzir as notas obtidas nas diferentes unidades curriculares.

Exercício 8.10 (Aplicação interactiva para gestão de livros)

Considere o tipo struct **LIVRO** e um array de structs do tipo LIVRO com tamanho 100 identificado por biblioteca:

```
typedef struct
{
        char    titulo[40];
        char    autor[20];
        float   numeroDoCatalogo;
        int     anoDePublicacao;
        int     numeroDaCopia;
} LIVRO;

#define numeroLivros 100

LIVRO biblioteca [numeroLivros];
```

As tarefas principais a desenvolver nesta aplicação são as seguintes:

1. Altere a estrutura LIVRO por forma a incluir dois campos adicionais: dataEmprestimo (tipo DATE a definir) e dataRetorno (tipo DATE).
2. Escreva uma função de saída do seguinte menu:

```
BIBLIOTECA DA UBI
1 – Inserir novo livro
2 – Procurar livro por título.
3 – Procurar livro por autor.
4 – Alterar nome do livro
5 – Apagar livro
6 – Registar data de empréstimo de um livro.
7 – Registar data de retorno de um livro.
8 – Listar livros existentes na biblioteca.
9 - Sair
```

3. Coloque todas as opções do menu a funcionar (o programa só deverá terminar quando o utilizador seleccionar a opção "Sair").

   Para isso escreva funções para:

   - Ler um livro.
   - Mostrar os dados de um livro.
   - Inserir um livro.
   - Procurar um livro através do seu título.

- Procurar um livro através do nome do seu autor.
- Alterar o título de um livro.
- Eliminar um livro dos registos da biblioteca.
- Registar a data de empréstimo de um livro.
- Registar a data de retorno de um livro.
- Listar todos os livros existentes.