Universidade Federal de Ouro Preto - UFOP
Instituto de Ciências Exatas e Biológicas - ICEB
Departamento de Computação - DECOM
Disciplina: BCC 201 Introdução à Programação

Exercícios

1. 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 de $a + bi$, em que a e b são números reais, sendo a designado por parte real e b por parte imaginária de um número complexo.

```
typedef struct Complexo
{
    float a;
    float b;
}TComplexo;
```

   Utilize funções para:

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

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

2. Usando a estrutura fornecida, referente a atletas, elabore um programa para ler os dados de 15 atletas de uma equipe, calcular a média das idades e das alturas, e finalmente, exibir os dados lidos e as duas médias calculadas.

```
typedef struct TipoAtleta{
     int   matricula;
     int   idade;
     float altura;
}TAtleta;
```

3. Usando a estrutura `TAtleta` do exercício anterior, crie um vetor para armazenar os dados dos 15 atletas da equipe. Crie também uma estrutura para representar os dados de uma equipe: a relação de 15 atletas, o nome fantasia da equipe, e a data em que ela foi fundada. Para armazenamento da data, crie uma estrutura contendo dia, mês e ano. São ao todo 10 equipes. Desenvolva um programa para realizar o cadastro e em seguida a exibição destes dados. Use `#define` para criação de contantes que permitam a modificação do número de atletas das equipes e o número de equipes.

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

```c
#define MAX 20

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

(a) 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.

(b) 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.

(c) Escreva um programa que leia do teclado dois registos, que escreva na tela se são idênticos (usando a função `equalEntry`), e que escreva ainda na tela qual deles vem em primeiro lugar (usando a função `comesFirst`).

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

```c
typedef struct Time
{
 int hh;
 int mm;
 double seconds;
} TIME;
```

(a) 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);`

(b) 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);`

6. Definir um tipo de dados `POINT` através de uma *struct*, depois 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:

(a) A função *nearestPoints* que determina os dois pontos mais próximos.

(b) A função *farthestPoints* que determina os dois pontos mais próximos.