Universidade de Itaúna

Ciência da Computação

Laboratório de Algoritmos I

Exercícios - Funçõs e Structs

1) Faça um programa em C que utilize structs para armazenar os dados de um funcionário de uma empresa. Um funcionário de uma empresa deve possuir:

    - Nome (string de até 30 caracteres)
    - Idade
    - Sexo (representado por um caractere, ‘M’ ou ‘F’)
    - CPF (armazenado em string)
    - Cargo que ocupa (string de ate 30 caracteres)
    - Salário
    - Data de Nascimento (dia e ano números inteiros, mês deve ser uma string)
    - Codigo do Setor onde trabalha (0-99)

    Você pode definir quantas estruturas achar necessário. Seu programa deve criar um vetor de 3 funcionários. Use a diretiva `define` para definir o tamanho do vetor. Em seguida, o usuário deve entrar com as informações para preencher esse vetor. Finalmente, seu programa deve imprimir o vetor preenchido.

2) Nesse exercício vamos trabalhar com compromissos de uma pessoa. Um compromisso possui um texto (string de até 200 caracteres), uma data e um horário. Uma data deve possuir dia, mês e ano, todos números inteiros. Um horário deve possuir hora, minuto e segundo, todos também números inteiros.

    a) Crie em seu programa todas as estruturas necessárias para armazenar as informações de compromissos.
    b) Crie agora um vetor de compromissos de 20 posições. Use a diretiva `define` para definir o tamanho desse vetor.
    c) Inicialize o vetor de compromissos com números aleatórios da seguinte forma: o dia da data deve ser um número entre 1 e 20, o mês deve ser um número entre 1 e 12, o ano deve ser um número entre 2016 e 2019. A hora do horário deve ser um número entre 0 e 23, o minuto um número entre 0 e 59, o segundo também um número entre 0 e 59. Por fim, o texto de todos os compromissos deve ser “Compromisso de teste gerado aleatoriamente.”.
    d) Imprima o vetor de compromissos inicializado anteriormente no seguinte formato:

```
========Lista de Compromissos========
Compromisso x:
	Data: xx/xx/xxxx
	Horario: xx:xx:xx
	Texto: Compromisso de teste gerado aleatoriamente.

Compromisso x:
	Data: xx/xx/xxxx
	Horario: xx:xx:xx
	Texto: Compromisso de teste gerado aleatoriamente.
```

3) Nesse exercício vamos trabalhar com pontos no plano cartesiano.
    a) Crie uma estrutura de dados de nome `Ponto`, que armazena um ponto do plano cartesiano. Um ponto deve possuir uma coordenada x e uma coordenada y, ambas do tipo double.
    b) Crie uma estrutura de dados de nome `Retangulo`, que armazena as informações de um retângulo no plano cartesiano. Um retângulo deve possuir dois pontos: um que representa seu ponto da superior esquerda, e um que representa seu ponto da inferior direita.
    c) Crie uma função de nome `distancia`, que receba como parâmetros duas variáveis do tipo `Ponto` e retorne um double, que representa o valor da distância entre eles. A distância entre dois pontos pode ser calculada por $\sqrt{(x_b - x_a)^2 + (y_b - y_a)^2}$. Utilize a função `sqrt()` biblioteca `math.h` para calcular a raiz quadrada!
    d) Crie uma função de nome `maisProximoOrigem` que receba como parâmetros duas variáveis do tipo `Ponto` e retorne o `Ponto` mais próximo à origem (lembre-se, a origem é o ponto de coordenadas (0.0, 0.0) do plano cartesiano). Essa função deve fazer uso da função `distancia()`, implementada anteriormente.
    e) Crie uma função de nome `estaContido`, que recebe como parâmetros uma variável do tipo `Ponto` e uma variável do tipo `Retangulo`. Essa função deve retornar `1` caso o ponto passado por parâmetro esteja dentro do retângulo passado por parâmetro, e `0` caso contrário.
    f) Usando as funções criadas anteriormente, escreva um programa que leia, via scanf(), as informações de dois pontos e um retângulo. Em seguida seu programa deve informar a distância entre os dois pontos informados, qual deles está mais próximo da origem, e se cada um dos pontos está ou não contido no retângulo informado. Segue um exemplo de entrada e saída do programa. Os dados sublinhados foram informados pelo usuário.

```
Entre com as coordenadas x e y do primeiro ponto: 5.0 4.0
Entre com as coordenadas x e y do segundo ponto: 1.0 1.0
Entre com as coordenadas x e y do ponto da superior esquerda do
retangulo: 0.0 4.0
Entre com as coordenadas x e y do ponto da inferior direita do retangulo:
4.0 0.0

Distancia entre os dois pontos: 5.000000

Ponto mais proximo a origem: (1.000000, 1.000000)

O ponto (5.000000, 4.000000) nao esta contido no retangulo!
O ponto (1.000000, 1.000000) esta contido no retangulo!
```