Exercícios 04 – TDA

4.1 Defina o tipo **Circulo** para armazenar as coordenadas do centro e a medida do raio de um círculo.

Implemente as seguintes operações:

criaCirculo Saída: um círculo Entrada: coordenadas do centro e o raio de um círculo Descrição: Atribui os valores iniciais	**areaCirculo** Entrada: Um círculo Retorno: A área do círculo
perímetroCirculo Entrada: Um circulo Retorno: O perímetro do círculo	

Escreva um programa em C para ler o raio de 2 círculos. Criar dois círculos com os respectivos raios. O primeiro localizado na origem e o segundo nas coordenadas 10,30. Exibir a área dos dois círculos e o perímetro do círculo que possui a maior área. Utilizar as funções **criaCirculo**, **areaCirculo** e **perímetroCirculo**.

OBS: Não referenciar diretamente os elementos internos da estrutura.

4.2 Implemente as seguintes operações:

exibeCirculo Entrada: um círculo Descrição: Exibe o dados de um círculo	**moveCirculo** Entrada/Saída: Um círculo Entrada: valor deslocado na direção X valor deslocado na direção Y Descrição: Translada o centro do círculo considerando os valores de deslocamentos fornecidos como argumentos.
comparaCirculos Entrada: 2 círculos Retorno: 0 se os círculos possuem o mesmo raio. 1 se o primeiro círculo possui um raio maior que o segundo. -1 se o segundo círculo possui um raio menor que o primeiro.	**estaDentroDoCirculo** Entrada: um círculo e as coordenadas x e y de um ponto. Retorno: 0 se o ponto está localizado fora do círculo. 1 se o ponto está localizado dentro do círculo.

Escreva um programa para ler o raio e as coordenadas do centro de 2 círculos. Criar **dois círculos** com os valores lidos. Exibir uma mensagem indicando se o primeiro círculo é maior, menor ou igual ao segundo. Utilize a função **comparaCirculos** para comparar os círculos. Imprimir os dados (coordenadas e raio) dos dois círculos com a função **exibeCirculo**. A seguir ler dois inteiros que representam deslocamentos nas direções X e Y. Aplicar o deslocamento no segundo círculo informado utilizando a função **moveCirculo**. Exibir novamente os dados do segundo círculo.

OBS: Não referenciar diretamente os elementos internos da estrutura.

4.3 Escreva um programa para criar um círculo com raio igual a 5 e coordenadas 10,5 para seu centro. A seguir ler as coordenadas de 5 pontos e exibir (para cada ponto) uma mensagem que indique se ele está dentro ou fora do círculo. Utilize a funçã **estaDentroDoCirculo**.

OBS: Não referenciar diretamente os elementos internos da estrutura.

4.4 Defina o tipo **Conta** para armazenar o número de uma conta e seu respectivo saldo.

Implemente as seguintes operações:

criaConta Saída: Uma conta Entrada: Número da conta Descrição: Atribui os valores iniciais. O saldo deve ser zerado.	**depositaNaConta** Entrada/Saída: Uma conta Entrada: valor do depósito. Descrição: Atualiza o atributo saldo com o valor do depósito.
retiraDaConta Entrada/Saída: Uma conta Entrada: valor da retirada. Descrição: Atualiza o atributo saldo com o valor da retirada.	**obtemSaldo** Entrada: uma conta Retorno: O saldo da conta

Escreva um programa para controlar a conta corrente e a conta poupança do sr. Otobildes. As contas são integradas de forma que quando não houver saldo suficiente na conta corrente uma transferência automática da conta poupança cobrirá um eventual saldo (o valor da transferência deve ser igual ao valor necessário para cobrir o saldo negativo). O programa deve ler o número e o saldo inicial da conta corrente e da poupança criando duas variáveis do tipo **Conta** (cada uma representa uma conta). A seguir ler uma quantidade indeterminada de duplas de dados representando respectivamente o código da operação (**1**.Depósito conta corrente **2**.Depósito poupança **3**.Retirada conta corrente **4**.Retirada poupança **5**.Fim) e o valor do movimento. O programa termina ao ser informado um código igual a **5** (nesta situação o valor do movimento não deve ser lido).

- Cada conta deve ser armazenada em uma variável do tipo **Conta**.
- A cada operação executada o programa deve exibir o saldo atualizado das duas contas.
- A operação de retirada da conta poupança só deverá ocorrer se houver saldo disponível, caso contrário a mensagem "Saldo insuficiente" deverá ser exibida.
- As operações de depósito e retirada devem ser realizadas com chamadas as funções **depositaNaConta** e **retiraDaConta**.
- A operação de retirada da conta corrente deverá ser executada da seguinte forma:

```
se saldo na conta corrente for insuficiente então
   se saldo na poupança for suficiente para cobrir o valor que falta então
       transferir o valor que falta para a conta corrente e executar a retirada
   senão
       exibir a mensagem "Saldo insuficiente"
   fim_se
senão
   executar a retirada
fim_se
```

OBS: Não referenciar diretamente os elementos internos da estrutura.

4.5 Defina o tipo **Retangulo** para armazenar as coordenadas dos pontos superior esquerdo e inferior direito. Utilizar o tipo **Ponto** desenvolvido em aula para declarar as coordenadas.

Implemente as seguintes operações:

criaRetangulo Saída: um retângulo Entrada: coordenadas dos pontos superior esquerdo (x1,y1) e inferior direito (x2,y2) de um retângulo Descrição: Atribui os valores iniciais	**areaRetangulo** Entrada: Um retângulo Retorno: A área do retângulo

Escreva um programa para ler as coordenadas dos pontos superior esquerdo e inferior direito de um retângulo. Criar um retângulo e imprimir a sua área.

OBS: Não referenciar diretamente os elementos internos da estrutura.