P2 Programação II 2013.2 Departamento de Informática /PUC-Rio

QUESTÃO 1

Considere que uma instituição mantém os dados biométricos de pessoas guardados em um vetor de ponteiros para o seguinte tipo estruturado:

```
struct bio {
 int idade;
 float peso;
 float altura;
};
typedef struct bio Bio;
```

Considere também que este vetor já está ordenado em ordem crescente de idade. Por exemplo, considere um vetor com ponteiros para os seguintes dados:

{18,60,2.00},{18,65,2.00},{18,60,1.90},{20,65,1.80},{20,40,1.70},{20,30,1.60},{20,40,1.55},{25,50,1.70},{30,40,1.70},{30,45,1.85}

1a) Escreva a função `buscaFaixaIdade` que, dada uma certa idade, disponibiliza os índices de início e fim dos registros desta idade usando a técnica de busca binária. Esta função deve retornar 1 se encontrar ao menos um registro com a idade pedida e -1 se não for encontrado qualquer registro com a idade solicitada. Você não pode criar novos tipos estruturados nem novos vetores. Por exemplo, para o vetor acima e para a idade 20, temos `v[3]` contendo um ponteiro para {20,65,1.80} e `v[6]` para {20,40,1.55}. Neste caso, a função retornará 1 e disponibilizará os valores de 3 e 6 para os índices procurados.

1b) Escreva a função `ordenaFaixaIdade` que, dados o vetor, o tamanho do vetor e a idade, ordena apenas a faixa da idade fornecida em ordem crescente de peso e altura, utilizando a técnica de Ordenação Rápida. Você deve usar obrigatoriamente a função `buscaFaixaIdade` da questão 1a (mesmo que você não a tenha feito) e deve evitar percorrer todo o vetor. Você pode usar quantas funções auxiliares quiser, suas ou das bibliotecas C. Você não pode criar novos vetores nem pode perder a ordenação por idade do vetor como um todo. Faça obrigatoriamente as comparações com o pivô através de uma função auxiliar. No exemplo da Questão 1a, você obteria o seguinte:

{18,60,2.00},{18,65,2.00},{18,60,1.90},{20,30,1.60},{20,40,1.55},{20,40,1.70},{20,65,1.80},{25,50,1.70},{30,40,1.70},{30,45,1.85}

QUESTÃO 2

Projete um TAD `Circulo`, sem dependência de outros TADs, onde você cria um novo tipo de dados que é um ponteiro para um tipo estruturado definido por 3 (três) componentes: as coordenadas x e y do centro do círculo e o raio do círculo; isto é:

```
typedef struct circulo * Circulo;
```

Definindo `Circulo` como ponteiro, temos uma visão ainda mais abstrata do que usando `typedef struct circulo Circulo` (isto é, escondemos ainda mais os detalhes de implementação). Tomando cuidado com esta forma de definir `Circulo`, implemente as funções deste TAD que sejam suficientes para um desenvolvedor externo escrever um programa no módulo `meuPrograma.c` contendo a função principal `main` e a função

```
int circuloAcima(Circulo c).
```

A função `circuloAcima`, que não pertence ao TAD, testa se o círculo `c` está acima do eixo horizontal x (o que é verdade para o círculo c_1 e c_3 e falso para o círculo c_2 da Fig. 1). Apresente estas implementações

mínimas nos espaços abaixo, colocando o nome dos módulos no local sublinhado e usando todos os `#includes` necessários. Lembre que projetar um TAD requer definir funções para criar e liberar o tipo.

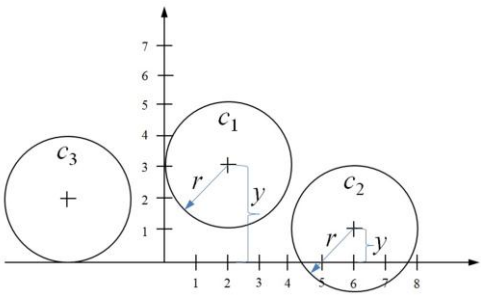

Fig. 1

Módulo Interface do TAD: c
Módulo usuário do TAD:
Módulo de implementação do TAD: