EXERCÍCIOS DE PREPARAÇÃO PARA P2

Considere que uma estatística coleta idades e pesos de pessoas anonimamente e guarda os dados em um vetor de ponteiros **v** para a seguinte estrutura:

```
struct dados
{
   int idade;
   int peso;
};
typedef struct dados Dados;
```

[1] Escreva a função **ordenaDados** que recebe o número total de dados e o vetor **v** e ordena este vetor em ordem crescente de idade e peso, usando o algoritmo de quick sort. Você deve obrigatoriamente escrever uma função auxiliar de comparação. Você pode usar o qsort da bibliioteca do C se desejar. Por exemplo, para um vetor **v** com ponteiros para as seguintes 8 estruturas:

```
{25,50},{20,30},{30,40},{20,65},{20,40},{18,60},{30,45},{18,65}
```

você obtém a seguinte ordenação:

```
{18,60},{18,65},{20,30},{20,40},{20,65},{25,50},{30,40},{30,45}
```

[2] Escreva a função de busca binária **buscaPesoMenor** que recebe o número total de dados, o vetor **v** e uma específica idade, e retorna o índice do dado com o menor peso referente à faixa da idade fornecida. Por exemplo, para o vetor ordenado da questão anterior e idade = 20, a função retorna o índice 2, que corresponde à estrutura {20,30}.
Você deve obrigatoriamente escrever uma função auxiliar de comparação.

Refaça os exercícios [1] e [2] projetando `Dados` como um TAD que disponibiliza as seguintes funções exportadas: **dadosCria**, **dadosAcessaIdade** e **dadosAcessaPeso**. Apresente a sua solução da seguinte forma:

[3a] Escreva a interface **dados.h** do TAD.

[3b] Escreva o módulo **dados.c** que implementa as funções do TAD. Coloque todos os comandos #include necessários.

[4] Reescreva a função **ordenaDados**, usando o TAD.

[5] Reescreva a função **buscaPesoMenor**, usando o TAD.

GABARITO

[1]

```
static int comparaDados(Dados * a, Dados * b)
{
   return (a->idade > b->idade || (a->idade==b->idade && a->peso > b->peso));
}

void ordenaDados(int n, Dados ** v) // Quick Sort
{
   Dados * x = v[0];
   Dados * temp;
   int a = 1;
   int b = n-1;
   if (n<=1)
      return;
   do
   {
      while (a < n && !comparaDados(v[a],x))
         a++;
      while (comparaDados(v[b],x))
         b--;
      if (a < b)
      {
         temp = v[a];
         v[a] = v[b];
         v[b] = temp;
         a++;
         b--;
      }
   } while (a <= b);
   v[0] = v[b];
   v[b] = x;
   ordenaDados(b,v);
   ordenaDados(n-a,&v[a]);
}
```

[2]

```
static int comparaIdade(int idade, Dados * b)
{
   if (idade < b->idade)
      return -1;
   else
      if (idade > b->idade)
         return 1;
      else
         return 0;
}

int buscaPesoMenor(int n, Dados ** v, int idade)
{
   int ini=0;
   int fim=n-1;
   int meio, cmp;
   while (ini <= fim)
   {
      meio=(ini+fim)/2;
      cmp=comparaIdade(idade,v[meio]);
      if (cmp<0)
         fim=meio-1;
      else
         if (cmp>0)
            ini=meio+1;
         else
         {  // busca o de menor peso
            while (meio > 0 && comparaIdade(idade,v[meio-1])==0)
               meio--;
            return meio;
         }
   }
   return -1;
}
```