Lista 8

1 Introdução

A ideia de um Tipo Abstrato de Dados (TAD) é esconder (abstrair) o dado do programador, assim como as funções e procedimentos escondem (abstraem) o código que faz as operações sobre os dados, no sentido que o importante é saber o que é feito, não como é feito.

É evidente que o programador tem acesso pleno aos dados, mas em programas complexos ele deve evitar alterá-los diretamente, mas sim passar por alterações controladas por chamadas de funções e procedimentos. Estas garantem a integridade dos dados, enquanto que o programador pode introduzir inconsistências quando acessa indevidamente alguma variável na memória.

Por exemplo, no exercício que segue, usaremos um vetor de inteiros cuja primeira posição armazena o tamanho deste vetor. Pois bem, todas as funções e procedimentos que usaremos cuidam muito bem da manutenção deste valor. Mas digamos que o programador "confiante" resolve alterar o valor da posição zero do vetor, por sua própria conta e assumindo a responsabilidade sobre isto, o que significa que ele poderá tornar o programa inconsistente. Sim, pois se ele escreve indevidamente na posição zero, vai alterar o tamanho do vetor.

Aqui estamos falando da diferença entre o que pode ser feito e o que não deve ser feito. Na arte de escrever programas é melhor pensar muito bem sobre o que não deve ser feito. Evidentemente o programador é soberano em suas decisões, mas via de regra, ele segue alguns princípios. Este é um deles.

2 O Tipo Abstrato de Dados Conjunto

No arquivo indicado abaixo nós fornecemos uma biblioteca de definições de tipos bem como de funções e procedimentos que operam sobre o TAD conjunto que é implementado em forma de um vetor cujos elementos estão ordenados em ordem crescente. A estrutura armazena o tamanho do vetor em sua posição zero. Apresentamos os códigos de todas as funções e procedimentos úteis para vários tipos de problemas diferentes, mas em particular para o problema que apresentaremos a seguir.

Como um bom TAD, as implementações fornecidas poderiam ser diferentes, por exemplo usando vetores não ordenados ou outras estruturas de dados que vocês conhecerão em algoritmos 2 ou programação 1.

Em todo caso, ao resolver este problema, vocês poderão usar esta biblioteca fornecida para resolver o problema, MAS SEM ALTERAR QUALQUER LINHA DO CÓDIGO FORNECIDO PARA AS FUNÇÕES E PROCEDURES. Nós garantimos que eles funcionam. Vocês poderão alterar apenas o programa principal, que está vazio no momento. Faz parte também do problema declarar as variáveis e eventualmente construir outras funções ou procedures que te ajudem a resolver o problema.

Em resumo, vocês poderão chamar qualquer função ou procedimento fornecido, mas não poderão alterar os seus códigos. Decidam quais são úteis ou não.

O importante é observar os protótipos das funções e procedimentos e saber utilizá-los para resolver o problema abaixo. Nós queremos que vocês apenas USEM as funções e procedimentos para resolver um problema.

Nós procuramos indicar nos comentários destas funções e procedimentos o comportamento delas, bem como uma indicação intuitiva do custo das operações para cada operação.

Claro que é um bom exercício ENTENDER os algoritmos fornecidos que manipulam a estrutura de dados, mas isso vocês podem fazer em um outro momento. Por exemplo, um

dia vocês podem tentar implementar uma procedure "diferença de conjuntos", mas não neste exercício, façam isso depois.

3 Como resolver o problema abaixo

Neste exercício vamos praticar o uso de Tipos Abstratos de Dados (TAD). Usaremos o TAD "tad_conjunto" que é fornecido na URL:

`http://www.inf.ufpr.br/cursos/ci055/tad_conjunto/vingadores.pas`

Você deve baixar este arquivo em algum diretório da sua conta pessoal. O importante deste exercício é que você saiba USAR as chamadas para as funções e procedimentos fornecidos para resolver o problema que segue abaixo. Claro que não precisa usar todas elas, escolha as que são pertinentes para o teu algoritmo. Se precisar, implemente outras. Em suma, vocês devem entender como resolver um problema usando conjuntos! Simples assim.

Também fornecemos dois casos de teste com as respectivas saídas esperadas para você testar o seu programa:

- `http://www.inf.ufpr.br/cursos/ci055/tad_conjunto/avengers-entrada1.txt`
- `http://www.inf.ufpr.br/cursos/ci055/tad_conjunto/avengers-saida.txt`
- `http://www.inf.ufpr.br/cursos/ci055/tad_conjunto/avengers-entrada2.txt`
- `http://www.inf.ufpr.br/cursos/ci055/tad_conjunto/avengers-saida2.txt`

Para testar seu programa, baixe estes arquivos, compile seu programa e rode o seguinte comando:

```
./vingadores < avengers-entrada1.txt | diff - avengers-saida1.txt
./vingadores < avengers-entrada2.txt | diff - avengers-saida2.txt
```

Se a saída deste comando imprimir qualquer coisa é porque seu programa está errado. A saída que não imprime nada indica que o programa está certo.

4 O problema

Um grupo de super-heróis recebeu um chamado e precisa montar uma equipe para ir resolver o problema. Cada um dos membros do grupo tem uma série de poderes, como super força, super velocidade, super visão etc. O chamado indicava uma lista de habilidades que pelo menos um dos membros da equipe enviada deveria ter. Caso uma habilidade não seja contemplada a missão falhará. O grupo de super-heróis preparou uma lista de possíveis equipes. Nossa tarefa, como parte da equipe de TI do grupo, é encontrar qual destas possíveis equipes satisfaz os critérios e tem o menor tamanho.

Você deve ler, inicialmente, uma lista de habilidades para cada super-herói. Os super-heróis são numerados, a partir de 1. As habilidades são números inteiros. Uma lista de habilidades termina com um 0 (zero). A entrada, desta parte, consistirá de uma sequência de listas de habilidades, terminando com uma lista vazia (que não é de nenhum super-herói).

Após esta parte, você deve ler os requisitos da missão, que também são descritos por uma lista de habilidades, nos mesmos moldes da parte anterior.

Em seguida você deve ler uma lista de equipes, onde cada equipe também é uma sequência de números (de cada super-herói) terminada com 0 (zero). A lista de equipes termina com uma equipe vazia.

Como saída você deve escrever a lista dos super-heróis da equipe que melhor satisfaz os requisitos. Caso nenhuma equipe satisfaça, você deve escrever "NENHUMA".

```
Exemplo de entrada:
1 2 3 4 0                   // super-herói 1
2 4 8 20 17 0               // super-herói 2
99 5 12 6 0                 // super-herói 3
1000 0                      // super herói 4
4 9 3 99 0                  // super-herói 5
0
4 99 1000 0                 // habilidades requeridas
1 2 3 0                     // equipe 1
1 2 3 4 5 0                 // equipe 2
2 4 5 0                     // equipe 3
5 0                         // equipe 4
4 5 0                       // equipe 5
1 2 5 0                     // equipe 6
0

Saída correspondente:
4 5
```