Tipo abstrato de dado

Origem: Wikipédia, a enciclopédia livre.

Em computação, o **tipo abstrato de dado** (**TAD**) é uma especificação de um conjunto de dados e operações que podem ser executadas sobre esses dados. Além disso, é uma metodologia de programação que tem como proposta reduzir a informação necessária para a criação/programação de um algoritmo através da abstração das variáveis envolvidas em uma única entidade fechada, com operações próprias à sua natureza.

Um exemplo prático disto é o de um estudante. Em um projeto anterior à teoria de TAD, um estudante seria representado por variáveis soltas (como seu nome, sua idade e sua matrícula) que seriam operadas separadamente, sem ligação lógica entre elas além do conhecimento do programador de que a variável trata-se do nome da "entidade" estudante.

Conceitualmente um programa passou a ser projetado pensando que não há o nome, idade e matrícula do estudante, mas simplesmente o **tipo** estudante. Este tipo, como um tipo simples (inteiro ou string), deve ter operadores próprios. Assim, o estudante deve possuir operações desejáveis ao programador, como duplicar sua informação, validar a matrícula, verificar a idade, etc.

Na prática, o TAD é implementado usando-se um tipo composto (*struct/record* - estrutura/registro) com os valores pertencentes ao TAD (nome, idade, matrícula). E por funções que operam esta estrutura.

Exemplo:

Alem do exemplo abaixo exitem os TAD básicos e padrão; Listas, Pilhas e Filas.

```
estrutura Estudante{
   Nome
   Idade
   Matricula
}

funcao Estudante_MaiorDeIdade(Estudante estudante) retorna booleano;
funcao Estudante_ValidaMatricula(Estudante estudante);
```

A abstração de informações através do TAD permitiu a melhor compreensão dos algoritmos e maior facilidade de programação, e por consequência aumentou a complexidade dos programas, tornando-se fundamental em qualquer projeto de software a modelagem prévia de seus dados.

Um dos problemas que são enfrentados no TAD é que ele é uma estrutura metafórica gerada pela modelagem. Porém, em nível de implementação, não há nenhuma segurança que as operações e regras de operação desejadas para este tipo serão respeitadas.

Posteriormente, essa metodologia foi incorporada à própria linguagem de programação para um protótipo do que é hoje a orientação a objetos (OOP - Object Oriented Programming), permitindo o controle do acesso às informações de um tipo, a herança e o polimorfismo.

Veja também

- Programação estruturada
- Orientação a objetos
- Engenharia de software
- Modelagem de dados
- Representação de conhecimento