Tipos Abstratos de Dados

Luiz Chaimowicz, Raquel O. Prates, Gisele L. Pappa e Ítalo Cunha

Algoritmos e estruturas de dados

- Algoritmo: sequência de ações executáveis para realizar uma tarefa
  - Trabalham sobre estruturas de dados
- Estruturas de dados: representam uma situação real de forma abstrata
  - Precisam suportar as operações do algoritmo

Representação de dados

- Dados podem estar representados (estruturados) de diferentes maneiras
  - Normalmente, a escolha da representação é determinada pelas operações realizadas sobre os dados
- Exemplo: números inteiros
  - Representação por palitos: || + |||| = ||||||
  - Representação decimal: 1278 + 321 = 1599

Representação de dados – exemplo

- Como representar tempo?
  - Depende das operações a serem realizadas

- **`time_t`**, tipo inteiro, conta o número de segundos desde 1/1/1970
- **`struct timespec`**, struct com dois inteiros, um **`time_t`** e um contador de nanosegundos
- **`struct tm`**, vários campos para segundo, minuto, dia, mês, ano, horário de verão, etc.
- **`double`**, pra armazenar intervalos de tempo

Representação de dados – exemplo

- Strings
  - Arranjo de **`char`**
    - Tabela ASCII, tabela EBCDIC
  - Arranjo de **`int`**
    - Unicode, representação de vários alfabetos
  - Figura bitmap

Qual a diferença entre um programa e um algoritmo?

- Um programa é uma realização concreta de um algoritmo abstrato, baseado em representações de dados específicas

- Programas precisam ser implementados numa linguagem que pode ser entendida e seguida pelo computador

Tipos Abstratos de Dados

- Encapsulam a representação dos dados e as operações que podem ser realizadas sobre eles
- Usuário do TAD **vs.** programador do TAD
  - Usuário só "enxerga" a interface, não a implementação
  - Não importa se a representação é feita com palitos, números decimais, ou em binário desde que a gente consiga somar, subtrair, multiplicar, etc.
- Os usuários de um TAD só têm acesso às operações disponibilizadas sobre os dados

TADs – Isolamento e reuso

- Podemos modificar a implementação do TAD sem modificar o código que usa o TAD
- E vice-versa, podemos modificar o código que usa o TAD sem modificar a implementação do TAD
- TAD pode ser reaproveitado em vários programas ou módulos

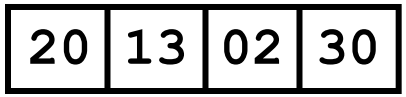
Exemplo de TAD: lista de números

- Operações:
  - Criar uma nova lista vazia
  - Inserir um número no final da lista

Programa usuário do TAD:

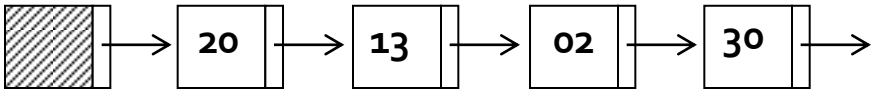
```
int main() {
    Lista *L;
    int x = 20;

    L = cria_lista();
    insere(L, x);
    ...
}
```

TAD implementado com vetor:




```
void insere(Lista *l, int x) {
    l->arranjo[...] = x;
    ...
}
```

TAD implementado com lista:




```
void insere(Lista *l, int x) {
    Celula *c = cria_celula(x);
    l->ultimo = c;
}
```

Implementação de TADs

- Em linguagens orientadas a objeto (C++, Java), implementação é feita com classes
  - Disciplinas posteriores
- Em linguagem estruturadas como C, implementação é feita com definições de tipos e implementação de funções

Definição de tipos

- Para simplificar, tipos complexos podem ser definidos como novos tipos
  - Não exagerar

```
typedef struct {
   char nome[48];
   int matricula;
   char conceito;
} Aluno;
typedef struct {
   int dia;
   int mes;
   int ano;
} data_t;
```

```
// Utilidade duvidosa:
typedef int[10] Vetor;
typedef char[48] Nome;
```

Modularização

- TADs são bons candidatos para se tornarem módulos
  - Implementação no .c
  - Usuário só precisa conhecer .h

Exemplo – TAD conta bancária

- Implemente um TAD conta bancária com campos número e saldo que suporte as as seguintes operações:
  - Iniciar uma nova conta com um número e saldo
  - Depositar um valor na conta
  - Sacar um valor da conta
  - Imprimir o saldo
- Teste o TAD