Tipos Abstratos de Dados
TADs

Renata Galante
Viviane Moreira

Universidade Federal do Rio Grande do Sul
Instituto de Informática

INF 01203 – Estruturas de Dados

Exemplo

- Folha de freqüência

Disciplina: INF 01203 – Estruturas de Dados
Semestre: 2009-2 **Turmas:** C
Professor: Renata Galante

matricula	nome	...	...	...
XXXX	Ana			
ZZZZ	Maria			
YYYY	Pedro			

Operações:
- Insere
- consulta
- excluir
- altera
- calculaMedia

Programa: *manipula* dados a respeito dos alunos matriculados

Modelagem de Dados

Identificação dos Dados
Abstração do mundo real

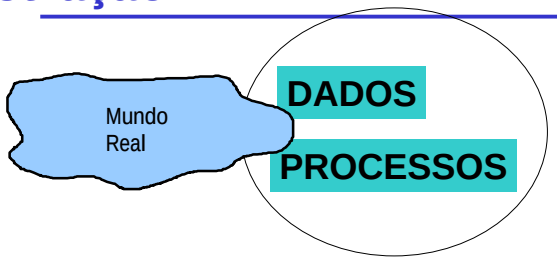
MODELO LÓGICO
Relacionamentos entre os dados
DADOS + OPERAÇÕES

MODELO FÍSICO
Alternativas de implementação
ALGORITMOS PARA AS OPERAÇÕES

Conceito de Abstração

- Representação de uma entidade onde apenas os atributos mais significativos em um dado contexto são apresentados;
- Permite representar uma coleção de entidades;
- Arma contra a complexidade em programação:
  - permite focar nos atributos essenciais e ignorar os não essenciais

Abstração



Informações relevantes para o sistema

Omissão de detalhes

- **abstração de dados**
- **abstração de processos**

Abstração de Processos

- Funções e procedimentos:
  - **função**: abstração de um valor (expressão)
    Exemplo: `x:= y + sqrt(z) / 3;`
  - **procedimento**: abstração de um comando
    Exemplo: ImprimeCabecalho;
  - **utilização**: invocação parametrizada ou não

```
meu programa:
        comando 1;
        comando 2;
        valor:= minhafuncao(10);
        ...
        comando n;
fim.
```

```
minhafuncao(par);
....comandos...
```

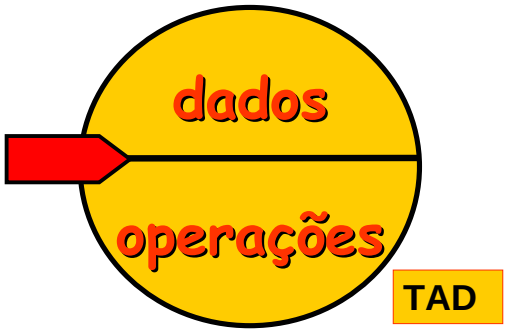

Abstração de Dados

- Abstraem a representação e as operações
- Tipos de abstração de dados:
  - Tipos de dados primitivos
  - Tipos de dados estruturados

Abstração de Dados

- Abstraem a representação e as operações
- Tipos de abstração de dados:
  - Tipos de dados primitivos
  - Tipos de dados estruturados
  - **Tipos abstratos de dados (TAD)**

Abstração de Dados



Tipos Abstratos de Dados

- Um TAD é uma forma de definir um novo tipo de dado juntamente com as operações que manipulam esse novo tipo de dado

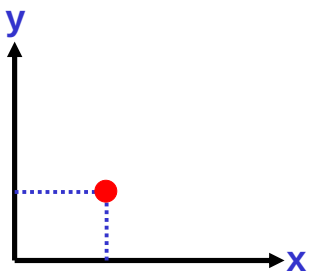
Tipos Abstratos de Dados

- **Um tipo abstrato de dados (em LP) é um tipo de dado que satisfaz as condições:**
  - **A representação ou a definição do tipo e as operações sobre variáveis desse tipo estão contidas numa única unidade sintática**
    - **MÓDULO**
  - **A representação interna do tipo (a implementação) não é visível de outras unidades sintáticas, de modo que só as operações oferecidas na definição do tipo podem ser usadas com as variáveis desse tipo**

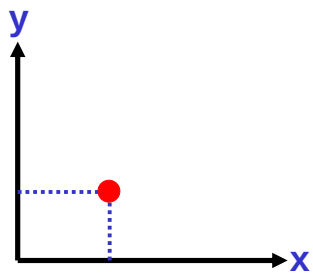
Para que servem?

- **Esquecer a forma de implementação (tipo concreto)**
- **Separar o código da aplicação do código da implementação**
- **Vantagens:**
  - **usar o mesmo tipo em diversas aplicações**
  - **pode-se alterar o tipo concreto usado sem alterar a aplicação**
- **REUTILIZAÇÃO**

Exemplo - Representar um Ponto

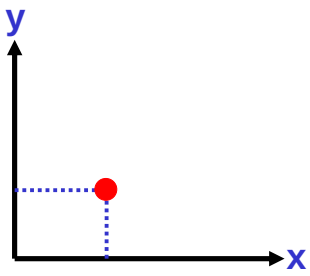


Exemplo - Representar um Ponto



- **Modelo**
  Par ordenado (x,y)
- **Dados representando o modelo**
  - **Coordenada X**
  - **Coordenada Y**

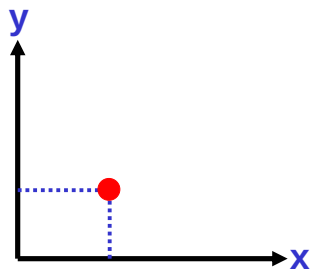
Exemplo - Representar um Ponto



Operações:

- **cria: operação que cria um ponto, alocando memória para as coordenadas x e y;**
- **libera: operação que libera a memória alocada por um ponto;**
- **acessa: operação que devolve as coordenadas de um ponto;**
- **atribui: operação que atribui novos valores às coordenadas de um ponto;**
- **distancia: operação que calcula a distância entre dois pontos.**

Exemplo - Representar um Ponto



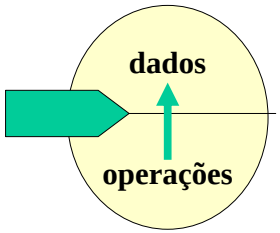
Operações:

- **cria (x,y)**
- **libera (ponto P)**
- **acessa (ponto P)**
- **atribui (ponto P, x,y)**
- **distancia (ponto P1, ponto P2)**

TADs: propriedades

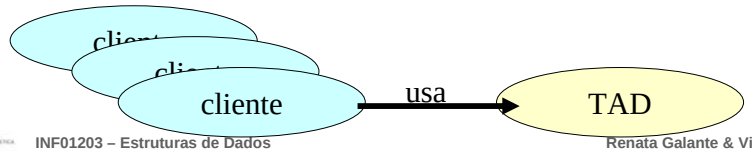
- Satisfazem as propriedades de
  - encapsulamento: definição isolada de outras unidades do programa
  - invisibilidade e proteção: representação do tipo deve ser acessada somente no ambiente encapsulado

- A LP deve possibilitar
  - ambiente encapsulado
  - proteção de dados
  - interface para acesso
  - Operações básicas



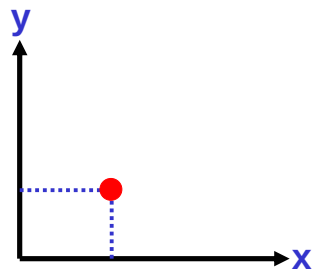
Vantagens – proteção de dados

- Código do cliente do TAD não depende da implementação
- A representação do tipo pode ser alterada sem ser necessário notificar o cliente
- Segurança:
  - clientes não podem alterar a representação
  - clientes não podem tornar os dados inconsistentes



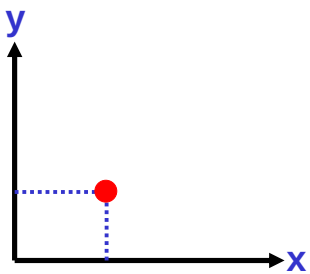
Implementação em C

Exemplo - Representar um Ponto



- **Modelo**
  Par ordenado (x,y)
- **Dados representando o modelo**
  - **Coordenada X**
  - **Coordenada Y**

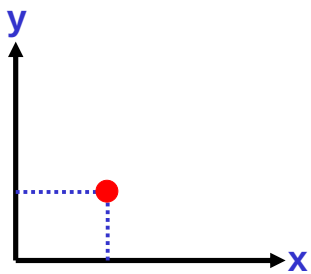
Exemplo - Representar um Ponto



Operações:

- **cria: operação que cria um ponto, alocando memória para as coordenadas x e y;**
- **libera: operação que libera a memória alocada por um ponto;**
- **acessa: operação que devolve as coordenadas de um ponto;**
- **atribui: operação que atribui novos valores às coordenadas de um ponto;**
- **distancia: operação que calcula a distância entre dois pontos.**

Exemplo - Representar um Ponto



Operações:

- **cria (x,y)**
- **libera (ponto P)**
- **acessa (ponto P)**
- **atribui (ponto P, x,y)**
- **distancia (ponto P1, ponto P2)**

TAD em C

- Interface (PROTÓTIPO)
  - # include "arquivo.h" -> *TAD usuários*
  - # include <arquivo.h> -> *TAD linguagem C*
- Programa.c
  - Código fonte

Exemplo - Implementação C

```
/* TAD: Ponto (x,y) */
/* Tipo exportado */
typedef struct ponto Ponto;

/* Funções Exportadas */

/* Cria coordenada */
Ponto* pto_cria (float x, float y);

/* Libera ponto */
void pto_libera(Ponto* p);

/* Acessa ponto */
void pto_acessa(Ponto* p, float* x, float* y);

/* Atribui */
void pto_atribui(Ponto* p, float x, float y);

/* Distância */
float pto_distancia(Ponto* p1, Ponto* p2);
```

ponto.h

Interface, chamada protótipo

Exemplo - Implementação C

```
# include <stdio.h>
# include "ponto.h"

struct ponto {
   float x;
   float y;
};

Ponto* pto_cria (float x, float y)
{
   Ponto* p = (Ponto*) malloc(sizeof(Ponto));
   if (p == NULL) {
      printf("Memória Insuficiente!"\n);
      exit(1);
}
p->x = x;
p->y = y;
return p;
}
```

ponto.C

Implementação da Interface

Exemplo - Implementação C

```
# include <stdio.h>
# include "ponto.h"

int main (void)
{
   Ponto* p = pto_cria(2.0, 1.0);
   Ponto* q = pto_cria(3.4, 2.1);
   float d = pto_distancia(p,q);
   printf("Distância entre ponto: %\n", d);
   pto_libera(q);
   pto_libera(p);
   return 0;
}
```

Aplicacao.C

Aplicação que usa a **Interface**

Questão

- Se um benfeitor fornecer a você uma INTERFACE ***ponto.h***, mas você pode ler apenas a interface. Você não pode ler as seções de implementação.
  - **Você pode escrever um programa que utilize esta INTERFACE *ponto*?**

Questão

- Se um benfeitor fornecer a você uma INTERFACE ***ponto.h***, mas você pode ler apenas a interface. Você não pode ler as seções de implementação.
  - **Você pode escrever um programa que utilize esta INTERFACE *ponto*?**
  - **SIM! A interface é suficiente para declarar as estruturas e variáveis *ponto*. Você também conhece os cabeçalhos e especificações de cada operação (*funções*).**

Bibliografia

- **1. Estruturas de Dados.** Nina Edelweiss, Renata Galante, Editora Bookman – Série de Livros Didáticos Informática UFRGS, 2009, Primeira Edição.
  - Capítulo 2
  - Exercícios: página 44 – exercícios de 1 a 6