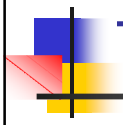
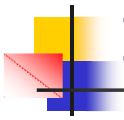
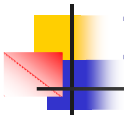


Tipos Abstratos de Dados

Prof. Rui Jorge Tramontin Jr.

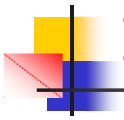


Índice



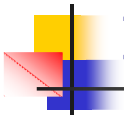
Introdução

- Algoritmo é uma sequência de instruções que manipulam estruturas de dados
  - Algoritmos visam resolver problemas.
- Logo, um algoritmo e suas estruturas de dados representam uma abstração (modelo) de uma situação real.
- Abstração é um conceito fundamental para solução de problemas.
  - → Suprimir detalhes irrelevantes, enfatizar os relevantes.



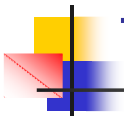
Introdução

- Como modelar um problema?
  - Depende da linguagem de programação utilizada.
- Linguagens de programação oferecem tipos predefinidos (simples ou estruturados).
- Exemplo: dados de um aluno (nome, curso, etc.)
  - Variáveis simples para cada atributo:
    - `char nome[30]; char curso[20]; ...`
  - Tipos estruturados:
    - `struct Aluno {char nome[30];...};`



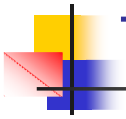
Introdução

- Além do repertório de tipos nativos, linguagens oferecem meios para a definição de tipos abstratos.
- Tipos abstratos:
  - São tipos novos criados pelo usuário.
  - Estendem o conjunto padrão de tipos.
  - Definem um conjunto de operações.
  - São definidos por usuários, na biblioteca padrão ou em bibliotecas de fornecedores diversos.



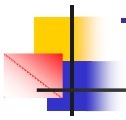
Tipo Abstrato de Dados (TAD)

- **Definição:** um TAD é definido como:
  - Uma coleção bem definida de dados a serem armazenados (estrutura de dados), e;
  - Um grupo de operadores que são aplicados para manipulação desses dados;
- Os valores armazenados devem ser manipulados **EXCLUSIVAMENTE** pelos operadores do TAD.



Tipo Abstrato de Dados (TAD)

- A especificação de um TAD descreve:
  - **quais dados** podem ser armazenados, e;
  - o que é possível fazer com esses dados através dos **operadores** do TAD.
- O TAD não descreve **como** as operações serão efetivamente implementadas.
- Operações podem ser implementadas de diferentes maneiras.



Tipo Abstrato de Dados (TAD)

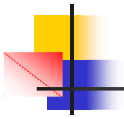
- O TAD encapsula a estrutura de dados. Os usuários do TAD usam operações para manipular os dados.
- Usuário do TAD x Programador do TAD
  - Usuário só "enxerga" a interface, não a implementação.
- Usuários podem abstrair a implementação específica.
  - Modificações na implementação ficam restritas ao TAD.

Exemplos de TADs

Mundo Real	Dados	Operações
Pessoa	▪idade da pessoa	▪nasce (idade = 0) ▪aniversário (idade = idade +1)
Fila de Espera	▪nome da pessoa ▪posição na fila	▪entra na fila (no fim) ▪sai da fila (o primeiro)
Cadastro de Funcionários	▪nome, cargo, salario	▪Cadastra funcionário ▪Exclui funcionário ▪modifica (cargo ou salário)
Conta Bancária	▪número da conta ▪saldo	▪saca ▪deposita ▪transfere

Exemplo de TAD: Strings em C

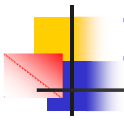
- A linguagem C não possui um tipo primitivo para representar *strings*.
- Strings em C são arrays de caracteres.
- Biblioteca "string.h" define funções para manipulação de strings:
  - **strcpy()** → cópia
  - **strlen()** → tamanho
  - **strcmp()** → comparação
  - **strcat()** → concatenação



Exemplo de TAD: listas

- Um TAD "lista" oferece as seguintes operações:
  - Cria lista vazia;
  - Insere número no fim da lista;
  - Remove de uma posição i.
- Operações são independentes da implementação:
  - Arrays;
  - Listas encadeadas.

```
int main() {
  Lista lista;
  int x = 20;
  criaListaVazia(lista);
  insere(x, lista);
  ...
}
```

Implementação de TADs

- Em linguagens orientadas a objetos (C++, Java)
  - Implementação é feita através de classes;
- Em linguagens estruturadas (C, Pascal)
  - Implementação é feita pela definição de tipos e implementação de funções;
- Foco da disciplina é linguagem C:
  - Tipos de dados definidos por **typedef e structs**;
  - **Funções** de manipulação desses tipos de dados;
  - Ambos são agupados em **módulos** (arquivos ".h" e ".c").

TADs em C: tipos de dados

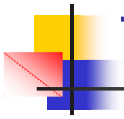
- Novos tipos de dados são definidos usando o comando **typedef**:

```
typedef struct
{
   char nome[100];
   int matricula;
   char conceito;

} TipoAluno;

typedef int[10] Vetor;
```

```
int main()
{
  TipoAluno aluno;
  Vetor v;

  aluno.nome = "João";
  ...
}
```

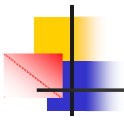

TADs em C: funções

- Além da definição de tipos, um TAD precisa de um conjuntos de que agem sobre tais tipos.
- Essas operações são implementadas na forma de **funções**.
- Como boa regra de programação, evita-se acessar os dados diretamente, fazendo o acesso só através das funções.
  - Em C, não há uma forma de proibir o acesso direto!



TADs em C: módulos

- Uma boa técnica de programação é implementar os TADs em arquivos separados do programa principal.
- Para isso geralmente separa-se a declaração e a implementação do TAD em dois arquivos:
  - NomeDoTAD.h : com a declaração do TAD (interface);
  - NomeDoTAD.c : com a implementação.
- O programa ou outros TADs que utilizam o seu TAD devem dar um **#include** no arquivo ".h".



Exemplo de TAD em C

- Modelo de uma conta bancária.
- O TAD se chama ContaBancaria, com os campos número e saldo.
- Clientes podem fazer as seguintes operações:
  - Iniciar uma conta com um número e saldo inicial;
  - Depositar um valor;
  - Sacar um valor;
  - Imprimir o saldo.

ContaBancaria.h

```
// definição do tipo
typedef struct {
   int numero;
   double saldo;
} ContaBancaria;

// cabeçalho das funções
void inicializa(ContaBancaria* conta, int numero,
double saldo);
void deposito(ContaBancaria* conta, double valor);
void saque(ContaBancaria* conta, double valor);
void imprime(ContaBancaria conta);
```


ContaBancaria.c

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

void inicializa(ContaBancaria* conta, int numero, double saldo)
{
   conta->numero = numero;
   conta->saldo = saldo;
}
void deposito(ContaBancaria* conta, double valor)
{
   conta->saldo += valor;
}
void saque(ContaBancaria* conta, double valor)
{
   conta->saldo -= valor;
}
void imprime(ContaBancaria conta)
{
   printf("Numero: %d\n", conta.numero);
   printf("Saldo: %f\n", conta.saldo);
}
```

Main.c

```
#include <stdio.h>
#include <stdlib.h>
#include "ContaBancaria.h"

int main (int argc, char **argv)
{
  ContaBancaria conta;

  inicializa(&conta, 918556, 300.00);
  printf("\nAntes da movimentacao:\n ");
  imprime(conta);

  deposito(&conta, 50.00);
  saque(&conta, 70.00);
  printf("\nDepois da movimentacao:\n ");
  imprime(conta);
}
```


Exercício

- Faça um TAD que represente um aluno.
- O tipo de dados deverá armazenar o nome, curso, fase, um vetor de notas e a quantidade de notas.
- O TAD deve possuir funções para:
  - Inicializar dados do aluno: recebe como parâmetros uma variável do tipo Aluno (por referência), o nome, curso e fase do aluno, e define a quantidade de notas = 0;
  - Adicionar uma nota: coloca a nota (passada de parâmetro) no vetor (na posição definida pelo campo "quantidade"), e incrementa a quantidade;
  - Calcular a média: retorna a média do aluno, com base nas suas notas.
  - Imprimir dados do aluno.