UFMG

Tipo Abstrato de Dados

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

(versão adaptada)

Livro “Projeto de Algoritmos”

Capítulo 1

3 Pontos Principais

- Algoritmo e Programa
- Tipo Abstrato de Dados
  - Qual papel do programador e do usuário do TAD
- Conceitos de typedef e struct

Qual a diferença entre um algoritmo e um programa?

Algoritmos e Estruturas de Dados

- **Algoritmo:**
  - Sequência de ações executáveis para a solução de um determinado tipo de problema
  - Exemplo: “Receita de Bolo”
  - Algoritmos trabalham sobre Estruturas de Dados

- **Estrutura de Dados**:
  - Conjunto de dados que representa uma situação real (estado do mundo)
    - Modo eficiente de armazenamento para facilitar seu acesso e modificação.

Representação dos Dados

- Os dados podem estar representados (estruturados) de diferentes maneiras
- Normalmente, a escolha da representação é determinada pelas operações que serão utilizadas sobre eles
- Exemplo: números inteiros
  - Representação por palitinhos: II + IIII = IIIIII
    - Boa para pequenos números (operação simples)
  - Representação decimal: 1278 + 321 = 1599
    - Boa para números maiores (operação complexa)

Programas

- Um programa é uma formulação concreta de um algoritmo abstrato, baseado em representações de dados específicas
- Os programas são feitos em alguma linguagem que pode ser entendida e seguida pelo computador
  - Linguagem de máquina
  - Linguagem de alto nível (uso de compilador)
    - Aqui vamos utilizar a Linguagem C

Linguagem C

- Criada em no início da década de 70 para a programação do sistema operacional Unix
- Uma das linguagens mais utilizadas no mundo, e serviu como base para outras como C++, C#, etc.

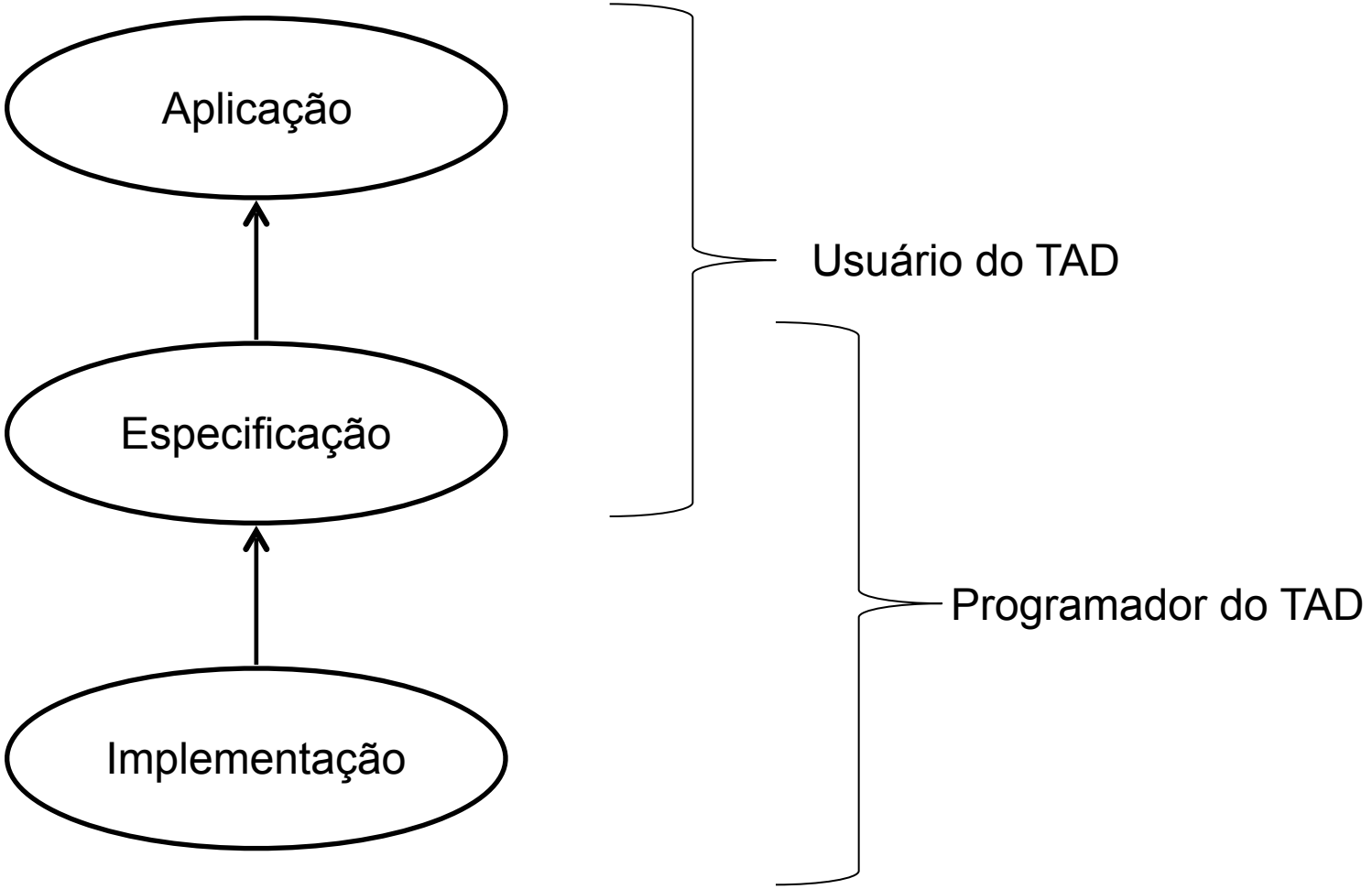
Tipos Abstratos de Dados (TADs)

- Agrupa a ***estrutura de dados*** juntamente com as ***operações*** que podem ser feitas sobre esses dados
- O TAD encapsula a estrutura de dados. Os usuários do TAD só tem acesso a algumas operações disponibilizadas sobre esses dados
- Usuário do TAD x Programador do TAD
  - Usuário só “enxerga” a interface, não a implementação

Tipos Abstratos de Dados (TADs)

- Dessa forma, o usuário pode abstrair da implementação específica.
- Qualquer modificação nessa implementação fica restrita ao TAD
- A escolha de uma representação específica é fortemente influenciada pelas operações a serem executadas

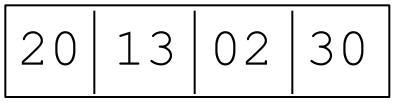
Tipos Abstratos de Dados (TADs)



Exemplo: Lista de números inteiros

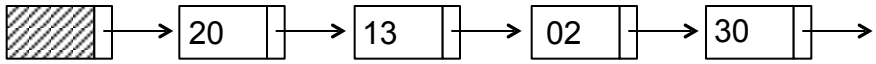
- Operações
  - Faz Lista Vazia
  - Insere número no começo da lista
  - Remove de uma posição i

Implementação por Vetores:




```
void Insere(int x, Lista L) {
   for(i=0;...) {...}
   L[0] = x;
}
```

Implementação por Listas Encadeadas




```
void Insere(int x, Lista L) {
    p = CriaNovaCelula(x);
    L.primeiro = p;
    ...
}
```

Programa usuário do TAD:

```
int main() {
   Lista L;
   int x;

   x = 20;
   FazListaVazia(L);
   Insere(x,L);
   ...
}
```

Implementação de TADs

- Em linguagens orientadas por objeto (C++, Java) a implementação é feita através de classes
- Em linguagens estruturadas (C, pascal) a implementação é feita pela definição de tipos juntamente com a implementação de funções
- Vamos utilizar os conceitos em C (**typedef** e **struct**)

Estruturas (Structs) em C

- Uma estrutura é uma coleção de uma ou mais variáveis, possivelmente de tipos diferentes, colocadas juntas sob um único nome para manipulação conveniente
- Exemplo:
  - para representar um aluno são necessárias as informações nome, matrícula, conceito
  - Ao invés de criar três variáveis, é possível criar uma única variável contendo três campos.
- Em C, usa-se a construção **struct** para representar esse tipo de dado

Estruturas (Structs) em C

- Sintaxe:

  struct nome **{**

  [tipo nome_da_variável] ;

  ...

  } [variável] ;

```
struct Aluno {
    string nome;
    int matricula;
    char conceito;
};
```

Estruturas (Structs) em C

```
#include<iostream>
#include<string>


struct Aluno {
    string nome;
    int matricula;
    char conceito;
};

main() {
    struct Aluno al, aux;

    al.nome = "Pedro"
    al.matricula = 200712;
    al.conceito = 'A';
    aux = al;
}
```

al:

Pedro	
200712	A

aux:

Pedro	
200712	A

Estruturas (Structs) em C

```
struct Aluno {
    string nome;
    int matricula;
    char conceito;
};

struct Professor{
    string nome;
    int matricula;
    string classes[3];
};

main() {
    struct Aluno al;
    struct Professor pr;

    al.nome = "Pedro";
    pr.nome = "José";
}
```

```
main() {
string alunoNome;
int alunoMatricula;
Char alunoConceito;
string alunoNome2;
int alunoMatricula2;
Char alunoConceito2;
string professorNome;
int professorMatricula;
string professoClasses[3];


 alunoNome = "Pedro"
 alunoMatricula = 200712;
 alunoConceito = 'A';
 alunoNome2 = alunoNome;
 alunoMatricula2 = alunoMatricula;
 alunoConceito2 = alunoConceito;
}
```

Declaração de Tipos

- Para simplificar, uma estrutura ou mesmo outros tipos de dados podem ser definidos como um novo tipo
- Uso da construção **typedef**
- Sintaxe: **typedef** tipo identificador;

```
typedef struct {
    string nome;
    int matricula;
    char conceito;
} TipoAluno;

typedef int Vetor[10];
```

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

    ...
}
```

TADs em C

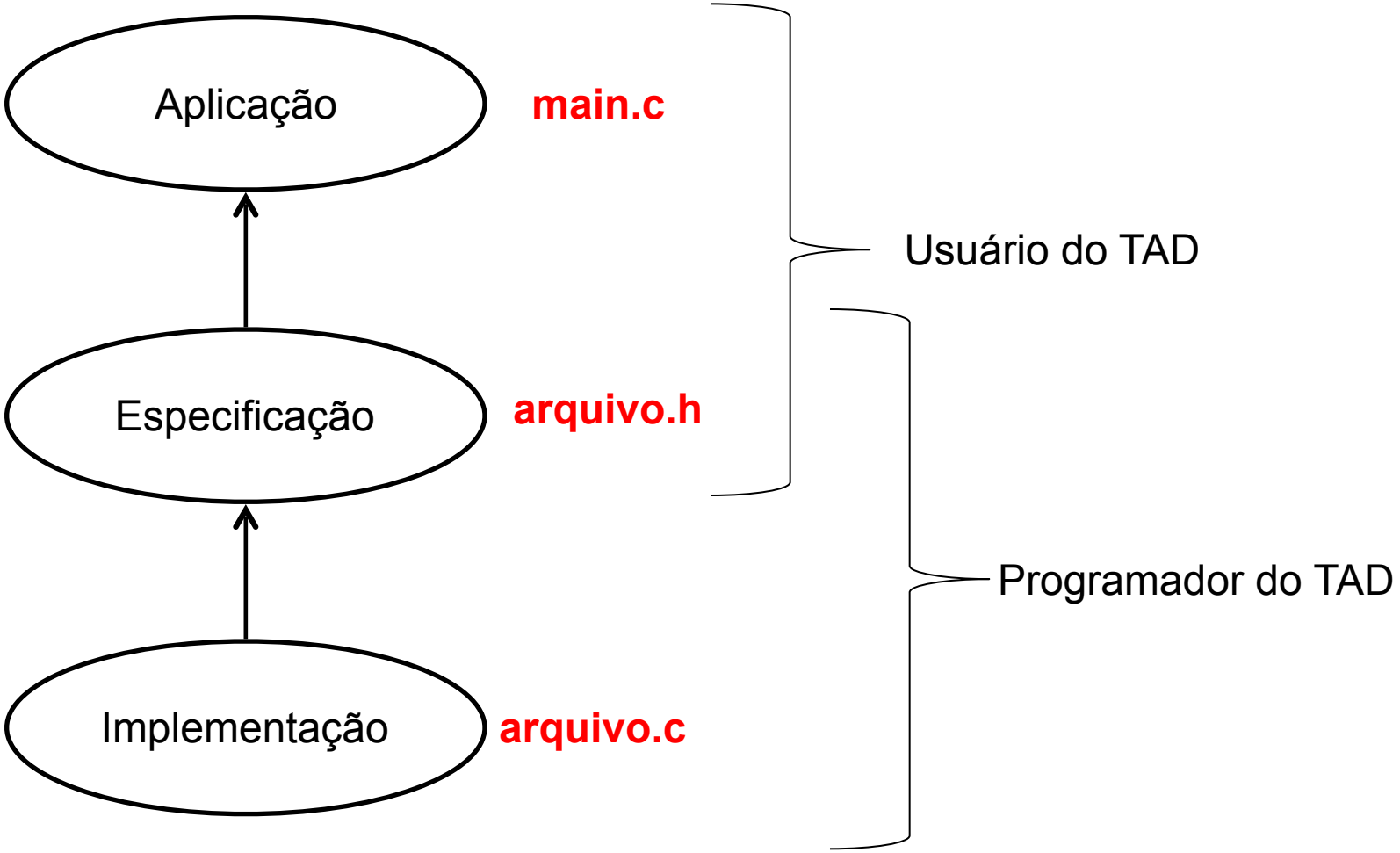
- Para implementar um Tipo Abstrato de Dados em C, usa-se a definição de tipos juntamente com a implementação de funções que agem sobre aquele tipo

- Como boa regra de programação, evita-se acessar o dado diretamente, fazendo o acesso só através das funções
  - Mas, diferentemente de C++ e Java, não há uma forma de proibir o acesso.

TADs em C

- 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
  - NomeDoTAD.c : com a implementação
- O programa ou outros TADs que utilizam o seu TAD devem dar um #include no arquivo .h

TADs em C


Aplicação
main.c
Especificação
arquivo.h
Implementação
arquivo.c
Usuário do TAD
Programador do TAD

Exemplo

- Implemente um TAD ContaBancaria, com os campos número e saldo onde os clientes podem fazer as seguintes operações:
  - Iniciar uma conta com um número e saldo inicial
  - Depositar um valor
  - Sacar um valor
  - Imprimir o saldo

- Faça um pequeno programa para testar o seu TAD

ContaBancaria.h

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

// cabeçalho das funções
ContaBancaria Inicializa (int, double);
void Deposito (ContaBancaria *, double);
void Saque (ContaBancaria *, double);
void Imprime (ContaBancaria);
```

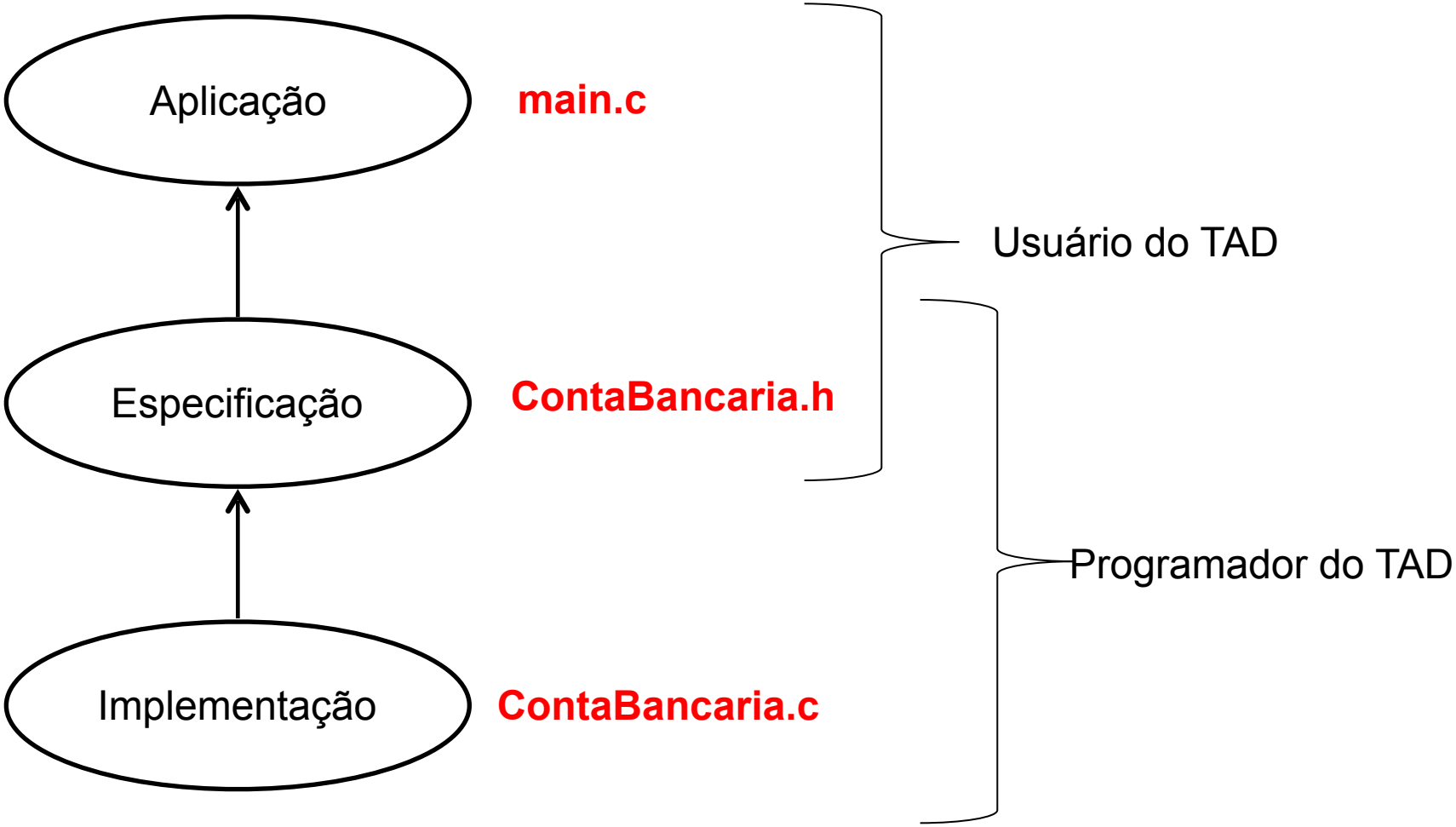
ContaBancaria.c

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

ContaBancaria Inicializa(int numero, double saldo) {
ContaBancaria conta;
       conta.numero = numero;
       conta.saldo = saldo;
       return conta;
}

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 - saldo: %f\n", conta.numero, conta.saldo);
}
```

Main.c

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

int main (void) {
ContaBancaria conta1;

    conta1 = Inicializa(918556, 300.00);
    printf("\nAntes da movimentacao:\n ");
    Imprime(conta1);

    Deposito(&conta1, 50.00);
    Saque(&conta1, 70.00);
    printf("\nDepois da movimentacao:\n ");
    Imprime(conta1);

    return(0);
}
```

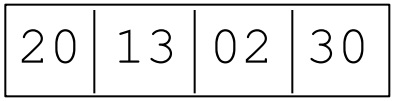
TADs em C


Aplicação
main.c
Especificação
ContaBancaria.h
Implementação
ContaBancaria.c
Usuário do TAD
Programador do TAD

TADs em C

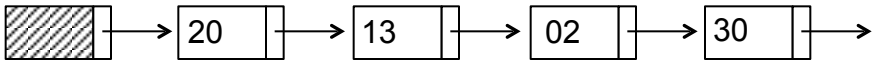
- Acesso direto dos dados (**errado!**)

Implementação por Vetores:




```
typedef struct {
   int dado[100];
} Lista;
```

Implementação por Listas Encadeadas




```
typedef struct item {
   int dado;
   struct item *prox;
} Item;

typedef struct {
   Item *inicio;
} Lista;
```

Programa usuário do TAD:

```
int main() {
   Lista L;

   L.dado[0] = 20;
}
```

Exercício

- Implemente um TAD **Número Complexo**
  - cada número possui os campos real e imaginário
  - Implemente as operações:
    - Inicializa: atribui valores para os campos
    - Imprime: imprime o número da forma “R + Ci”
    - Copia: Copia o valor de um número para outro
    - Soma: Soma dois números complexos
    - EhReal: testa se um número é real

- Faça um pequeno programa para testar o seu TAD