UNIVERSIDADE FEDERAL DO PARÁ
INSTITUTO DE CIÊNCIAS EXATAS E NATURAIS
FACULDADE DE COMPUTAÇÃO

Estrutura de Dados I / Projeto de Algoritmos I

Prof. Denis Rosário
Email: denis@ufpa.br

Agenda

- **Unidade 1: Conceito de algoritmos, estruturas de dados e programas**
  - Tipos de dados e tipos abstratos de dados
  - Como medir o tempo de execução de um programa
  - Introdução técnicas de análise de algoritmos

Algoritmos, Estrutura de Dados e Programas

- Os algoritmos fazem parte do dia-a-dia das pessoas. Exemplos de algoritmos:
  - instruções para o uso de medicamentos,
  - indicações de como montar um aparelho,
  - uma receita de culinária.
- Sequência de ações executáveis para a obtenção de uma solução para um determinado tipo de problema.

Algoritmos, Estrutura de Dados e Programas

- Segundo Dijkstra, um algoritmo corresponde a uma descrição de um padrão de comportamento, expresso em termos de um conjunto finito de ações.
  - Executando a operação a + b percebemos um padrão de comportamento, mesmo que a operação seja realizada para valores diferentes de a e b.

Programas

- Programar é basicamente estruturar dados e construir algoritmos.
- Programas são formulações concretas de algoritmos abstratos, baseados em representações e estruturas específicas de dados.
- Programas representam uma classe especial de algoritmos capazes de serem seguidos por computadores.

Programas

- Um computador só é capaz de seguir programas em linguagem de máquina (sequência de instruções obscuras e desconfortáveis).
- É necessário construir linguagens mais adequadas, que facilitem a tarefa de programar um computador.
- Uma linguagem de programação é uma técnica de notação para programar, com a intenção de servir de veículo tanto para a expressão do raciocínio algorítmico quanto para a execução automática de um algoritmo por um computador.

Estrutura de dados

- Estruturas de dados e algoritmos estão intimamente ligados:
  - não se pode estudar estruturas de dados sem considerar os algoritmos associados a elas,
  - assim como a escolha dos algoritmos em geral depende da representação e da estrutura dos dados.
- Para resolver um problema é necessário escolher uma abstração da realidade, em geral mediante a definição de um conjunto de dados que representa a situação real.
- A seguir, deve ser escolhida a forma de representar esses dados.

Estrutura de dados

- A importância de Estrutura de dados decorre do fato de que os dados precisam ser armazenados em um computador para serem processados e gerar informação.
- Este armazenamento pode ser tanto em disco quanto em memória RAM.
- O armazenamento em disco é necessário na maior parte das aplicações, mas o acesso aos arquivos em disco é mais lento do que o acesso em memória RAM.

Estrutura de dados

- Isto acontece pelo fato do menor tempo de resposta dos componentes mecânicos do qual são constituídos os sistemas de leitura e gravação em disco.
- Por outro lado, a velocidade de circulação de informações em memória RAM é muito maior.
- Portanto, o uso eficiente da memória RAM é uma característica determinante da grande maioria dos programas de computador.

Estrutura de dados

- Em programas envolvendo grande quantidade de informações, como, por exemplo, simulações usadas em jogos de computador ou ferramentas de auxílio à tomada de decisões, os tipos de estrutura de dados usadas pode ter grande influência no desempenho do sistema.

Tipos de Dados

- Caracteriza o conjunto de valores a que uma constante pertence, ou que podem ser assumidos por uma variável ou expressão, ou que podem ser gerados por uma função.
- Tipos simples de dados são grupos de valores indivisíveis (como os tipos básicos integer, boolean, char e real do Pascal).
  - Exemplo: uma variável do tipo boolean pode assumir o valor verdadeiro ou o valor falso, e nenhum outro valor.
- Os tipos estruturados em geral definem uma coleção de valores simples, ou um agregado de valores de tipos diferentes

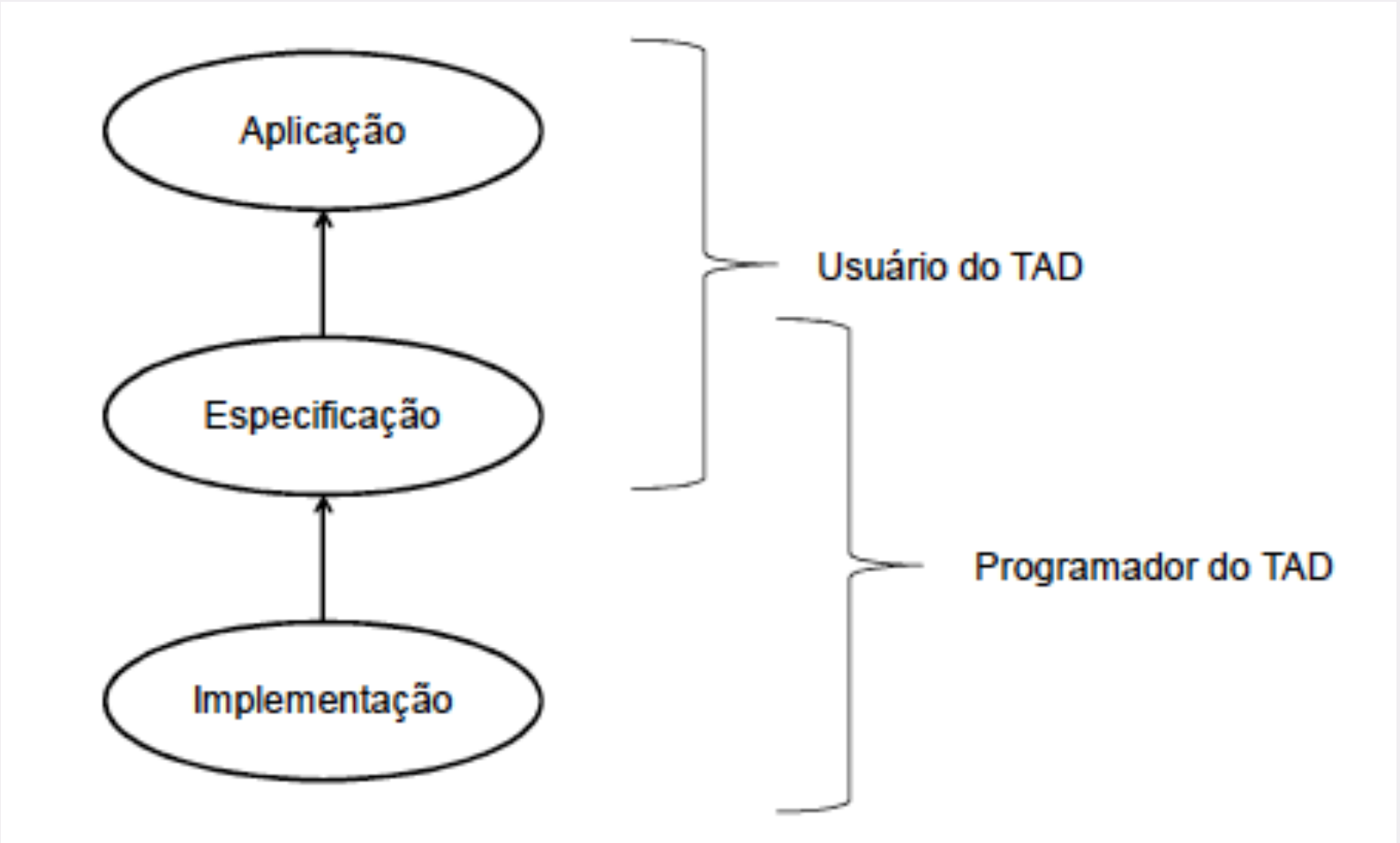
Tipos Abstratos de Dados (TAD)

- Modelo matemático, acompanhado das operações definidas sobre o modelo.
  - Exemplo: o conjunto dos inteiros acompanhado das operações de adição, subtração e multiplicação.
- TADs são utilizados como base para o projeto de algoritmos.
- A implementação do algoritmo em uma linguagem de programação exige a representação do TAD em termos dos tipos de dados e dos operadores suportados.

Tipos Abstratos de Dados (TAD)

- A representação do modelo matemático por trás do tipo abstrato de dados é realizada mediante uma estrutura de dados.
- Podemos considerar TADs como generalizações de tipos primitivos e procedimentos como generalizações de operações primitivas.
- O TAD encapsula tipos de dados. A definição do tipo e todas as operações ficam localizadas numa seção do programa.

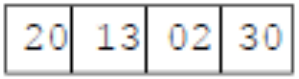
Tipos Abstratos de Dados (TAD)



Exemplo: Lista de números inteiros

- Operações
  - Faça a lista vazia;
  - Obtenha o primeiro elemento da lista; se a lista estiver vazia, então retorne nulo;
  - insira um elemento na lista.

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 de programação orientadas a objetos (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 variáveis, é possível criar uma única variável contendo 3 campos
  - Em C, usa-se a construção struct para representar esse tipo de dado

Estruturas (Structs) em C

- Sintaxe:

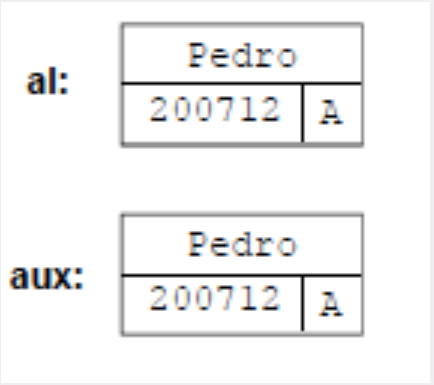
```
struct nome {
    [tipo nome_da_variável] ;
    ...
} [variável] ;
```

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

Estruturas (Structs) em C

- Exemplo:

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

using namespace std;

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

int main(void) {
  struct Aluno al, aux;
  al.nome = "pedro";
  al.matricula = 200712;
  al.conceito = 'a';
  aux = al;
  cout << al.nome << " " << al.matricula << " " << al.conceito <<
  endl;
  cout << aux.nome << " " << aux.matricula << " " << aux.conceito;
  return 0;
}
```



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

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

```
1  typedef struct {
2    int numero;
3    double saldo;
4  } contaBancaria;
5
6  contaBancaria inicializa (int, double);
7  void deposito (contaBancaria *, double);
8  void saque (contaBancaria *, double);
9  void imprime (contaBancaria);
```

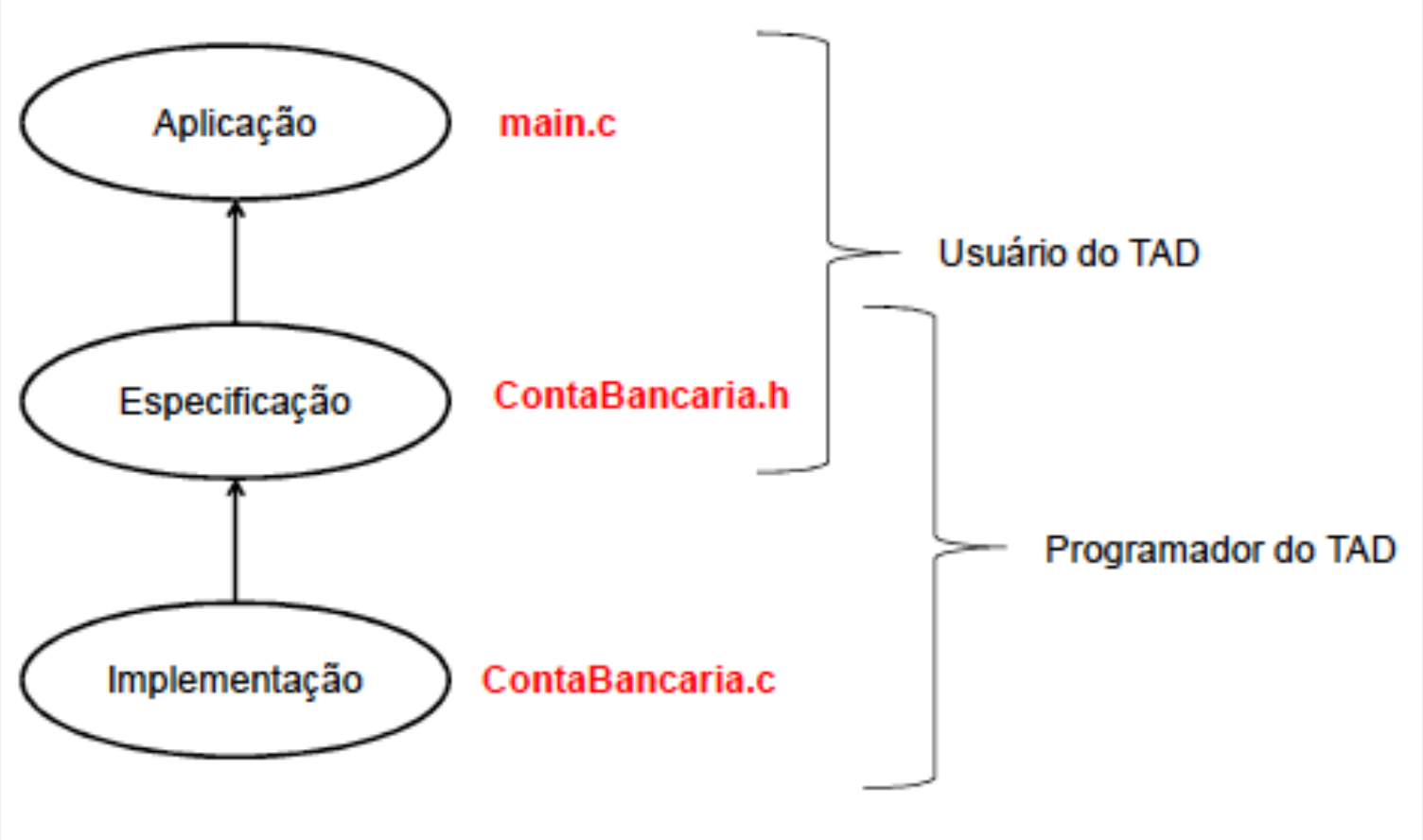
contaBancaria.cpp

```cpp
#include <iostream>
#include "ContaBancaria.h"

using namespace std;

contaBancaria inicializa (int num, double sal){
  contaBancaria conta;
  conta.numero = num;
  conta.saldo = sal;
  return conta;
}
void deposito (contaBancaria *conta, double valor){
  conta->saldo += valor;
}
void saque (contaBancaria *conta, double valor){
  conta->saldo -= valor;
}
void imprime (contaBancaria conta){
  cout << "numero " << conta.numero << " saldo R$ = "
  << conta.saldo << "\n";
}
```

main.cpp

```cpp
#include <iostream>
#include <string>
#include "ContaBancaria.h"

using namespace std;

int main() {
    contaBancaria conta;
    conta = inicializa(11, 100);
    cout << "antes da movimentação " << "\n";
    imprime(conta);

    deposito(&conta, 50);

    imprime(conta);

    saque(&conta, 100);

    cout << "depois da movimentação " << "\n";
    imprime(conta);
}
```

TADs em C


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

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

NumeroComplexo.h

```
1   typedef struct {
2     int real;
3     int img;
4   } NumeroComplexo;
5
6   NumeroComplexo inicializa(int, int);
7   void imprime (NumeroComplexo);
8   void copia (NumeroComplexo*, NumeroComplexo);
9   NumeroComplexo soma(NumeroComplexo, NumeroComplexo);
10  int ehReal(NumeroComplexo);
```

main.cpp

```cpp
#include <iostream>
#include "NumeroComplexo.h"

using namespace std;

int main() {
  NumeroComplexo a, b, c, d;
  a = inicializa(2,5);
  cout << "a = ";
  imprime(a);
  b = inicializa(1,2);
  cout << "b = ";
  imprime(b);
  c = soma(a,b);
  cout << "c = ";
  imprime(c);
  d = inicializa(5,0);
  cout << "d = ";
  imprime(d);
  if(ehReal == 0)
    copia(&d, a);
  cout << "d = ";
  imprime(d);

}
```