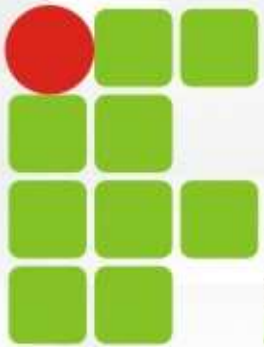
TIPOS ABSTRATOS DE DADOS

Profa.: Mirlem R. R. Pereira



INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
AMAZONAS

www.ifam.edu.br

Introdução

- Tipos de dados, estruturas de dados e tipo abstrato de dados.
  - Termos parecidos, mas com significados diferentes

Introdução

- Tipos de dados
  - Define a forma como um dado deve ser armazenado ou recuperado, bem como os possíveis valores que ele pode assumir e as operações que podem ser efetuadas sobre os mesmos.
  - Exemplo:
    - inteiro - permite valores inteiros e operações de adição, multiplicação, subtração e divisão;

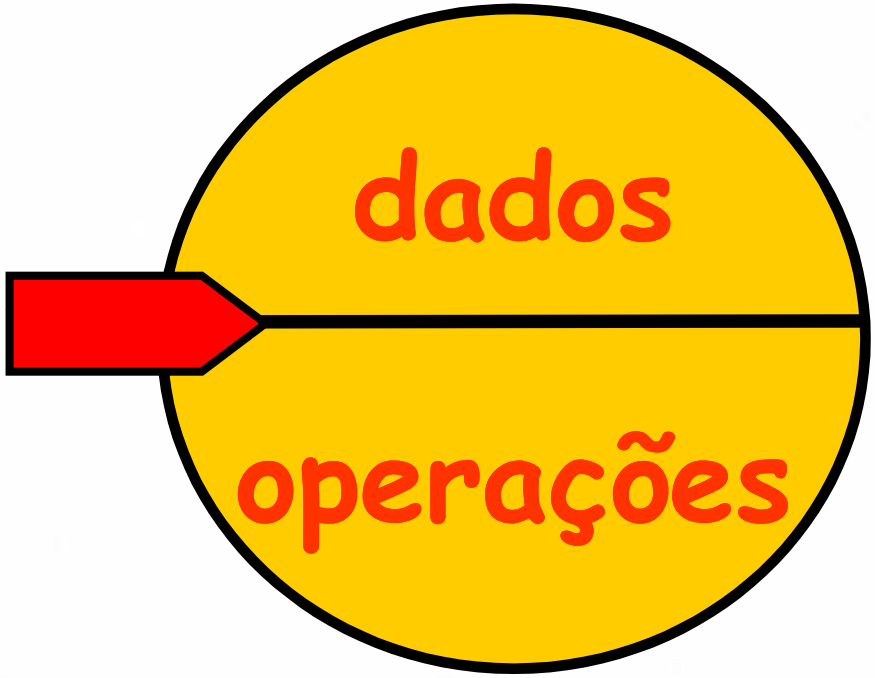
Obs. Esses tipos estão intrinsecamente relacionados com o hardware.

Introdução

- Estrutura de dados
  - Tipos estruturados são estruturas de dados já pré-definidas na linguagem de programação
  - O programador pode definir outras estruturas de dados para armazenar as informações que seu programa precisa manipular
  - Vetores, registros, listas encadeadas, pilhas, filas, árvores, grafos, são exemplos de estruturas de dados típicas utilizadas para armazenar informação em memória principal

INSTITUTO FEDERAL AMAZONAS

Tipos Abstratos de Dados - TADs



Tipos Abstratos de Dados - TADs

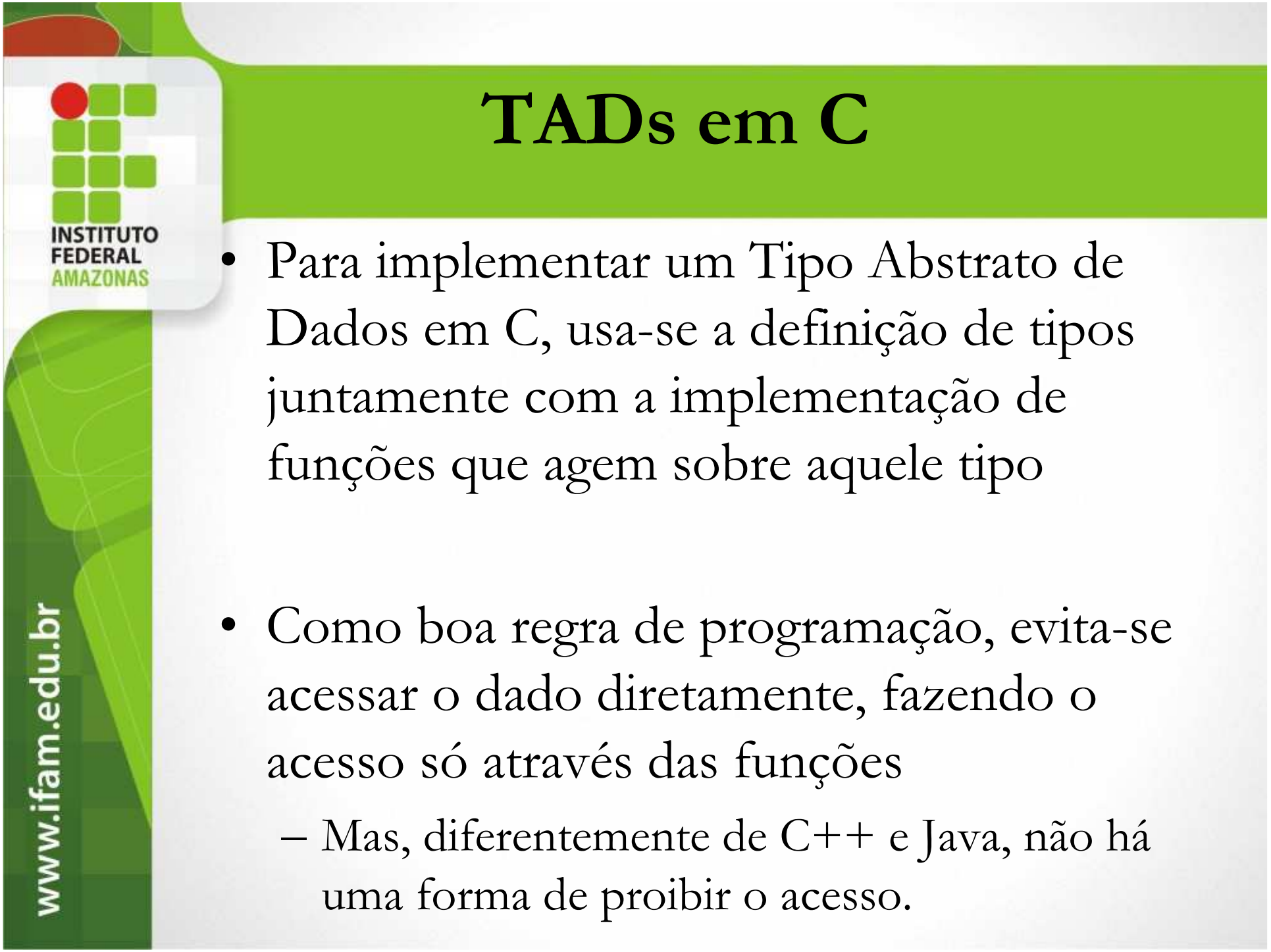
- Agrupa a estrutura de dados juntamente com as operações que podem ser feitas sobre esses dados
- O TAD encapsula a estrutura de dados. Os usuários do TAD só tem acesso a algumas operações disponibilizadas sobre esses dados
- Usuário do TAD x Programador do TAD
  - Usuário só "enxerga" a interface,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

Implementação de TADs

- Em linguagens orientadas por objeto (C++, Java) a implementação é feita através de classes
- Em linguagens estruturadas (por exemplo C) a implementação é feita pela definição de tipos juntamente com a implementação de funções



TADs em C

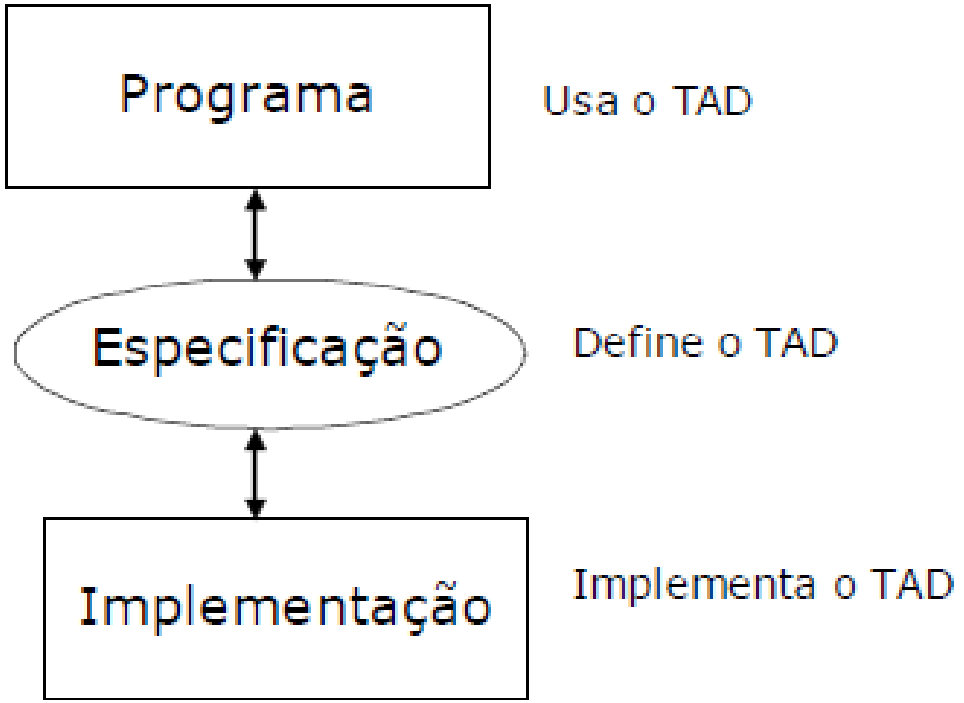
- Para implementar um Tipo Abstrato de Dados em C, usa-se a definição de tipos juntamente com a implementação de funções que agem sobre aquele tipo

- 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 pelo menos 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

Visão em Camadas



Exemplo 01

- Implemente um TAD Cilindro, com os dados raio e altura onde os clientes podem fazer as seguintes operações:
  - Calcular Área
  - Calcular Volume

Geometria.h

```
#include <stdio.h>
#include <stdlib.h>
#define PI 3.14159

float volume_cilindro(float raio, float altura);

float area_cilindro(float raio, float altura);
```

Geometria.c

```
#include <math.h>
#include "geometria.h"

float volume_cilindro(float raio, float altura){
  float volume = PI * pow(raio,2) * altura;
  return volume;
}
float area_cilindro(float raio, float altura){
  float area = 2 * PI * raio * (altura + raio);
  return area;
}
```

Principal.c

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

int main (void){
  float raio, altura, volume, area;

  printf("Entre com o valor do raio e da altura:
");
  scanf("%f %f", &raio, &altura);
  volume = volume_cilindro(raio, altura);
  area = area_cilindro(raio, altura);
  printf("Volume do cilindro: %f \n", volume);
  printf("Area do cilindro: %f \n", area);
  getch();
return 0;
}
```

Exemplo 02

- 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

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 (void)
{
    ContaBancaria conta1;
    Inicializa(conta1, 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);

    system("PAUSE");
    return(0);
}
```

EXERCÍCIO

1. Desenvolva um TAD para um CUBO. Inclua as funções de inicializações necessárias e as operações que retornem a sua área e o seu volume.

2. Defina o TAD, FRAÇÃO. Utilize dois números inteiros, um representando o numerador outro o denominador. Crie as operações: criar a fração, acessar numerador e acessar o denominador.

3. Escreva uma especificação de tipos abstratos de dados (TAD) para os números complexos, *a* + *bi*, usando estruturas com partes reais e complexas. Escreva operações para somar e multiplicar tais números.

4. Defina TDA, VETOR. Crie operações: soma dos elementos, maior elemento do vetor e a média dos elementos.

5. Defina TDA, Matriz. Crie operações básicas, como soma de duas matrizes, produto de duas matrizes e cálculo da matriz inversa.