UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

Câmpus Campo Mourão

Ciência da Computação

Algoritmos e Estrutura de Dados 1

Tipo Abstrato de Dados

Prof. Rafael Liberato

liberato@utfpr.edu.br

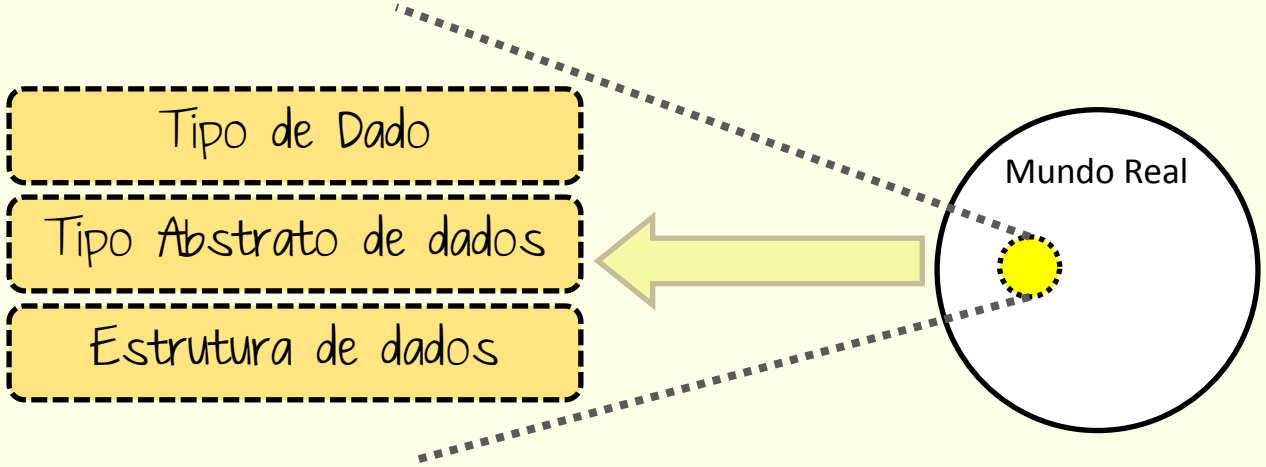
Roteiro

- Representação dos dados
  - Tipo de Dados
  - TAD - Tipo Abstrato de Dados
  - Estrutura de Dados
  - Exemplos

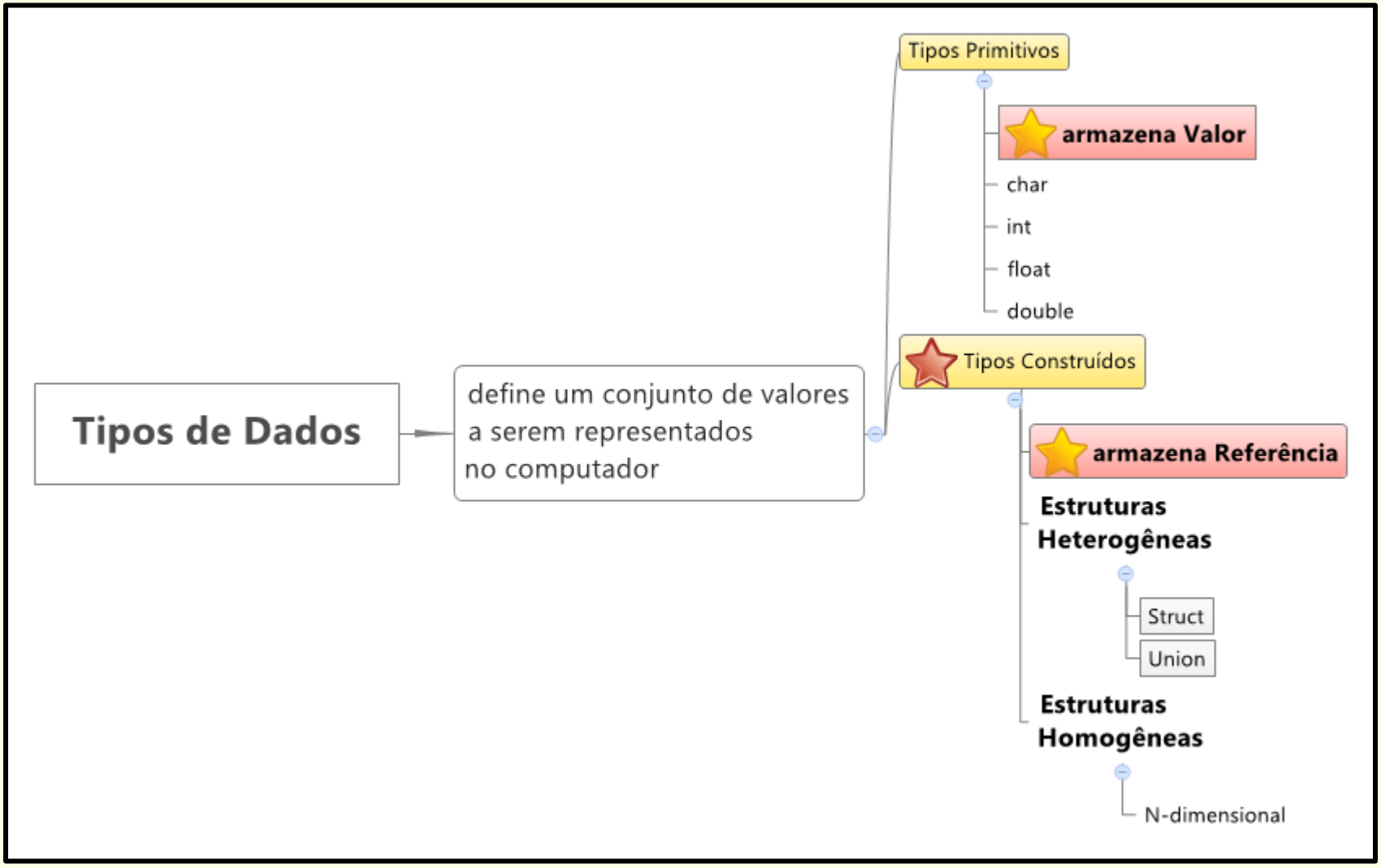
Representação dos dados

Representação dos dados

- **Representando informações do mundo real em nossos programas**
  - Tipo de dados
  - TAD - Tipo Abstrato de Dados
  - Estrutura de Dados

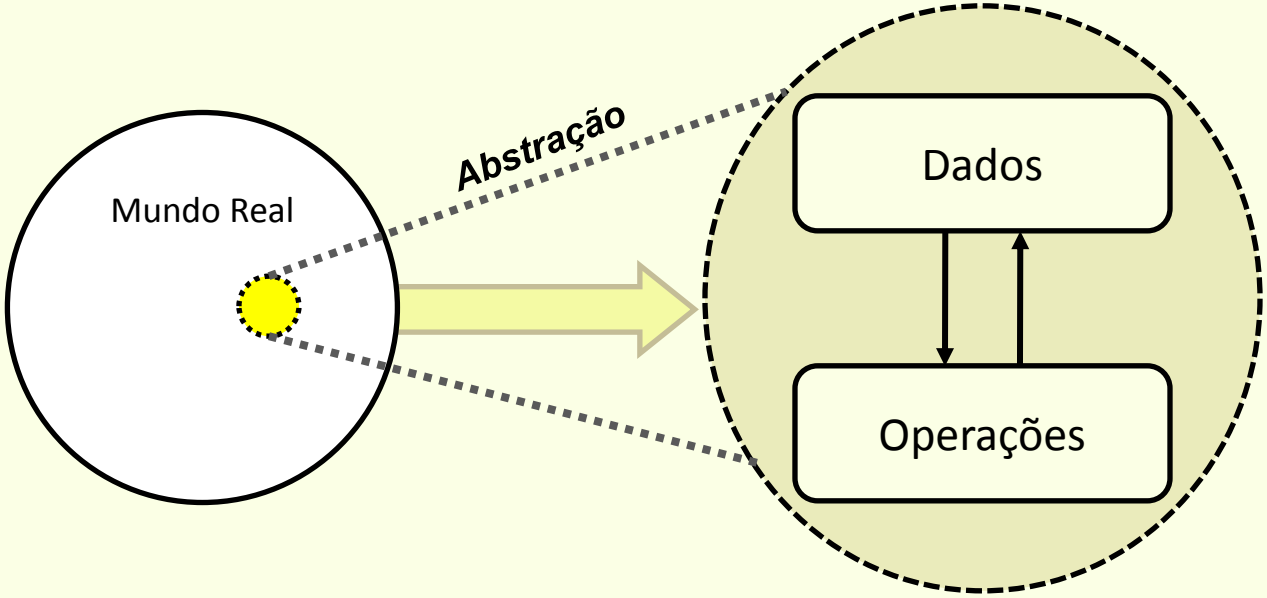


Tipo de Dados



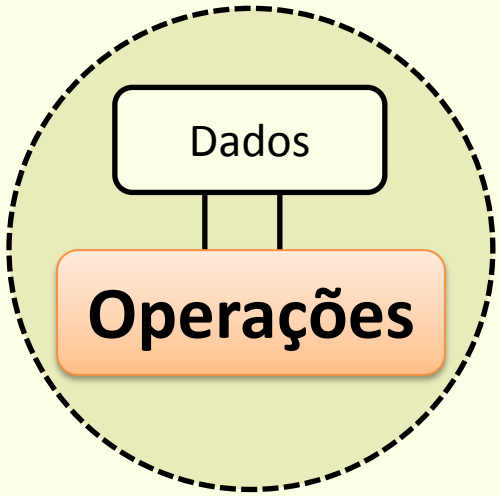
Tipo Abstrato de Dados

- A palavra ABSTRATO do TAD representa o processo de abstração de uma entidade que será modelada computacionalmente.
  - Concentrando-se nas informações relevantes para o problema.
  - Omitindo detalhes que não interessam.



Tipo Abstrato de Dados

- Um TAD consiste de um novo tipo de dado, juntamente com as **operações** que manipulam este tipo de dado

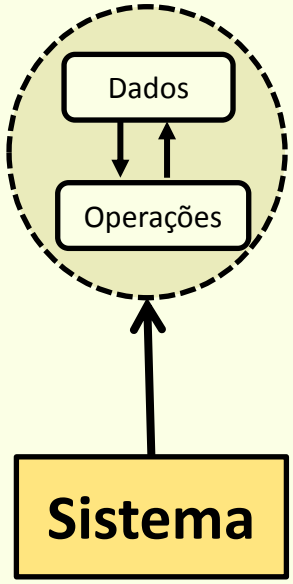


O QUE a operação realizará

COMO implementar a operação

Tipo Abstrato de Dados

- A característica essencial de um TAD é a **separação entre conceito e implementação**.

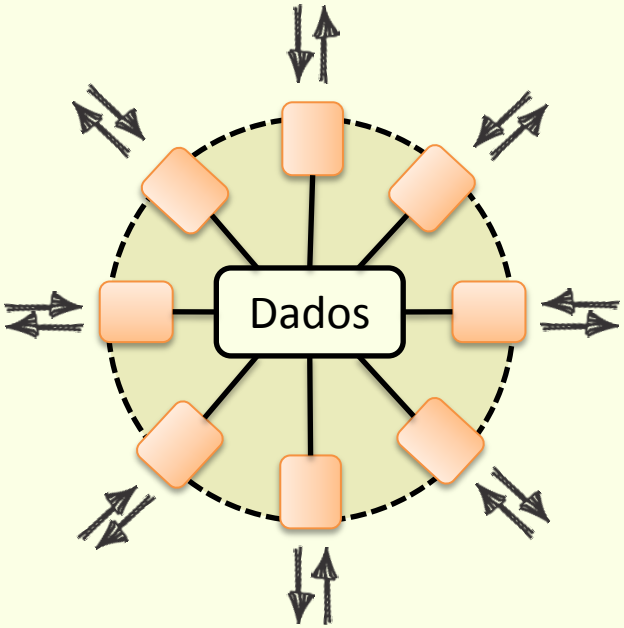


⇨ Os termos "ocultamento de informação" e "encapsulamento" são utilizados para descrever esta habilidade

⇨ Serão fornecidas a descrição dos valores e o conjunto de operações do TAD, mas a implementação fica invisível e inacessível.

Tipo Abstrato de Dados

- **Um TAD deve ser independente**

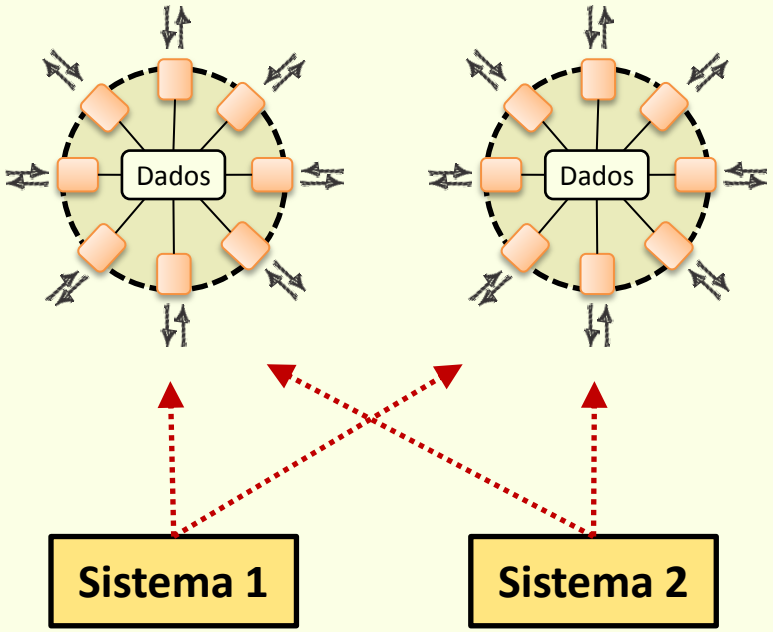


Operação (função)

- O conjunto de operações definem a interface de comunicação entre o TAD e o mundo externo
- O sistema somente se comunica com o TAD por meio de sua interface (operações)
- Mudanças na implementação do TAD não altera o programa, desde que a interface não mude.

Tipo Abstrato de Dados

- Normalmente o TAD é compilado em uma Unidade separada
- Separar a aplicação (PROGRAMAS) dos dados(ESTRUTURA DE DADOS)



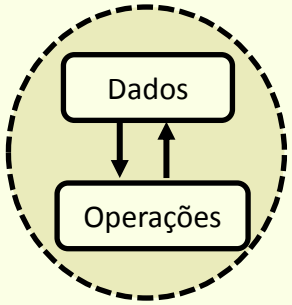
Tipo Abstrato de Dados

- Considerações ao se construir um TAD
  - Especificar o conjunto de dados e as operações sobre eles (DADOS + OPERAÇÕES)
  - Perspectiva sobre O QUE se deseja fazer
  - Não se define COMO fazer (implementação)
  - Separa conceito da implementação
  - O código deve ser fracamente acoplado

Estrutura de Dados

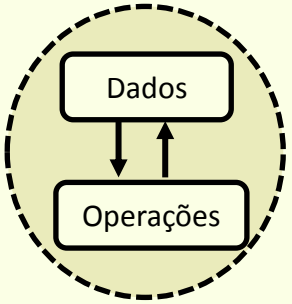
- Podemos considerar que é a materialização de um TAD

- É neste momento em que nos preocupamos com
  - Detalhes de implementação;
  - Eficiência dos algoritmos
  - Organização dos dados



Estrutura de Dados

- **Para definir um TAD precisamos saber:**
  - Quais informações serão armazenadas?
  - Quais operações serão disponibilizadas?



- **Após a definição do TAD, vamos implementar a Estrutura de Dados**
  - Definimos a forma de organização dos dados: Array, lista dinâmica, arquivo no disco, banco de dados, etc.
  - Escrevemos as operações para manipular os dados

Estrutura de Dados

- Depois de implementada a Estrutura de Dados, o conjunto de operações define a INTERFACE de utilização
  - É o que fica "visível".

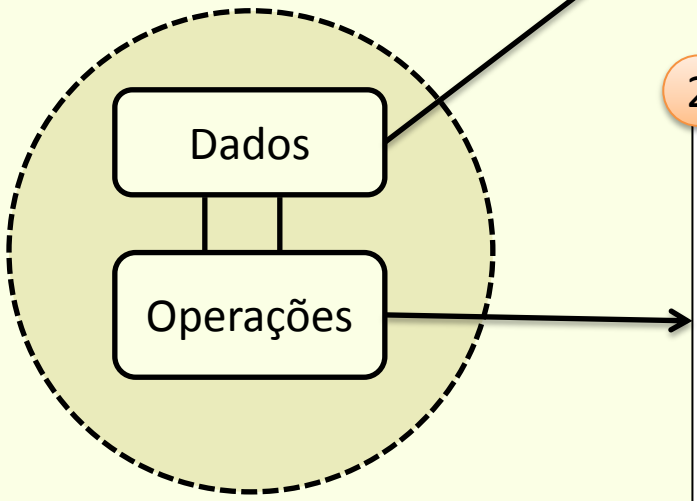
- Quando programamos pensando na INTERFACE
  - Troca-se a implementação, mas não muda a interface
  - Podemos trocar o carro movido a gasolina para um carro movido a álcool sem reaprender a dirigir.

Exemplos

Exemplo

Vamos especificar o Tipo Abstrato de Dados chamado Date

1. Quais dados serão manipulados?
2. Quais operações serão disponibilizadas para manipular os dados? O que precisamos?



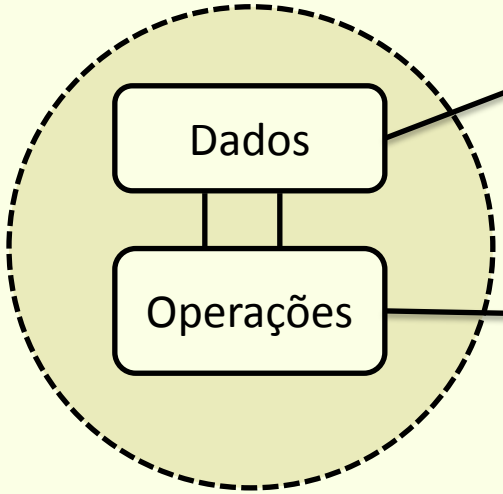
1. Uma data **01/12/2010**

2.
- Criar uma data
- Recuperar a data no formato brasileiro
- Recuperar a data no formato americano
- Alterar a data
- Recuperar somente o dia da data
- Recuperar somente o mês da data
- Recuperar o mês da data por extenso
- Recuperar somente o ano da data
- Validar uma data

Exemplo

Vamos especificar o Tipo Abstrato de Dados chamado Date

1. Quais dados serão manipulados?
2. Quais operações serão disponibilizadas para manipular os dados? O que precisamos?



1

```
typedef struct{
    int d;
    int m;
    int y;
}Date;
```

2

- Date* createDate(int d, int m, int y);
- int getDate(Date *date, char* str);
- int getAmericanDate(Date *date, char* str);
- int setDate(Date *date, int d, int m, int y);
- int getDay(Date *date);
- int getMonth(Date *date);
- int getWritingMonth(Date *date);
- int getYear(Date *date);
- int checkDate(Date *date);

Exemplo

```
typedef struct{
        int d;
        int m;
        int y;
}Date;


Date* createDate(int d, int m, int y);          //Criar uma data
int getDate(Date *date, char* str);             //Recuperar a data no formato brasileiro
int getAmericanDate(Date *date, char* str);     //Recuperar a data no formato americano
int setDate(Date *date, int d, int m, int y);   //Alterar a data
int getDay(Date *date);                         //Recuperar somente o dia da data
int getMonth(Date *date);                       //Recuperar somente o mês da data
int getWritingMonth(Date *date);                //Recuperar o mês da data por extenso
int getYear(Date *date);                        //Recuperar somente o ano da data
int checkDate(Date *date);                      //Validar uma data
```

Referências