UTFPR

Ciência da Computação

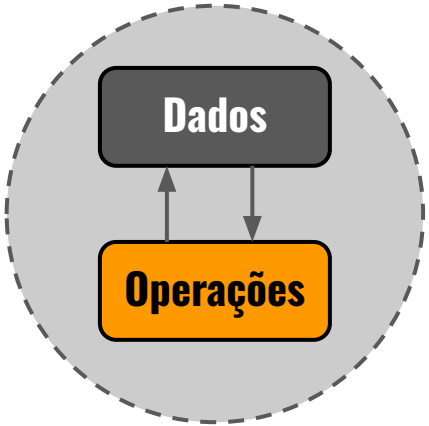
Algoritmos e Estrutura de Dados 1

TAD - Tipo Abstrato de Dados

Rafael Liberato
liberato@utfpr.edu.br

Introdução

- Um TAD é uma forma de definir um novo tipo de dado juntamente com as operações que manipulam esse novo tipo.



Vantagens

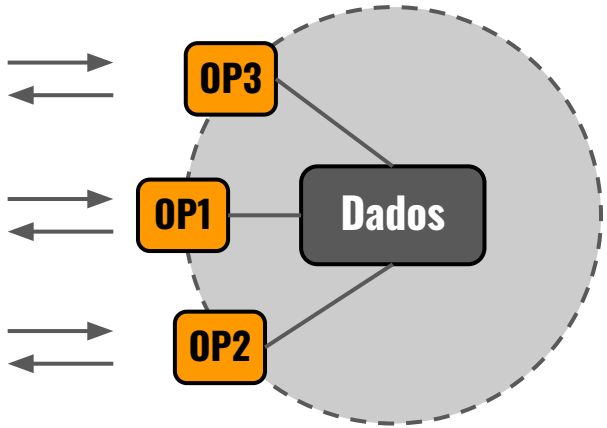
Reutilização. Quanto mais independente (acoplamento fraco) o TAD for, mais reutilizável ele será. Essa característica é importante para que possamos reutilizar esse tipo sempre que precisarmos, tal como uma biblioteca.

Separação entre conceito e implementação. As aplicações que utilizarão o TAD somente precisarão saber quais informações o tipo armazena e quais operações estão disponíveis. Os detalhes de implementação não interessam

Introdução

As operações representam a interface de comunicação entre o mundo externo e os dados.

Se preservarmos a interface (comportamento das funções), podemos alterar os dados sem impactar o mundo externo.



estudo de caso

Estudo de caso

Para guiar o nosso aprendizado, vamos utilizar um estudo de caso descrevendo o desenvolvimento de um TAD em 3 etapas

1. Definição
2. Utilização
3. Desenvolvimento

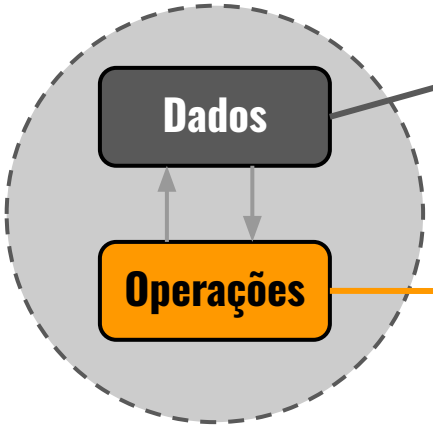
Como estudo de caso, vamos especificar e desenvolver um TAD para representar uma **data**.

1. Definição

Nesta etapa, a pergunta que norteará nosso trabalho é:

O QUE eu preciso?

*Não vamos nos preocupar em **como** vamos implementar, mas somente **no que** precisamos*



O QUE eu preciso?

dia, mês e ano

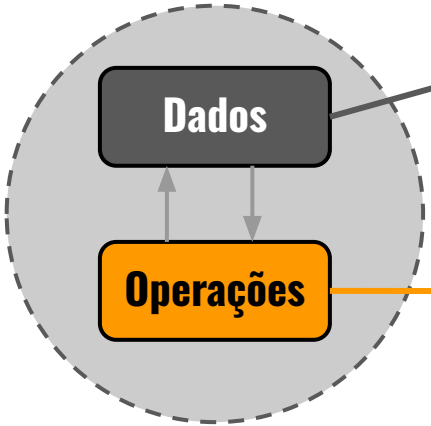
Op1: Criar uma data
Op2: Destruir uma data
Op3: Duplicar uma data
Op4: Alterar uma data
Op5: Recuperar o dia da data
Op6: Recuperar o mês (número) da data
Op7: Recuperar o ano da data
Op8: Recuperar a data no formato "dia/mês/ano"
Op9: Recuperar o mês da data por extenso

1. Definição

Nesta etapa, a pergunta que norteará nosso trabalho é:

O QUE eu preciso?

*Não vamos nos preocupar em **como** vamos implementar, mas somente **no que** precisamos*



O QUE eu preciso?

typedef struct data **Data**;

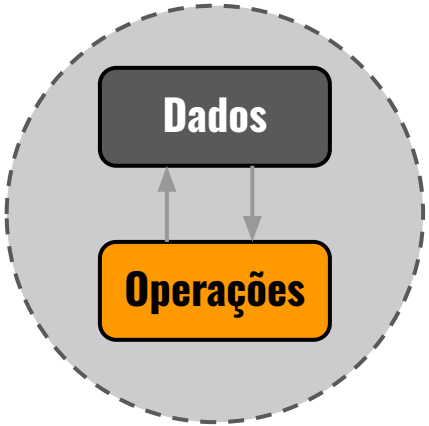
Op1: Data* data_criar(int dia, int mes, int ano);
Op2: void data_desalocar(Data** d);
Op3: Data* data_clone(Data* d);
Op4: bool data_alterar(Data* d, int dia, int mes, int ano);
Op5: int data_dia(Data* d);
Op6: int data_mes(Data* d);
Op7: int data_ano(Data* d);
Op8: bool data_string(Data* d, char* string);
Op9: bool data_mesExtenso(Data* d, char* str);

1. Definição

Nesta etapa, a pergunta que norteará nosso trabalho é:

O QUE eu preciso?

*Não vamos nos preocupar em **como** vamos implementar, mas somente **no que** precisamos*



O QUE eu preciso?

```
/*** DADOS ****/
typedef struct data Data;

/*** OPERAÇÕES ****/
Data* data_criar(int dia, int mes, int ano);
void data_desalocar(Data** d);
Data* data_clone(Data* d);
bool data_alterar(Data* d, int dia, int mes, int ano);
int data_dia(Data* d);
int data_mes(Data* d);
int data_ano(Data* d);
bool data_string(Data* d, char* str);
bool data_mesExtenso(Data* d, char* str);
```

2. Utilização

A utilização das funções sem conhecer sua implementação também é uma ótima forma de te auxiliar na implementação. Se for capaz de usar a função, significa que você compreendeu o que ela faz e isso te dará clareza na implementação.

```
/*** DADOS ****/
typedef struct data Data;

/*** OPERAÇÕES ****/
Data* data_criar(int dia, int mes, int ano);
void data_desalocar(Data** d);
Data* data_clone(Data* d);
bool data_alterar(Data* d, int dia, int mes, int ano);
int data_dia(Data* d);
int data_mes(Data* d);
int data_ano(Data* d);
bool data_string(Data* d, char* str);
bool data_mesExtenso(Data* d, char* str);
```

```
include “tad_data.h”

int main(){
   Data* d1= data_criar(01, 10, 2021);
   char dataStr[15];
   data_string(d1, dataStr);
   printf(“%s”, dataStr);
}
```

→ resultado esperado: 01/10/2021

Faça isso com todos os protótipos criados

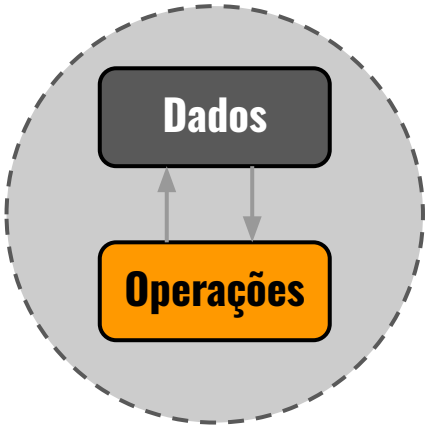
3. Desenvolvimento

A implementação de um tipo abstrato de dados é o momento para responder todas as perguntas da perspectiva **COMO**.

- **Como** os dados devem ser modelados e organizados na memória?
- **Como** os dados devem ser manipulados por suas operações?
- **Como** as operações transformarão os parâmetros de entrada nos resultados esperados?

3. Desenvolvimento

Como os dados serão organizados na memória?




```
typedef struct data {



} Data;
```

```
int dia;
int mes;
int ano;
```

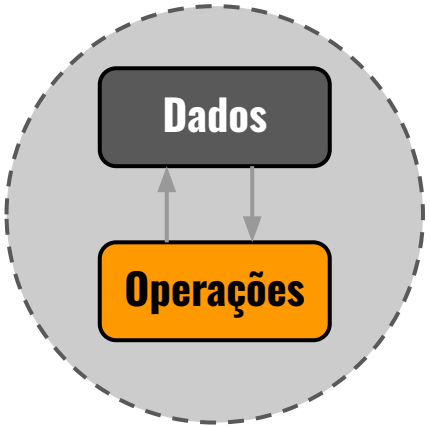
```
int dados[3]
```

```
char dia[3];
char mes[3];
char ano[5];
```

```
char dados[12]
```

3. Desenvolvimento

Como as funções serão implementadas?



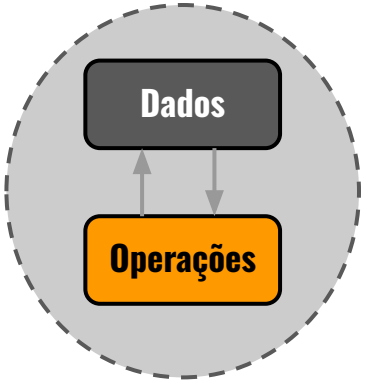

```
typedef struct data {
    int dia;
    int mes;
    int ano;
} Data;
```

```
Data* data_criar(int dia, int mes, int ano);
void data_desalocar(Data** d);
Data* data_clone(Data* d);
bool data_alterar(Data* d, int dia, int mes, int ano);
int data_dia(Data* d);
int data_mes(Data* d);
int data_ano(Data* d);
bool data_string(Data* d, char* str);
bool data_mesExtenso(Data* d, char* str);
```

3. Desenvolvimento

Como as funções serão implementadas?

```
typedef struct data {
    int dia;
    int mes;
    int ano;
} Data;
```



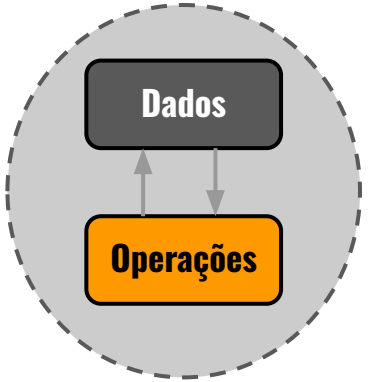

```
Data* data_criar(int dia, int mes, int ano){
    Data* d = (Data*) calloc(1,sizeof(Data));
    d->dia = dia;
    d->mes = mes;
    d->ano = ano;
    return d;
}
```

```
void data_desalocar(Data** enderecoData){
    free(*enderecoData);
    *enderecoData = NULL;
}
```

3. Desenvolvimento

Como as funções serão implementadas?

```
typedef struct data {
    int dia;
    int mes;
    int ano;
} Data;
```




```
Data* data_clone(Data* d){
    if(d==NULL) return NULL;

    Data* novo = data_criar(d->dia, d->mes, d->ano);
    return novo;
}
```

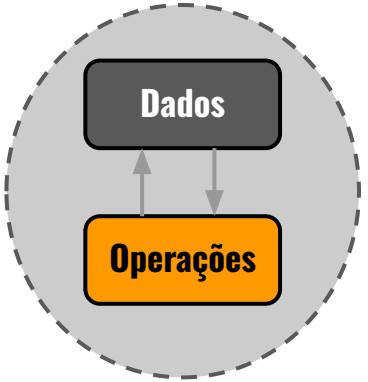
```
bool data_alterar(Data* d, int dia, int mes, int ano){
    if(d == NULL) return false;

    d->dia = dia;
    d->mes = mes;
    d->ano = ano;
    return true;
}
```

3. Desenvolvimento

Como as funções serão implementadas?

```
typedef struct data {
    int dia;
    int mes;
    int ano;
} Data;
```




```
int data_dia(Data* d){
    if(d == NULL) return -1;
    return d->dia;
}
```

```
int data_mes(Data* d){
    if(d == NULL) return -1;
    return d->mes;
}
```

```
int data_ano(Data* d){
    if(d == NULL) return -1;
    return d->ano;
}
```

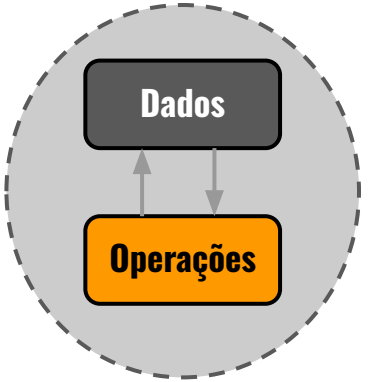
```
bool data_string(Data* d, char* str){
    if(d == NULL) return false;

    sprintf(str, "%02d/%02d/%d", d->dia, d->mes, d->ano);
    return true;
}
```

3. Desenvolvimento

Como as funções serão implementadas?

```
typedef struct data {
    int dia;
    int mes;
    int ano;
} Data;
```




```
bool data_mesExtenso(Data* d, char* str){
    if(d == NULL) return false;

    char meses[12][10] = {"janeiro","fevereiro","marco","abril","maio","junho",
    "julho","agosto","setembro","outubro","novembro","dezembro"};

    strcpy(str, meses[d->mes-1]);
    return true;
}
```


Fim