

Aula 12

Tipo Abstrato de Dados

Prof. Me. Sérgio Carlos Portari Jr
UEMG – Campus de Frutal

Representação dos Dados

-  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: $II + III = IIIII$
 - Boa para pequenos números (operação simples)
 - Representação decimal: $1278 + 321 = 1599$
 - Boa para números maiores (operação complexa)

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

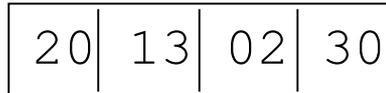
Tipos Abstratos de Dados (TADs)

- 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

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
- Como vocês não viram o conceito de orientação por objetos*, vamos utilizar os conceitos de C (**Typedef e Structs**)
- Orientação por objetos (classes, etc) vai ser vista em outras disciplinas (LP)

Estruturas (Structs) em C / C++

- Uma estrutura é uma coleção de uma ou mais variáveis, possivelmente de tipos diferentes, colocadas juntas sob um único nome para manipulação conveniente
- Por 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 / C++

```
#include <stdio.h>
#include <string.h>
```

```
struct aluno {
char nome[40];
int matricula;
char conceito;
};
```

```
int main()
{
    struct aluno al, aux;
    strcpy(al.nome, "Sergio");
    al.matricula = 10023;
    al.conceito = 'A';
    aux = al;
    printf("Aluno: %s", aux.nome);
    return 0;
}
```

al:

Sergio	
10023	A

aux:

Sergio	
10023	A

Declaração de Tipos

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

```
#include <stdio.h>
#include <string.h>

typedef struct aluno {
char nome[40];
int matricula;
char conceito;
} TipoAluno;
```

```
int main()
{
    TipoAluno al, aux;

    strcpy(al.nome, "Sergio");
    al.matricula = 10023;
    al.conceito = 'A';
    aux = al;
    printf("Aluno:
%s", aux.nome);
    return 0; }
```

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.cpp : 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

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

ContaBancaria.cpp

```
#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);
}

//continua no proximo slide
```

ContaBancaria.cpp

```
//continuacao
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);

    return(0);
}
```

Exercício

1- Especificar e implementar de forma independente o TAD ponto, para representar pontos no espaço bidimensional (\mathbb{R}^2). Este TAD deve permitir criar de um ponto e calcular a distância entre dois pontos.

O programa principal (aplicativo) deve ler as coordenadas de 2 pontos (digitadas pelo usuário) e imprimir na tela a distância entre estes pontos. Para isto, este aplicativo deverá criar os dois pontos, calcular e mostrar a distância entre eles.

A distância entre os pontos é calculada por:

$$\sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2}$$