FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

Estrutura de Dados

Elidiane Martins
elidiane@fgf.edu.br

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

Conceitos:

- Algoritmos: Sequência de ações expressas em termos de uma linguagem de programação, constituindo parte da solução de um tipo determinado de problema
- Estrutura de dados: Complementa o algoritmo na constituição da solução do problema considerado
- Programa: Programar é basicamente estruturar dados e construir algoritmos.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

Tipo de dado

- Tipo de dados: caracteriza o conjunto de valores a que uma constante/variável ou função pode assumir;
- Pode ser:
  - Elementar: Domínio de valores indivisível (int, char, float, bool…)
  - Estruturado:
    - Coleção homogênea do mesmo tipo: Matriz, vetores.
    - Estruturas heterogêneas de tipos diferentes: Structs.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

TAD

Tipo Abstrato de Dados:

- Encapsulam a representação dos dados e as operações que podem ser realizadas sobre eles
- Usuário do TAD vs. programador do TAD
  - Usuário só “enxerga” a interface, não a implementação
  - Os usuários de um TAD só têm acesso às operações disponibilizadas sobre os dados

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

TAD

Tipo Abstrato de Dados:

- Podemos modificar a implementação do TAD sem modificar o código que usa o TAD;
- E vice-versa, podemos modificar o código que usa o TAD sem modificar a implementação do TAD
- TAD pode ser reaproveitado em vários programas ou módulos

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

TAD

Tipo Abstrato de Dados:

- Em linguagens orientadas a objeto (C++, Java, Ruby, PHP, etc), implementação é feita com classes (O futuro de aguarda);
- Em linguagem estruturadas como C, implementação é feita com definições de tipos e implementação de funções;
- TAD é o conceito básico para a abordagem orientada a objetos.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

TAD

Tipo Abstrato de Dados:

- TAD, portanto, estabelece o conceito de tipo de dado divorciado da sua representação
- Definido como um modelo matemático por meio de um par (v,o) em que:
  - v é um conjunto de valores;
  - o é um conjunto de operações sobre esses valores;
  - Ex.: tipo real
  - v = R
  - o = {+, -, *, /, =, , <=, >=}
  - Ao pensar nas operações, é necessário definir suas entradas, saídas, précondição e póscondição.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

TAD

Para definir um TAD:

- O programador descreve o TAD em dois módulos separados:
- Um módulo contém a definição do TAD: representação da estrutura de dados e implementação de cada operação suportada;
- Outro módulo contém a interface de acesso: apresenta as operações possíveis
- Dessa forma, outros programadores podem, por meio da interface de acesso, usar o TAD sem conhecer os detalhes representacionais e sem acessar o módulo de definição

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

TAD

- Característica essencial de TAD é a separação entre a definição conceitual – par (v, o) – e a implementação (ED específica):
  - O programa só acessa o TAD por meio de suas operações, a ED nunca é acessada diretamente;
  - "ocultamento de informação"

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

TAD

- Programador tem acesso a uma descrição dos valores e operações admitidos pelo TAD:
- Programador não tem acesso à implementação;
  - A implementação é ”invisível” e inacessível
  - Ex. posso criar um programa que use operações de curtir e compartilhar postagens, mas não preciso saber como essas operações são realmente implementadas.
- Quais as vantagens?

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

TAD

- Reuso;
- Manutenibilidade;
- Portabilidade;
- Agilidade;
- Corretude;
- Eficiência;

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

TAD

- E na prática?????
- Em primeiro lugar é criado o TAD, com a definição do tipo e a definição das operações (O QUE ele faz);
  - Exemplo.h
- Em seguida deve ser criado o módulo, com a implementação das operações definidas.
  - Exemplo.c
- Por fim, qualquer programa cliente poderá utilizar as operações do TAD criado.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

TAD

- E na prática?????
- Vamos imaginar um TAD para representar um ponto no R^2.
  - Valores: números reais;
  - Operações:
    - cria ponto,
    - exclui ponto,
    - verifica valores,
    - troca valor de x,
    - troca valor de y, etc.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

TAD

- Operações:
  - cria ponto:
    - **Entrada**: um x e um y;
    - **Saída**: ponto criado;
    - **Pre condição**: nenhuma;
    - **Pós condição**: O ponto criado com as coordenadas x e y.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

TAD

- Operações:
  - Exclui ponto:
    - **Entrada**: Um ponto;
    - **Saída**: Ponto excluido;
    - **Pre condição**: O ponto precisa existir;
    - **Pós condição**: Ponto excluído e espaço liberado.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

TAD

- Operações:
  - Verifica valores:
    - **Entrada**: Um ponto;
    - **Saída**: Valor de X e Valor de Y;
    - **Pre condição**: O ponto precisa existir;
    - **Pós condição**: coordenadas X e Y impressas.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

TAD

- Operações:
  - Troca valor de X:
    - **Entrada**: Um ponto e um novo valor para x;
    - **Saída**: Valor de X alterado;
    - **Pre condição**: nenhuma;
    - **Pós condição**: O X com um novo valor e o valor de Y inalterado.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

TAD

- Operações:
  - Troca valor de Y:
    - **Entrada**: Um ponto e um novo valor para y;
    - **Saída**: Valor de Y alterado;
    - **Pre condição**: nenhuma;
    - **Pós condição**: O Y com um novo valor e o valor de X inalterado.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

TAD

- O TAD do ponto, arquivo **ponto.h**, é algo do tipo.

```
typedef struct ponto{
  float x;
  float y;
}Ponto;

void criaponto(Ponto *p,float x, float y);
void vervalores(Ponto *a);
void trocaX(Ponto *a, float x);
void trocaY(Ponto *a, float y);
Ponto * exclui(Ponto *a);
```

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

O módulo deste TAD, que traz a implementação da operação, o arquivo **ponto.c**, é algo do tipo:

```c
#include <stdio.h>
#include <stdlib.h>
#include "ponto.h"

void criaponto(Ponto *p,float valorX, float valorY){
  p->x=valorX;
  p->y=valorY;
}
void vervalores(Ponto *a){
  if(a==NULL) {
    printf("Ponto não existe\n");
    return;}
  printf("valor de x é %f: \n", a->x);
  printf("valor de y é %f: \n", a->y);
}
void trocaX(Ponto *a, float valorX){
  a->x=valorX;
}
void trocaY(Ponto *a, float valorY){
  a->y=valorY;
}
Ponto * exclui(Ponto *a){
  a=NULL;
return a;
}
```

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Detalhes

A operação **criaponto** recebe como parâmetros: um endereço para um tipo Ponto e dois valores do tipo ponto flutuante.

A operação deve acessar o campo x do endereço recebido e adicionar o valor do primeiro ponto flutuante;

Em seguida acessa o campo y, e insere o valor do segundo ponto flutuante passado.

```
void criaponto(Ponto *p,float valorX, float valorY){
  p->x=valorX;
  p->y=valorY;
}
```

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Detalhes

A operação **trocaX** recebe como parâmetros: um endereço para um tipo Ponto e um valor do tipo flutuante para armazenar no campo x do endereço passado.

```
void trocaX(Ponto *a, float valorX){
  a->x=valorX;
}
```

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Detalhes

A operação **trocaY** recebe como parâmetros: um endereço para um tipo Ponto e um valor do tipo flutuante para armazenar no campo y do endereço passado.

```
void trocaY(Ponto *a, float valorY){
    a->y=valorY;
}
```

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

FGF

Detalhes

A operação **exclui** recebe como parâmetro: um endereço para um tipo Ponto;

A operação adiciona NULL ao ponto e retorna o endereço do ponto.

```
Ponto * exclui(Ponto *a){
  a=NULL;
return a;
}
```

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Detalhes

A operação **vervalores** tem como principal foco mostrar os valores do ponto. Contudo, é precido que o ponto exista para que esses valores sejam exibidos. Isso é uma pré-condição para executar a operação.

A operação recebe o endereço de um tipo Ponto, e deve ser capaz de imprimir o valor armazenado nos campos x e y;

```
void vervalores(Ponto *a){
  if(a==NULL) {
    printf("Ponto não existe\n");
    return;}
  printf("valor de x é %f: \n", a->x);
  printf("valor de y é %f: \n", a->y);
}
```

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

O cliente: main.c

```c
#include <stdio.h>
#include "ponto.h"

int main(){
  Ponto p, *l=&p;

  criaponto(l,3.2,7.5);
  vervalores(l);

  trocaX(l, 1.2);
  vervalores(l);

  trocaY(l, 9.5);
  vervalores(l);

  l=exclui(l);
  vervalores(l);
}
```

```
[pcbib-02:TADLuz elidiane$ gcc -c ponto.c main.c
[pcbib-02:TADLuz elidiane$ gcc -o prog ponto.o main.o
[pcbib-02:TADLuz elidiane$ ./prog
 valor de x é 3.200000:
 valor de y é 7.500000:
 valor de x é 1.200000:
 valor de y é 7.500000:
 valor de x é 1.200000:
 valor de y é 9.500000:
 Ponto não existe
```

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Outras operações

Outras operações podem ser adicionadas no TAD. Para tanto, a operação deve ser definida no cabeçalho (.h) e implementada no módulo (.c), para que o cliente possa usá-la. Ciente disso, crie as seguintes operações:

- Uma operação que receba dois pontos e **troque os valores** deles;
- Uma operação que receba um ponto e **informe em qual quadrante** do plano cartesiano este ponto se encontra (1, 2, 3 ou 4).
- Uma operação que receba um ponto e **altere o seu quadrante**, conforme informado pelo usuário.
- Uma operação que receba 2 pontos e realize **a soma deles**;
- Uma operação que receba 2 pontos e realize a **subtração deles**;
- Uma operação que receba dois pontos e calcule a **distância** entre eles.

FACULDADE INTEGRADA
DA GRANDE FORTALEZA

Referências

SILVA, O. Q. Estrutura de Dados e Algoritmos usando C – Fundamentos e Aplicações. Rio de Janeiro: Editora Ciência Moderna ltda., 2007

OLIVEIRA, U. Programando em C Fundamentos. Volume 1. Rio de Janeiro: Editora Ciência Moderna. 2008.

TENENBAUM, A. M. Estrutura de Dados usando C. Makron books. 1991.