Algoritmos e Estruturas de Dados I
Aula 01: Introdução

Prof. Márcio Porto Basgalupp

créditos: Prof. Jurandy G. Almeida Jr.

Universidade Federal de São Paulo
Departamento de Ciência e Tecnologia

Qual a diferença entre um algoritmo e um programa?

Algoritmos e Estruturas de Dados

UNIFESP
UNIVERSIDADE FEDERAL DE SÃO PAULO
1933

- Algoritmo
  - Sequência de ações executáveis para a solução de um determinado tipo de problema
  - Exemplo: “Receita de Bolo”
  - Em geral, algoritmos trabalham sobre …
- Estruturas de Dados
  - Conjunto de dados que representa uma situação real
  - Abstração da realidade
  - Estruturas de Dados e Algoritmos estão intimamente ligados

Representação dos Dados

- Dados podem ser representados (estruturados) de diferentes maneiras
- 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)

Programas

- Um programa é uma formulação concreta de um algoritmo abstrato, baseado em representações de dados específicas
- Os programas são feitos em alguma linguagem que pode ser entendida e seguida pelo computador
  - Linguagem de máquina
  - Linguagem de alto nível (uso de compilador)

Tipos Abstratos de Dados (TADs)

- Agrupa 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 têm acesso a algumas operações disponibilizadas sobre esses dados
- Usuário do TAD x Programador do TAD
  - Usuário só “enxerga” a interface, não a implementação

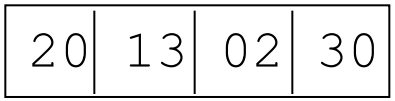
Tipos Abstratos de Dados (TADs)

- 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 Inteiros

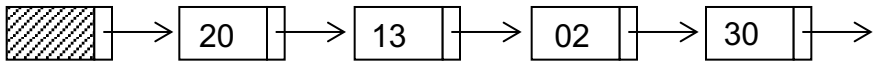
- Insere número no começo da lista

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 a 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**
  - Conceitos de C: **typedef** e **struct**

Estruturas em C

- Uma estrutura é uma coleção de uma ou mais variávies colocadas juntas sob um único nome para manipulação conveniente
  - Por exemplo, para representar um aluno são necessárias as informações nome, matrícula e 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

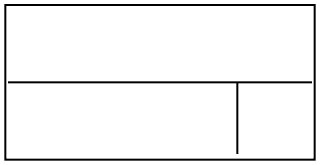
Estruturas em C

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

struct Aluno {
  char nome[20];
  int matricula;
  char conceito;
};

int main(void)
{
  struct Aluno al, aux;

  strcpy(al.nome, "Pedro");
  al.matricula = 200712;
  al.conceito = 'A';
  aux = al;
  printf("%s", aux.nome);
}
```

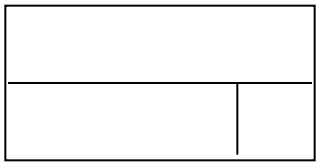
Estruturas em C

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

struct Aluno {
  char nome[20];
  int matricula;
  char conceito;
};

int main(void)
{
  struct Aluno al, aux;

  strcpy(al.nome, "Pedro");
  al.matricula = 200712;
  al.conceito = 'A';
  aux = al;
  printf("%s", aux.nome);
}
```

al:



aux:



Estruturas em C

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

struct Aluno {
  char nome[20];
  int matricula;
  char conceito;
};

int main(void)
{
  struct Aluno al, aux;

  strcpy(al.nome, "Pedro");
  al.matricula = 200712;
  al.conceito = 'A';
  aux = al;
  printf("%s", aux.nome);
}
```

al:

Pedro	
200712	A

aux:

Estruturas em C

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

struct Aluno {
  char nome[20];
  int matricula;
  char conceito;
};

int main(void)
{
  struct Aluno al, aux;

  strcpy(al.nome, "Pedro");
  al.matricula = 200712;
  al.conceito = 'A';
  aux = al;
  printf("%s", aux.nome);
}
```

al:

Pedro	
200712	A

aux:

Pedro	
200712	A

Declaração de Tipos

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

```
typedef struct {
  string nome;
  int matricula;
  char conceito;
} TAluno;

typedef int[10] TVetor;
```

```
int main(void)
{
  TAluno al;
  TVetor v;

  ...
}
```

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 ele
- Como boa regra de programação, evita-se acessar o dado diretamente, fazendo o acesso **somente 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.c: com a implementação
- O programa ou outros TADs que utilizam o seu TAD deve dar um #include do arquivo .h

Exemplo

- Implemente um TAD Conta Bancária 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;

// cabeçalho das funções
void Inicializa (ContaBancaria*, int, double);
void Deposito   (ContaBancaria*, double);
void Saque      (ContaBancaria*, double);
void Imprime    (ContaBancaria);
```

ContaBancaria.c

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

void Inicializa(ContaBancaria* pconta, int numero, double saldo) {
     pconta->numero = numero;
     pconta->saldo = saldo;
}

void Deposito(ContaBancaria* pconta, double valor) {
     pconta->saldo += valor;
}

void Saque(ContaBancaria* pconta, double valor) {
     pconta->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

- Implemente um TAD Número Complexo com os campos **real** e **imaginário** onde os usuários podem fazer as seguintes operações:
  - Atribuir valores para os campos
  - Imprimir o número na forma “R + C i"
  - Copiar o valor de um número para outro
  - Somar dois números complexos
  - Testar se um número é real

- Faça um pequeno programa para testar o seu TAD

NumeroComplexo.h

```
// definição do tipo
typedef struct {
   double real;
   double imag;
} NumeroComplexo;

// cabeçalho das funções
NumeroComplexo Atribui (double, double);
NumeroComplexo Soma    (NumeroComplexo, NumeroComplexo);
NumeroComplexo Copia   (NumeroComplexo);
void           Imprime (NumeroComplexo);
int            EhReal  (NumeroComplexo);
```

NumeroComplexo.c

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

NumeroComplexo Atribui(double real, double imag) {
     NumeroComplexo c;
     c.real = real;
     c.imag = imag;
     return c;
}

NumeroComplexo Soma(NumeroComplexo a, NumeroComplexo b) {
     NumeroComplexo c;
     c.real = a.real + b.real;
     c.imag = a.imag + b.imag;
     return c;
}

...
```

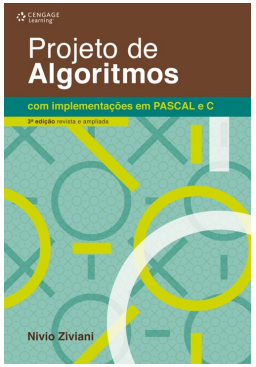
NumeroComplexo.c

```
NumeroComplexo Copia(NumeroComplexo c) {
      return c;
}

void Imprime(NumeroComplexo c) {
     printf("%f + %f i\n", c.real, c.imag);
}

int EhReal(NumeroComplexo c) {
     return (c.imag == 0.0);
}
```

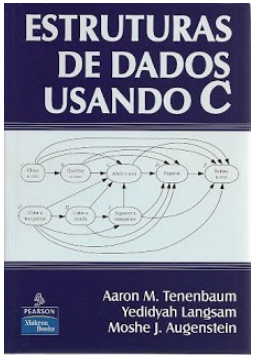
main.c

```
#include<stdio.h>
#include<stdlib.h>
#include "NumeroComplexo.h"

int main(void)
{
    NumeroComplexo a, b, c, d;
    a = Atribui(50.0,  70.0);
    b = Atribui(50.0, -70.0);
    c = Soma(a, b);
    if (EhReal(c))  d = Copia(a);
    else            d = Copia(b);
    Imprime(d);

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

Leitura Recomendada



ZIVIANI, N. **Projeto de Algoritmos com Implementações em Pascal e C**. 3ª Edição. Cengage Learning, 2010. **Seções 1.1 e 1.2**



TENENBAUM, A. M.; LANGSAM, Y.; AUGENSTEIN, M. J. **Estruturas de Dados usando C**. Pearson Makron Books, 2008. **Capítulo 1**