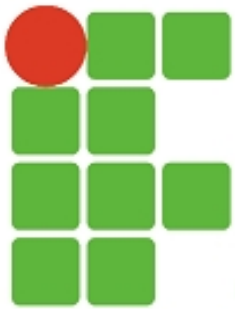


INSTITUTO FEDERAL DE
EDUCAÇÃO, CIÊNCIA E TECNOLOGIA
PARAÍBA
Campus Campina Grande

Curso Engenharia da Computação
Programação e Estruturas de Dados

3 Tipos Abstratos de Dados e Listas Lineares

Prof. César Rocha
cesarocha@ifpb.edu.br

Objetivos

- *Explorar os conceitos fundamentais acerca da definição de* ***tipos abstratos*** *usando C*
  - *Desafios em modelar entidades do mundo real, dados e operações em um TAD, como materializar estes conceitos usando a linguagem C e bibliotecas*
- *Overview sobre* ***listas lineares***
  - *Suas propriedades e operações fundamentais, listas* ***seqüenciais*** *e* ***encadeadas***
- *Este módulo é breve, mas constitui um importante instrumento para os próximos módulos do curso*

Parte I : Tipos Abstratos de Dados

Motivação

- *Em linguagens de programação, as variáveis e constantes são classificadas de acordo com um* ***tipo*** *de dados* ***nativo*** *da linguagem em uso*
  - *Ex: Todos os tipos primitivos (float, int, double, char,..)*
  - *Estes tipos são utilizados para* ***representar*** *números inteiros, caracteres, etc.*
  - *Uma vez que o tipo é estabelecido, também podemos saber que* ***operações*** *são suportadas por este tipo*
- *Entretanto, para resolvermos problemas complexos no mundo real com programação, faz-se necessário modelar outros tipos de dados...*

Motivação

- *Estudo de caso breve e simples:*
  - *Implementando um **sistema de caixa** numa linguagem procedural*
- *Você pergunta ao seu contratante: "Mas...o que é um sistema de caixa?"*
  - *É um sistema que cadastra **produtos** e **clientes** e relaciona-os numa **venda***
  - *Operações básicas deste tipo de sistema (resumidamente...)*
    - *Incluir produto numa venda, cadastrar um cliente novo, calcular total de uma venda, fechar venda, etc.*

Motivação

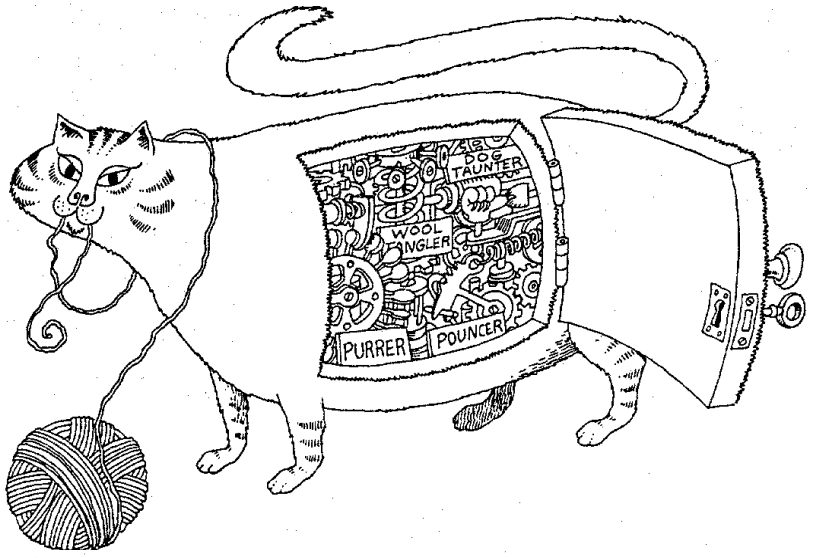
- *No exemplo anterior, dizemos que o programador deverá "levantar" uma abstração da realidade*
  - *Isso em função de um conjunto de dados que representa cada uma das entidades e suas operações*
  - *O termo "abstrato" significa "esquecer a forma de implementação" (ou a estratégia).*
  - *Em seguida, deve ser escolhida a forma (como) de representar tais dados na linguagem escolhida*
    - *Que recursos são oferecidos pela linguagem?*
- *A partir da **abstração**, estaremos dando início a modelagem de tipos complexos*

Definição

- ***Tipo Abstrato de Dado***
  - *É uma abstração da realidade, mediante a definição de um conjunto de dados que a representa e quais operações serão realizadas sobre estes dados*
- *Implementar um TAD numa linguagem de programação é encontrar uma forma de representá-la utilizando esta linguagem escolhida*
  - *Utilizando os tipos nativos e suas operações suportadas pelo computador*
  - *Em C, a implementação destes TADs se dá basicamente através de* ***ponteiros*** *e* ***estruturas***

Regras básicas de modelagem

1. ***Encapsulamento:*** *não é preciso saber os detalhes de implementação do TAD. Seu usuário irá se preocupar apenas com base nas operações oferecidas (interface)*



2. ***Modularização:*** *é a divisão de uma tarefa maior e mais complexa em tarefas menores e, provavelmente, mais fáceis de implementar e testar*

Boas práticas usando C

- **Antes:**
  - Para que as funções implementadas em um arquivo.c possam ser usadas por um outro módulo C, este precisa conhecer os cabeçalhos das funções oferecidas por arquivo.c
    - 1ª tentativa: repetição dos cabeçalhos das funções
    - E se um arquivo usa funções de vários outros módulos?
- **Depois:**
  - Um módulo de funções C normalmente é associado a um arquivo de mesmo nome e que contém apenas os cabeçalhos (arquivo.h)das funções oferecidas por este módulo e, eventualmente, os tipos de dados que ele exporte (typedef's, struct's, etc).

Boas pr

pilha.h

```
// definição estrutura
typedef struct  pilha{
        ...
}TPilha;
// Protótipos
// método responsável por...
Pilha* criaPilha( int tamanho );
// método responsável por...
int consultaTopo( void );
// método responsável por...
void empilha (Pilha* p, int elem);
// método responsável por...
int desempilha( void );
```

Extraia do TAD apenas os protótipos e crie uma biblioteca em um arquivo header

pilha.c

```
//implementação do TAD
#include "pilha.h"
#include<bibliotecas C>

// Corpo das funções
   com assinatura
```

Este arquivo .h irá conter toda a interface do TAD e definições de tipos. Deverá definir apenas os protótipos das funções e ser muito bem documentado. Ao final, ele poderá ser utilizado em qualquer outro programa, desde que informado no include.

prog3.c

prog2.c

prog1.c

```
/* main.c */
#include "pilha.h"
#include<bibliotecas C>
main() {
  /* declare e use o TAD */

}
```

Parte 2 :
Listas Lineares

Conceitos

- *Listas, por definição, são estruturas formadas por um* ***conjunto de dados*** *de forma a preservar a relação de* ***ordem linear*** *entre eles*
  - *lista telefônica, lista de clientes de uma agência bancária, lista de setores de disco*
- *Uma lista contém* ***nós****, os quais são acessados através de operações que esta lista possui*
- *Podemos* ***representar*** *as listas da seguinte forma:*

Listas Lineares

- *Propriedades de uma lista:*
  - *Existem **n** elementos na **sequência**;*
  - ***E1** é o primeiro elemento da sequência;*
  - ***En** é o último elemento da sequência;*
  - *Para todo i, j entre 1 e n, se i<j, então o elemento Ei antecede o elemento Ej;*
- *Portanto, podemos concluir que, em uma lista, existem elementos **sucessores** e **antecessores**:*
  - *E2 é sucessor de E1 e antecessor de E3*
  - *En não possui sucessores, mas E1 possui*

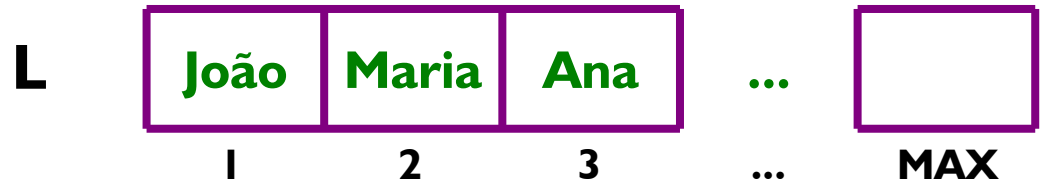
Operações

- *A literatura é unânime quanto às operações básicas realizadas numa estrutura lista:*
  1. ***criar*** *uma lista vazia*
  2. *verificar se uma lista* ***está vazia***
  3. *obter o* ***tamanho*** *da uma lista*
  4. ***obter/modificar*** *o valor do elemento em uma determinada posição na lista*
  5. *obter a* ***posição*** *de elemento cujo valor é dado*
  6. *inserir um* ***novo elemento*** *após (ou antes) uma determinada posição na lista*
  7. ***remover*** *um elemento em uma determinada posição da lista*
  8. ***exibir*** */ imprimir os elementos de uma lista*
  9. ***concatenar*** *duas listas*

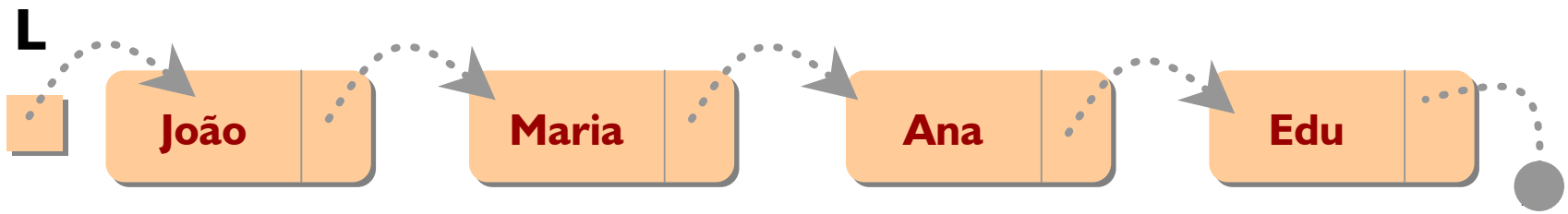
Tipos de listas

- *Neste estágio, estaremos trabalhando com dois tipos de listas lineares:* ***seqüencial*** *e* ***encadeada***

  ❶ ***Seqüencial:*** *os nós desta lista são armazenados em endereços seqüenciais. Materializada na forma de um* ***vetor*** *(arranjo ou matriz)*



  ❷ ***Encadeada:*** *seqüência de elementos encadeados por* ***ponteiros****, ou seja: dado + ponteiro próx. elemento*



Para um bom aproveitamento:

- *Resolva todas as questões da **lista de tipos abstratos***
- *Procure o professor ou monitor da disciplina e questione conceitos, listas, etc.*
- *Não deixe para codificar tudo e acumular assunto para a primeira avaliação.*
  - *Este é apenas um dos assuntos abordados na prova!*