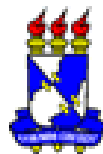


Variáveis

Prof. Alberto Costa Neto
alberto@ufs.br

Linguagens de Programação



Departamento de Computação
Universidade Federal de Sergipe

Retrospectiva

- Linguagem de Expressões (LE)
 - Valores e Avaliação de Expressões
 - Programa = Expressão
- Linguagem Funcional (LF)
 - LE + Abstração de Funções
 - Programa = Expressão
- Linguagem Imperativa (LI)
 - LE + Variáveis e Comandos
 - Um programa = **Comando** !!!



Variáveis

“Uma vez que o programador tenha entendido o uso de variáveis, ele entendeu a essência da programação”. (Dijkstra)

- Exagero?
 - Conceito de variável em LPs funcionais e lógicas é diferente do conceito nas LPs imperativas
- Paradigma Imperativo: **variável** é uma entidade que contém um valor, o qual pode ser inspecionado e atualizado sempre que necessário



Variáveis

- Abstração para células de memória
- Nomes dados a posições de memória
- Memória = coleção de células (modelo abstrato de armazenamento)



Variáveis

1) Espaço de memória é alocado e associado à variável X

Apenas um **NOME**

FF00 00000000

FF01 00000000

FF02 00000000

FF03 00000000

2) Valor 7 atribuído a X é colocado no espaço de memória associado a X

ENDEREÇO: posição da primeira célula ocupada

FF00 00000000

X — FF01 00000111

FF02 00000000

FF03 00000000

3) Valor corrente de X é somado a 3 e resultado é colocado no espaço de memória associado a X

FF00 00000000

X — FF01 00001010

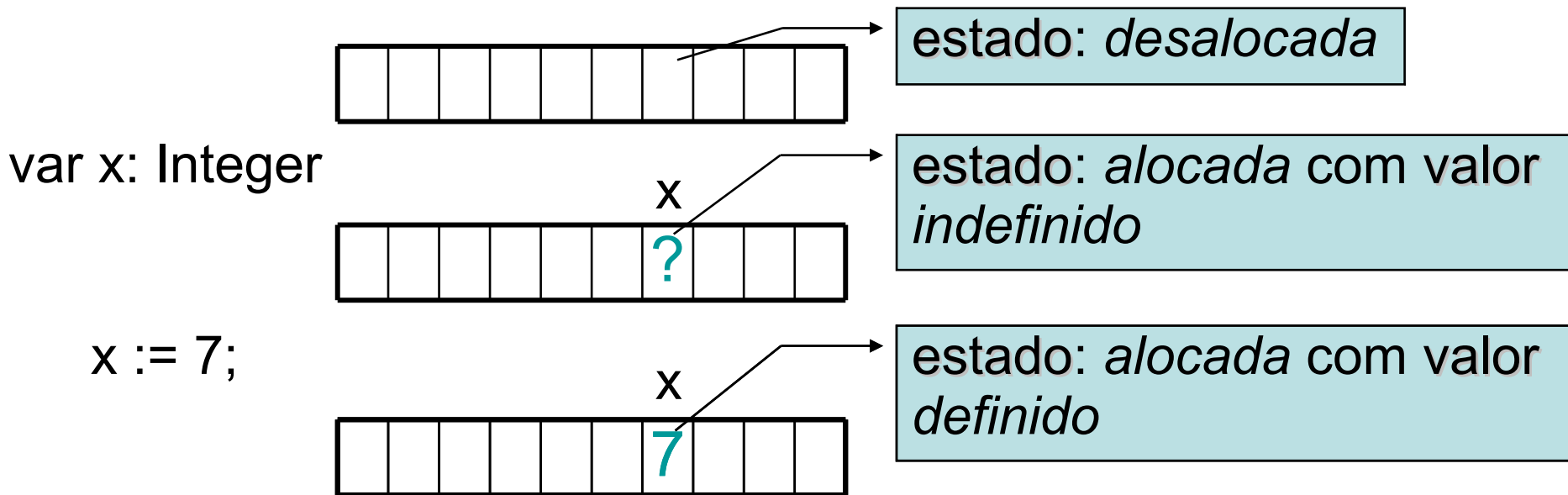
FF02 00000000

FF03 00000000



Variáveis

- Simplificação do modelo abstrato de armazenamento
 - memória: composta de células
 - célula possui um estado corrente e um valor



Variáveis

- Propriedades de uma Variável:
 - Nome
 - Valor (conteúdo do endereço de memória)
 - Tipo (faixa de valores possíveis e operações permitidas)
 - Tempo de Vida



Nome

- É obrigatório?
 - Não. Ex: variáveis dinâmicas (só são referenciadas através de ponteiros)
- Dois nomes diferentes podem representar a mesma célula de memória

C

```
int r = 0;  
int &s = r;  
s = 10;
```

Pascal

```
var r : integer;  
    s : integer absolute r;  
begin  
    r := 0;  
    s := 10; {r recebe 10}  
end.
```



Valor

- Considere em Pascal $n := n+1$
- As duas ocorrências de n descrevem entidades diferentes
 - O primeiro n representa uma célula de memória (endereço de memória)
 - O segundo n representa o conteúdo da célula



Tipo

- Especifica o conjunto de operações e valores que uma variável possui.
- A especificação dos tipos pode ser:
 - Explícita (C, C++, Java, Pascal)
 - Sintática (FORTRAN)
 - Variáveis iniciadas por I, J e K => inteiras
 - Semântica (ML)
 - Deduzido a partir das operações efetuadas sobre a variável



Tempo de Vida

- Intervalo de tempo entre a criação da variável (alocação) e a sua destruição (desalocação)
- Classificação do tempo de vida (vinculações de armazenamento)
 - Variáveis estáticas
 - Variáveis *stack*-dinâmicas
 - Variáveis *heap*-dinâmicas
 - Variáveis persistentes



Variáveis Estáticas

- Variáveis vinculadas a células de memória antes do início da execução do programa
 - Em tempo de compilação
- Funcionam como variáveis globais (ainda que definidas em blocos mais internos)
 - Tempo de vida = toda execução do programa
- Exemplos:
 - Java e C++: modificador *static*
 - Pascal não possui



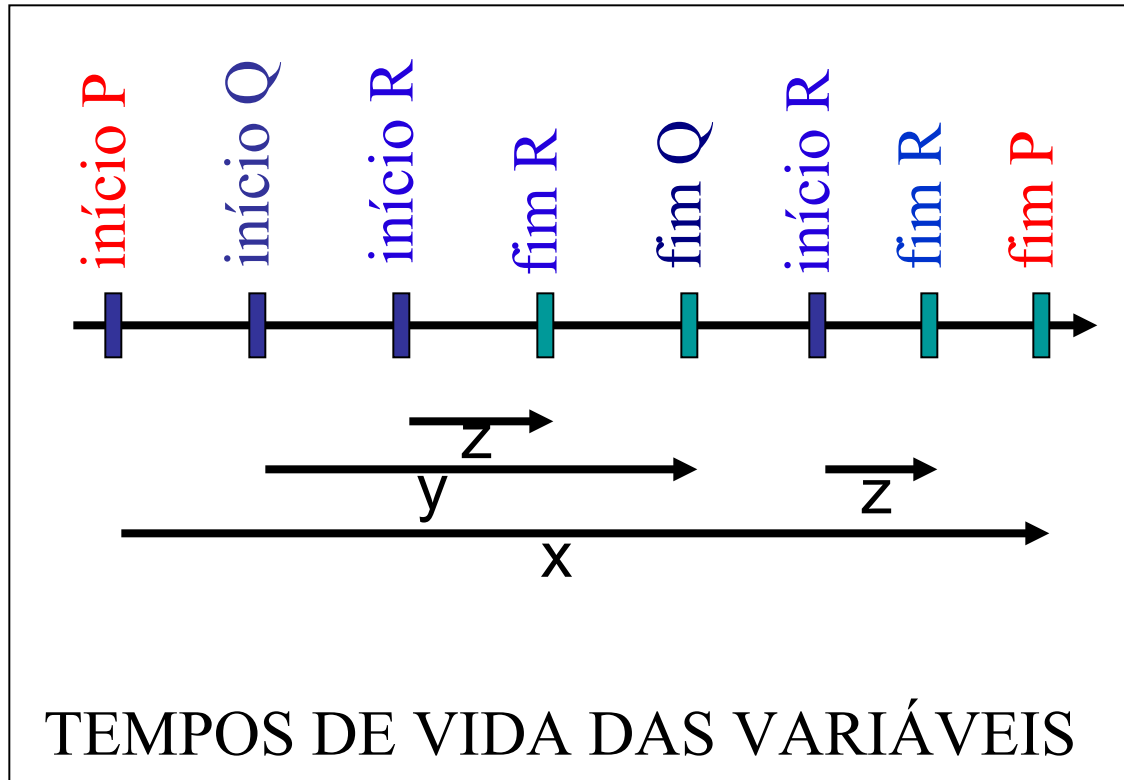
Variáveis *Stack*-Dinâmicas

- Variáveis criadas durante a elaboração de declarações
 - Alocação e vinculação com um endereço de memória resultado de uma declaração
- Variável local: declarada dentro de um bloco para ser usada somente dentro dele
- Variável global: declarada no bloco mais externo de um programa



Variáveis *Stack*-Dinâmicas

```
program P;  
  var x : ...;  
  procedure R;  
    var z : ...;  
  begin ...  
  end;  
  procedure Q;  
    var y : ...;  
  begin  
    ... R ...  
  end;  
begin  
  ... Q ... R ...  
end.
```



Variáveis *Heap*-Dinâmicas

- Pode ser criada e destruída em qualquer instante
 - Criada por um comando
 - São anônimas
 - Acessada através de um ponteiro
- A criação de uma variável heap é feita por um operador de alocação
 - Retorna um ponteiro para a nova variável
 - Exemplo Pascal: **new**



Variáveis *Heap*-Dinâmicas

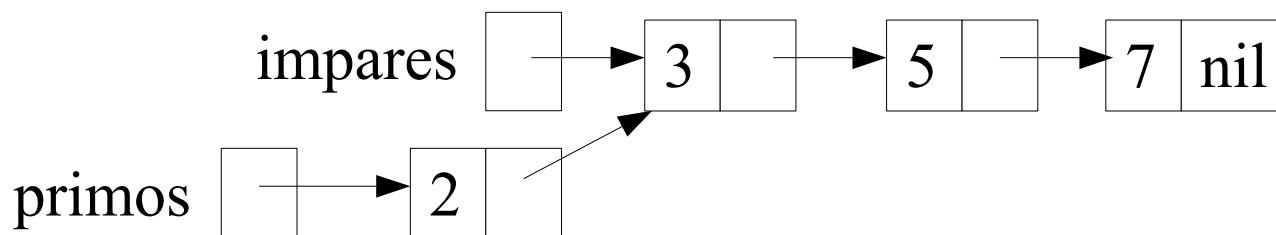
- Tempo de vida
 - Enquanto existir uma referência para a variável
 - Em Java: coleta de lixo automática
 - Enquanto não for desalocada por um **operador de desalocação**
 - Em Pascal: **dispose**
- Problema:
 - Operador de desalocação introduz Referências Soltas (Dangling References)
 - Ponteiro para uma variável que já foi desalocada



Variáveis *Heap*-Dinâmicas

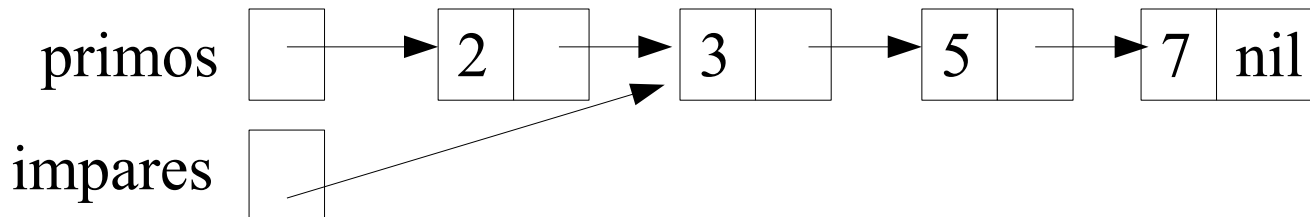
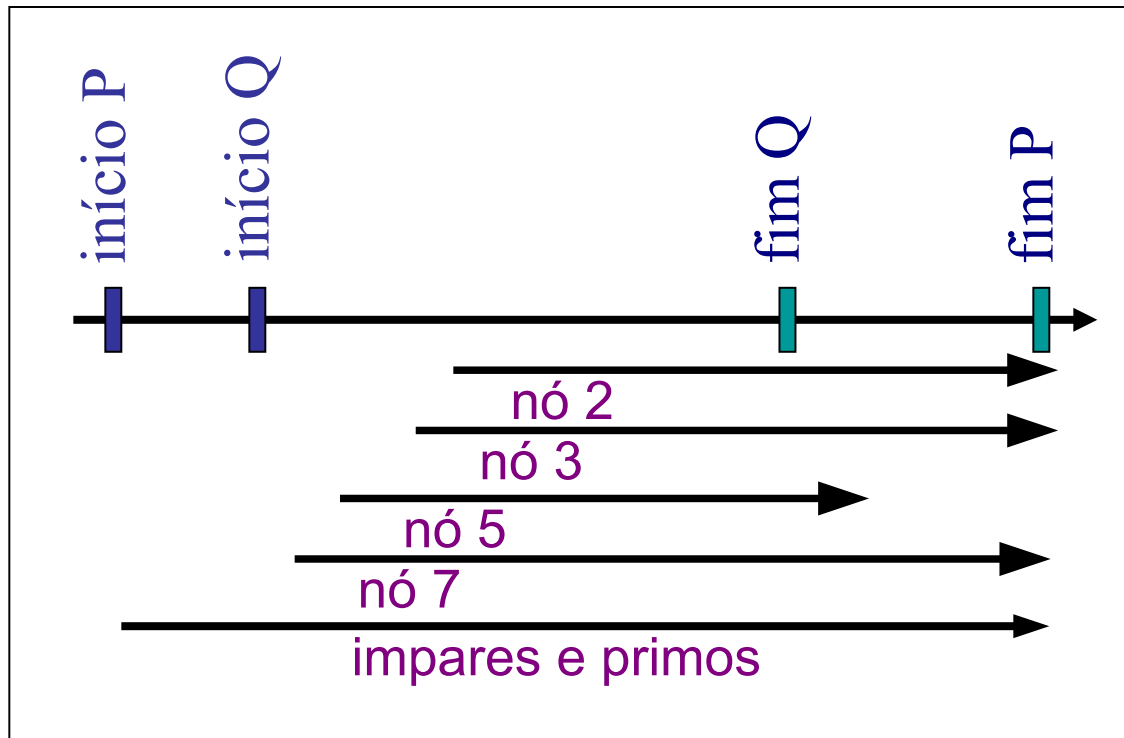
```
type IntList = ^IntNode;  
    IntNode = record  
        head : Integer;  
        tail : IntList  
    end;  
var impares, primos : IntList;  
function cons (h: Integer; t: IntList) : IntList;  
var l : IntList;  
begin  
    new(l);  
    l^.head := h; l^.tail := t;  
    cons := l;  
end;  
...  
impares := cons (3, cons (5, cons (7, nil)));  
primos := cons (2, impares);  
...
```

Pascal



Variáveis *Heap-Dinâmicas*

```
program P;  
  var impares, primos : ...;  
  procedure Q;  
  begin  
    impares := ...;  
    primos := ...;  
  end;  
begin  
  ... Q ...  
  remove nó 5  
  ...  
end.
```



Variáveis Compostas

- Variáveis Compostas são variáveis de um tipo composto
 - Ocupa várias células de memória
 - Constituída de variáveis mais simples (componentes)



Variáveis Compostas

- Exemplo:

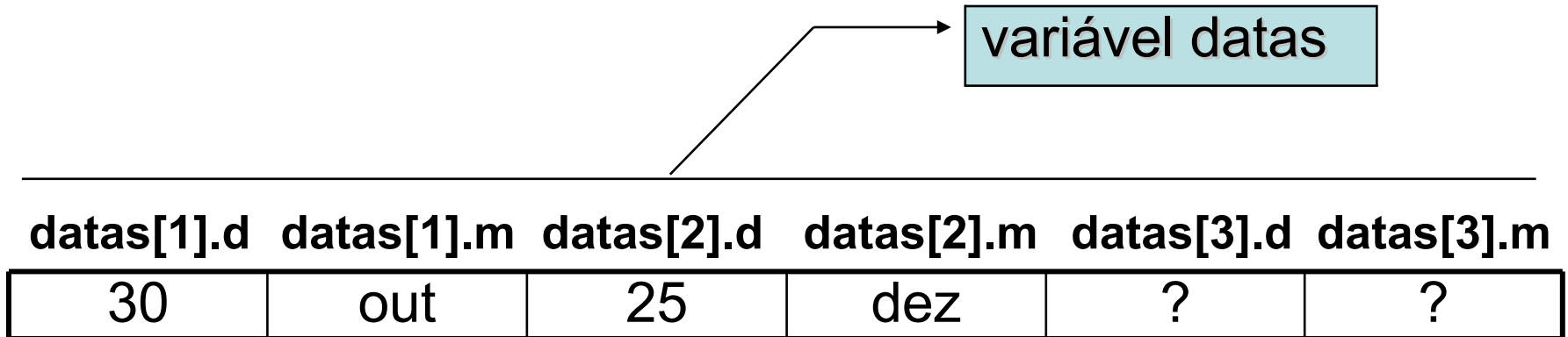
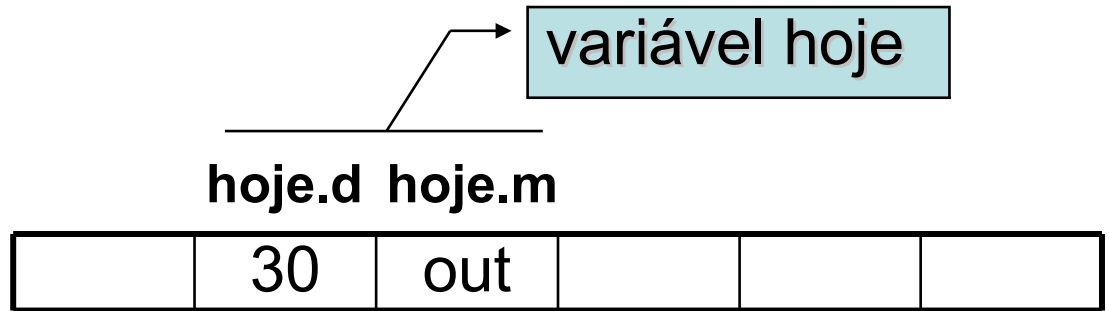
Pascal

```
type Mes = {jan, fev, mar, ..., out, nov, dez};  
      Dia = 1..31;  
      Data = record  
              d: Dia; m: Mes;  
            end;  
var hoje : Data;  
var datas : array[1..3] of Data;
```



Variáveis Compostas

```
...  
hoje.d := 30;  
hoje.m := out;  
...  
datas[1] = hoje;  
datas[2].m = dez;  
datas[2].d = 25;
```



Variáveis Compostas

- Inspeção e Atualização
 - Total: todos os componentes são alterados de uma só vez
 - Seletiva: os componentes são alterados passo a passo

...

hoje.d := 30;

hoje.m := out;

...

datas[1] = hoje;

datas[2].m = dez;

datas[2].d = 25;

Total e Seletiva

Seletiva



Variáveis Compostas

- Exemplo:

ML

Ñ possui atualização seletiva

```
type data = int*int  
val today : data ref = ref (14,12)
```



Total



Variáveis Compostas::Array

- Mapeamento:
Índices \rightarrow Variáveis componentes ($\text{int} \rightarrow \text{int}$)
- Índices finitos e consecutivos \rightarrow
componentes são armazenados em posições contíguas de memória
- Tipos de arrays
 - Estáticos
 - Semi-estáticos
 - Semi-dinâmicos
 - Dinâmicos



Variáveis Armazenáveis

- Variáveis compostas ocupam várias células na memória => Atualização seletiva afeta apenas um subconjunto dessas células
- Variáveis Armazenáveis
 - Tipos de valores que só podem ser armazenados em células simples
 - Não podem ser atualizados seletivamente
- Em Pascal: valores primitivos, conjuntos, ponteiros
- Em ML: primitivos, registros, listas, abstração de funções, referencias a variáveis



Constantes

- Predefinidas (literais)

```
char x = 'g';
```

```
int y = 3;
```

```
char* z = "bola";
```

```
/* int *w = &3; */ // Provocaria erro de compilação
```

```
*z = 'c';
```

- Declaradas em C

```
const float pi = 3.1416;
```

```
float raio, area, perimetro;
```

```
raio = 10.32;
```

```
area = pi * raio * raio;
```

```
perimetro = 2 * pi * raio;
```



Constantes

- Declaradas em C++

```
int* x;
```

```
const int y = 3;
```

```
const int* z;
```

```
// y = 4;
```

```
// y++;
```

```
// x = &y;    // x não é um apontador para uma constante int
```

```
z = &y;    // Permitido porque z é um apontador para uma  
           // constante inteira (const int)
```



Sugestões de Leitura

- Concepts of Programing Languages (Robert Sebesta)
 - Seções 5.2, 5.3, 5.4.3
- Programming Language Concepts and Paradigms (David Watt)
 - Capítulo 3 (Seções 3.1 até 3.4)
- Linguagens de Programação (Flávio Varejão)
 - Capítulo 4 (menos Seção 4.3.2)

