# Linguagens de Programação

## Variáveis

Bruno Lopes

Instituto de Computação
UFF

# Propriedades desejáveis

Legibilidade: A leitura do programa é facilmente compreendida?

Redigibilidade: A implementação reflete o algoritmo? A redação é sucinta?

Confiabilidade: É fácil detectar "enganos" do programador?

Eficiência: Roda rápido?

Facilidade de Aprendizado: É enxuta?

Ortogonalidade: Conceitos podem ser combinados livremente?

Reusabilidade: É possível aproveitar partes em outros programas?

Modificabilidade: É fácil alterar programas?

Portabilidade: Roda da forma esperada em diferentes plataformas?

Instituto de Computação

*Once a programmer has understood the use of variables, he has understood the essence of programming.*

Edsger Dijkstra

No paradigma imperativo!

Instituto de Computação

*Once a programmer has understood the use of variables, he has understood the essence of programming.*

Edsger Dijkstra

No paradigma imperativo!

Instituto de Computação

# Variáveis

**Definição**

Uma variável é uma entidade da computação que contém um valor, o qual pode ser inspecionado e atualizado sempre que necessário.

- abstração de célula de memória
- armazena o estado de uma entidade da computação.

Instituto de Computação

# Variáveis

**Definição**

Uma variável é uma entidade da computação que contém um valor, o qual pode ser inspecionado e atualizado sempre que necessário.

- abstração de célula de memória
- armazena o estado de uma entidade da computação.

Instituto de Computação

# Variáveis

**Definição**

Uma variável é uma entidade da computação que contém um valor, o qual pode ser inspecionado e atualizado sempre que necessário.

- abstração de célula de memória
- armazena o estado de uma entidade da computação.

Instituto de Computação

Variáveis de programação

*Vs*

Variáveis aritméticas

Instituto de Computação

## Variáveis simples

Uma única célula de memória.

## Variáveis composts

Um grupo contíguo de células de memória.

```
type S = (a,b);
  T = (c,d,e);
  P = record
    prim: S; seg: T;
  end;
var prod, prod2: P;
begin
  prod.prim := a;
  prod.seg := c;
  prod2 := prod;
end.
```

Instituto de Computação

## Variáveis simples

Uma única célula de memória.

## Variáveis composts

Um grupo contíguo de células de memória.

```
type S = (a,b);
  T = (c,d,e);
  P = record
    prim: S; seg: T;
  end;
var prod, prod2: P;
begin
  prod.prim := a;
  prod.seg := c;
  prod2 := prod;
end.
```

Instituto de Computação

# Características

**Nome:**
- definidos pelo programador
- existem variáveis que não possuem nomes
- linguagens costumam permitir sinônimos

**Endereço:** posição de memória da primeira célula ocupada pela variável

**Tipo:**

**Valor:** em conformidade com o tipo

Instituto de Computação

# Características

**Nome:**
- definidos pelo programador
- existem variáveis que não possuem nomes
- linguagens costumam permitir sinônimos

**Endereço:** posição de memória da primeira célula ocupada pela variável

**Tipo:**

**Valor:** em conformidade com o tipo

Instituto de Computação

# Características

**Nome:**
- definidos pelo programador
- existem variáveis que não possuem nomes
- linguagens costumam permitir sinônimos

**Endereço:** posição de memória da primeira célula ocupada pela variável

**Tipo:**

**Valor:** em conformidade com o tipo

Instituto de Computação

# Características

**Nome:**
- definidos pelo programador
- existem variáveis que não possuem nomes
- linguagens costumam permitir sinônimos

**Endereço:** posição de memória da primeira célula ocupada pela variável

**Tipo:**

**Valor:** em conformidade com o tipo

Instituto de Computação

# Características

**Nome:**
- definidos pelo programador
- existem variáveis que não possuem nomes
- linguagens costumam permitir sinônimos

**Endereço:** posição de memória da primeira célula ocupada pela variável

**Tipo:**

**Valor:** em conformidade com o tipo

Instituto de Computação

# Características

**Nome:**
- definidos pelo programador
- existem variáveis que não possuem nomes
- linguagens costumam permitir sinônimos

**Endereço:** posição de memória da primeira célula ocupada pela variável

**Tipo:**

**Valor:** em conformidade com o tipo

Instituto de Computação

# Características

**Nome:**
- definidos pelo programador
- existem variáveis que não possuem nomes
- linguagens costumam permitir sinônimos

**Endereço:** posição de memória da primeira célula ocupada pela variável

**Tipo:**

**Valor:** em conformidade com o tipo

Instituto de Computação

# Características

Tempo de Vida: Período em que existe memória alocada para a variável

| | |
|---|---|
| globais | por toda a execução do programa |
| locais | pela execução do bloco onde foram declaradas |
| dinâmicas (*heap*) | arbitrariamente |
| estáticas | a partir do ponto onde foram declaradas |
| persitentes | sobrevivem além do programa |

Instituto de Computação

# Características

Tempo de Vida: Período em que existe memória alocada para a variável

- globais: por toda a execução do programa
- locais: pela execução do bloco onde foram declaradas
- dinâmicas (*heap*): arbitrariamente
- estáticas: a partir do ponto onde foram declaradas
- persitentes: sobrevivem além do programa

Instituto de Computação

# Características

Tempo de Vida: Período em que existe memória alocada para a variável

- globais: por toda a execução do programa
- locais: pela execução do bloco onde foram declaradas
- dinâmicas (*heap*): arbitrariamente
- estáticas: a partir do ponto onde foram declaradas
- persitentes: sobrevivem além do programa

Instituto de Computação

# Características

Tempo de Vida: Período em que existe memória alocada para a variável

- globais: por toda a execução do programa
- locais: pela execução do bloco onde foram declaradas
- dinâmicas (*heap*): arbitrariamente
  - estáticas: a partir do ponto onde foram declaradas
  - persitentes: sobrevivem além do programa

Instituto de Computação

# Características

**Tempo de Vida:** Período em que existe memória alocada para a variável

- **globais** por toda a execução do programa
- **locais** pela execução do bloco onde foram declaradas
- **dinâmicas (*heap*)** arbitrariamente
- **estáticas** a partir do ponto onde foram declaradas
- **persitentes** sobrevivem além do programa

Instituto de Computação

# Características

**Tempo de Vida:** Período em que existe memória alocada para a variável

- **globais** por toda a execução do programa
- **locais** pela execução do bloco onde foram declaradas
- **dinâmicas (*heap*)** arbitrariamente
- **estáticas** a partir do ponto onde foram declaradas
- **persitentes** sobrevivem além do programa

Instituto de Computação

# Variáveis locais

```
procedure rec(b:integer);
begin
  if(b = 0) then
    rec(b+1);
end;

begin
  rec(0);
end.
```

Instituto de Computação

# *Heap*

Variáveis que residem no *Heap* podem ser criadas e apagadas a qualquer momento.

- Como são criadas?
- Como são acessadas?

Instituto de Computação

# *Heap*

Variáveis que residem no *Heap* podem ser criadas e apagadas a qualquer momento.

- Como são criadas?
- Como são acessadas?

Instituto de Computação

# Alocação de memória (*heap*)

Escolha do elemento

- Primeiro encontrado
- Tamanho mais próximo
- Maior tamanho

**Desafios**

- Fragmentação
- Alocar mais memória que o necessário
- Algoritmo de desfragmentação

Instituto de Computação

# Alocação de memória (*heap*)

Escolha do elemento

- Primeiro encontrado
- Tamanho mais próximo
- Maior tamanho

**Desafios**

- Fragmentação
- Alocar mais memória que o necessário
- Algoritmo de desfragmentação

Instituto de Computação

# Desalocação de memória (*heap*)

Escolha do elemento

Pascal,C,C++: explícita — a cargo do programador.

Java: implícita — coleta de lixo.

Explícita: eficiente, porém mais trabalhoso e menos confiável.

Implícita: confortável e segura, mas pode gerar *overhead*.

Instituto de Computação

# Desalocação de memória (*heap*)

Escolha do elemento

Pascal,C,C++: explícita — a cargo do programador.

Java: implícita — coleta de lixo.

Explícita: eficiente, porém mais trabalhoso e menos confiável.

Implícita: confortável e segura, mas pode gerar *overhead*.

Instituto de Computação

# Métodos de alocação de variáveis

Estática: em tempo de carga

- Mau dimensionamento das variáveis
- Espaço desperdiçado com subrotinas que podem não ser executadas
- Impedimento de uso de recursividade

Dinâmica: contígua no vetor de memória

- Esgotamento rápido do vetor
- Desalocação e realocação pouco eficientes

Pilha + *Heap*

Instituto de Computação

# Persistência

**Transientes:** tempo de vida da variável limitado pela ativação do programa que a criou.

**Persistentes:** tempo de vida da variável transcende a ativação do programa que a criou.

Instituto de Computação

Incorporar mecanismos de persistência de dados em linguagens para facilitar a programação em aplicações onde a persistência é necessária.

LP onde não exista diferença entre entidades transientes e persistentes (Persistência Ortogonal)

Instituto de Computação

# Arquivos

Arquivos Seriais

Arquivos Diretos: implementação com tabela indexada ou vetor de componentes.

Operações de Abertura e Fechamento: conversão de dados para formato sequencial binário.

Instituto de Computação

# Persistência Ortogonal

- Mesmos tipos para variáveis persistentes e transientes.
- Nenhuma distinção entre o código que lida com variáveis persistentes e o que lida com variáveis transientes.
- Identificação de persistência através da percepção da continuidade do uso.
- Eliminação de Conversões de Entrada e Saída (30% do código).
- Não existem ainda na prática.

Instituto de Computação

# Persistência Ortogonal

- Mesmos tipos para variáveis persistentes e transientes.
- Nenhuma distinção entre o código que lida com variáveis persistentes e o que lida com variáveis transientes.
- Identificação de persistência através da percepção da continuidade do uso.
- Eliminação de Conversões de Entrada e Saída (30% do código).
- Não existem ainda na prática.

Instituto de Computação

# Persistência Ortogonal

- Mesmos tipos para variáveis persistentes e transientes.
- Nenhuma distinção entre o código que lida com variáveis persistentes e o que lida com variáveis transientes.
- Identificação de persistência através da percepção da continuidade do uso.
- Eliminação de Conversões de Entrada e Saída (30% do código).
- Não existem ainda na prática.

Instituto de Computação

# Persistência Ortogonal

- Mesmos tipos para variáveis persistentes e transientes.
- Nenhuma distinção entre o código que lida com variáveis persistentes e o que lida com variáveis transientes.
- Identificação de persistência através da percepção da continuidade do uso.
- Eliminação de Conversões de Entrada e Saída (30% do código).
- Não existem ainda na prática.

Instituto de Computação

# Persistência Ortogonal

- Mesmos tipos para variáveis persistentes e transientes.
- Nenhuma distinção entre o código que lida com variáveis persistentes e o que lida com variáveis transientes.
- Identificação de persistência através da percepção da continuidade do uso.
- Eliminação de Conversões de Entrada e Saída (30% do código).
- Não existem ainda na prática.

Instituto de Computação