Linguagens de Programação

Expressões e 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

Expressões

Definição

Uma frase do programa que, ao ser avaliada, produz como resultado um valor.

Elementos:

- Operadores
- Operandos
- Resultado

Instituto de Computação

Expressões

Definição

Uma frase do programa que, ao ser avaliada, produz como resultado um valor.

Elementos:

- Operadores
- Operandos
- Resultado

Instituto de Computação

Expressões

Definição

Uma frase do programa que, ao ser avaliada, produz como resultado um valor.

Elementos:

- Operadores
- Operandos
- Resultado

Instituto de Computação

Expressões

Classificação

- Simples
- Composta

Notação:

- Prefixada
- Infixada
- Posfixada

```
x>y?x:y
```

Instituto de Computação

Expressões

Classificação

- Simples
- Composta

Notação:

- Prefixada
- Infixada
- Posfixada

```
x>y?x:y
```

Instituto de Computação

Expressões

Classificação

- Simples
- Composta

Notação:

- Prefixada
- Infixada
- Posfixada

```
x>y?x:y
```

Instituto de Computação

Aridade dos operadores

```
(+ 1)
(+ 1 2)
(+ 1 2 4)
(+ 3 4 1 2)
...
```

Instituto de Computação

Aridade dos operadores

```
#define JUST3(a, b, c, ...) (a), (b), (c)
#define FUNC(...) func(JUST3(__VA_ARGS__, 0, 0))

FUNC(x) --> func((x), (0), (0))

FUNC(x,y) --> func((x), (y), (0))
```

Instituto de Computação

Aridade dos operadores

```
#define JUST3(a, b, c, ...) (a), (b), (c)
#define FUNC(...) func(JUST3(__VA_ARGS__, 0, 0))

FUNC(x) --> func((x), (0), (0))

FUNC(x,y) --> func((x), (y), (0))
```

Instituto de Computação

Origem dos operadores

- Pré-existentes
- Definidos pelo programados
- Composição de operadores

```
val par = fn (n: int) => (n mod 2 = 0)
val negação = fn (t:bool) => if t then false else true
val impar = negação o par
```

Instituto de Computação

Origem dos operadores

- Pré-existentes
- Definidos pelo programados
- Composição de operadores

```
val par = fn (n: int) => (n mod 2 = 0)
val negação = fn (t:bool) => if t then false else true
val impar = negação o par
```

Instituto de Computação

Tipos de operações

- Literais
- Constantes
- Variáveis

- Estáticas
- Dinâmicas

- Aritméticas
- Reais
- Booleanas

- Binárias
- Condicionais

- Referenciamento
- Derreferenciamento

- Categóricas
- Interativas

Instituto de Computação

Literais

Resultam no valor explícito do texto do programa.

Instituto de Computação

Agregação

```
void f(int i) {
     int a[] = {3 + 5, 2, 16/4};
     int b[] = {3*i, 4*i, 5*i};
     int c[] = {i + 2, 3 + 4, 2*i};
  }
```

Instituto de Computação

Construtores

```
Data d = new Data()
```

Instituto de Computação

Binárias

```
void main() {
     int j = 10;
     char c = 2;
     printf(\%d\n", ~c);        /* imprime -3 */
     printf(\%d\n", j & c);     /* imprime   2 */
     printf(\%d\n", j | c);     /* imprime 10 */
     printf(\%d\n", j ^ c);     /* imprime   8 */
     printf(\%d\n", j << c);    /* imprime 40 */
     printf(\%d\n", j >> c);    /* imprime   2 */
  }
```

Instituto de Computação

Efeitos colaterais

```
x = 3.2 * ++c;
x = 2;
y = 4;
z = (y = 2 * x + 1) + y;
```

Instituto de Computação

Funções e efeitos colaterais

- atualização de variável global
- passagem por referência
- impressão de valores ou gravação em arquivo

Instituto de Computação

Avaliando expressões compostas

Precedência de operadores

```
if a > 5 and b < 10 then
```

Ausência de precedência

Falta de redigibilidade!

Instituto de Computação

Avaliando expressões compostas

Precedência de operadores

```
if a > 5 and b < 10 then
```

Ausência de precedência

Falta de redigibilidade!

Instituto de Computação

```
x = a + b - c;
y = a < b < c;

x = **p;
if (!!x) y = 3;
y = !x++; // Como avaliar?
```

Instituto de Computação

```
i = 2;
a[i] = i++; // Atualiza a[2] ou a[3]?
```

Instituto de Computação

Curto-circuito

```
i = 0;
enquanto ((i < n)  e (a[i] <> valor procurado))
       incrementa i;
fim
```

```
i = 1;
if (b < 2*c || a[ i ++ ] > c) { a[i]++ };
```

Instituto de Computação

Pascal: expressões não são avaliadas em curto circuito, a menos que o compilador implemente por diretiva de compilação.

C, C++: && e ||, operadores booleanos, são avaliadas em curto circuito; & e |, operadores binários, não são avaliadas em curto circuito.

Java: & e | não são avaliadas em curto circuito; && e || são avaliadas em curto circuito.

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