UNIP

UNIVERSIDADE PAULISTA

Paradigmas de Linguagens

Aula 3: Expressões e Instruções de Atribuição

Professora Sheila Cáceres

Roteiro

- Introdução
- Expressões aritméticas
- Conversões de tipo
- Expressões Relacionais e Booleanas
- Instruções de atribuição

Introdução

- Expressões são o meio fundamental de especificar computações em uma linguagem de programação.
- Para entender a avaliação de expressões é necessário estar familiarizado com as ordens de avaliação de operadores e de operandos.

Expressões Aritméticas

- A avaliação automática de expressões aritméticas foi uma das principais metas das primeiras linguagens de programação.
- A finalidade de uma expressão aritmética é especificar uma computação aritmética. A implementação deve causar:
  - Buscar operandos
  - Executar as operações aritméticas sobre eles.

Expressão Aritmética

Uma expressão Aritmética consiste de:

- **operadores e operandos:** Um operando pode ser considerado como uma das entradas de um operador.

  Por exemplo, em ***3* + *6***,

  - **+** é o operador
  - ***3*** e ***6*** são os operandos.
- **Parênteses ( )**
- **chamadas a função**

Exemplo

Double da = (5.8 + 4.5) / db;

Operadores

Tipos de Operadores

- **Unários** – Um operador é unário quando tem um operando (-, ++).

  Ex: a++

- **Binários** – dois operandos (-, +, *, /).
  - Ex: a + b
- **Ternário** – três operandos (? :)
  - Ex: (condição) ? Verdadeiro : Falso

    (A < B) ? 1 : 0

Ordem de Avaliação de Operadores

- Como deve ser avaliada a seguinte expressão?

  a + b * c / d;

- Qual poderia ser?
  - ((a+b) * c) / d
  - a + ((b* c) / d)
  - a + (b* (c / d))

  Resposta: depende de vários factores mas na maioria de linguagens modernas funciona igual que na matemática tradicional.

Ordem de Avaliação de Operadores

Para avaliar a ordem dos operadores em que devemos resolver as operações aritméticas, precisam-se considerar:

- Regra de Precedência
- Regra de Associatividade
- Parênteses
- Expressões Condicionais

Precedência

- As regras de precedência de operadores definem a **ordem** na qual operadores de diferentes níveis de precedência são avaliados.
- Níveis de precedência comuns

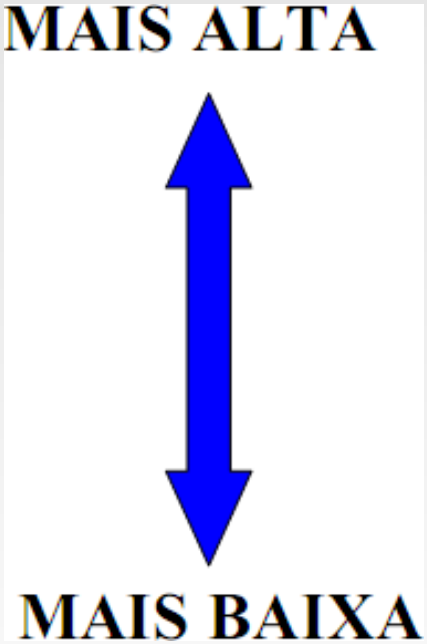
1. operadores unários
2. ** ou ^ (se a linguagem o suporta)
3. *, /
4. +, -

Ver exemplo externo

Precedência

- Operadores de precedência mais alta numa expressão aritmética serão resolvidos antes que os operadores de baixa precedência

Exemplos:



FORTRAN	PASCAL	C	Ada
**	*, /, **div, mod**	++ , -- (pós-fixo)	**, **abs**
*, /	+ , - (todos)	++ , -- (prefixo)	*, /, **mod**
+, - (todos)		+ , - (unário)	+, - (unário)
		*, / , %	+, - (binário)
		+, - (binário)	

Associatividade

- Regras de associatividade para avaliação de expressões definem em qual ordem operadores **adjacentes da mesma precedência** são avaliados
  - EX: a+b+c+d (esquerda para direita ou direita para esquerda?)
- Regra típica
  - Esquerda para a direita,
  - As vezes exponenciação que é da direita para a esquerda.
- Em APL todos os operadores têm precedência igua e são associativos da direita para a esquerda

Ver exemplo externo

Associatividade

Linguagem	Regra de Associatividade
FORTRAN	Esquerda : *, / , + , - Direita : **
Pascal	Esquerda : Todos
C	Esquerda : ++ pós-fixo, -- pós-fixo, * , / , % , + binário , - binário Direita : ++ prefixo, -- prefixo, + unário , - unário
C++	Esquerda : * , / , % , + binário , - binário Direita : ++, -- , + unário , - unário
Ada	Esquerda : todos, exceto ** Direita : **

Parênteses

- Podem-se alterar as regras de precedência e de associatividade colocando parênteses nas expressões
- Linguagens que permitem parênteses em expressões aritméticas poderiam dispensar todas as regras de precedência e simplesmente associar todos os operadores da esquerda para a direita ou vice-versa
  - Opção feita na linguagem de programação APL.
- Exemplo: ***(a+b)*c*** → soma é feita primeiro devido aos parenteses, mesmo que a multiplicação tenha precedência sobre a adição.

Ver exemplo externo

Expressões Condicionais

- Também podemos realizar operações aritméticas dentro de expressões condicionais.
- **Operador ternário *? :***
  - C, C++ e Java
  - Exemplo:

```
res = (cont == 0)? 0 : soma/cont
```

  - Equivalente – if-then-else

```
if (cont == 0)
    res = 0
else res = soma /cont
```


Ver exemplo externo

Ordem de avaliação de operandos

Além dos operadores, também devemos considerar os operandos que participam nas operações aritméticas. A seguir ilustra-se cada operando e como devemos obter os valores que representam.

Os operandos podem ser:

- ***Variáveis***:
  - Buscar seu valor na memória
- ***Constantes***:
  - Algumas vezes é necessário buscar na memória;
  - Outras vezes a constante está na própria instrução de máquina.
- ***Expressões parêntizadas:***
  - avaliar todos operandos primeiro antes que seu valor possa ser usado como operando
- Funções por referência:
  - O caso de maior interesse, pois a ordem da avaliação é crucial.

Sobrecarga de Operadores

- Usar um operador para mais do que um propósito
- Exemplo
  - Em Java
    - + para adição de quaisquer operandos de tipo numérico
    - Em Java (+) para concatenar cadeias.
  - Em C:
    - A = B * C // Multiplicação
    - A = * ref; // Referência

Sobrecarga de Operadores

- C++ e Ada permitem que programador defina a sobrecarga de operadores.

- Problema potencial:
  - Programadores podem definir sobrecarga de operadores sem sentido;
  - Legibilidade pode ficar comprometida.

Conversões de Tipo

- Uma conversão de estreitamento
  - transforma um valor para um tipo que não pode armazenar todos os valores do tipo original
    - double para float (double é muito maior do que float)
- Uma conversão de alargamento
  - transforma um valor para um tipo que pode incluir, pelo menos, aproximações de todos os valores do original
    - float para double (float é menor do que double)
- Uma expressão de modo misto é aquela que possui operandos de tipos diferentes.
  - Exemplo?

Coersões em Expressões

- Um operador pode ter operandos de tipos diferentes (expressões de modo misto).
- Deve se definir convenções para as conversões de tipos porque as operações binárias usualmente usam operandos do mesmo tipo.
- Quando os operandos não são do mesmo tipo, o compilador escolhe um dos operandos para ser coagido.

```
double a, d=20.5;
int b=10;
a = (b*d);
System.out.println(a);
```

```
Saida:
205.0

(b é coagido a double)
```

Conversão de Tipo Explícita

- Chamada de *casting* em linguagens baseadas em C
  - Exemplos
    - C: (int) numero
    - Ada: Float (soma)
  - Obs: a sintaxe em Ada é similar a chamada de funções

Erros em Expressões

- Erros em Expressões (causados por):
  - Limitações aritméticas:
    - Ex. Divisão por zero
  - Limitações da aritmética computacional:
    - Ex. overflow de inteiros
    - Overflow de ponto flutuante

Expressões Relacionais e Booleanas

Expressões Relacionais

- Tem dois operandos e um operador relacional
- Um operador relacional **compara** os valores de seus dois operandos.
- O valor desta expressão é booleano (exceto quando a linguagem não define booleano).
- Os operadores relacionais normalmente são sobrecarregados para uma variedade de tipos.
- Os símbolos de operadores variam bastante entre linguagens.

Expressões Relacionais

Operação	Ada	Java	FORTRAN 90
Igual	=	==	.EQ. ou ==
Diferente	/=	!=	.NE. ou <>
Maior que	>	>	.GT. ou >
Menor que	<	<	.LT. ou <
Maior que ou igual	>=	>=	.GE. ou >=
Menor que ou igual	<=	<=	.LE. ou <=

Expressões Relacionais

- Operadores relacionais sempre tem menor precedência do que os aritméticos.
  - Ex: a + 1 < 2 * b

    expressões aritméticas são avaliadas primeiro

Ver exemplo externo

Expressões Booleanas

- Operandos são booleanos.
- O resultado é booleano

FORTRAN 77	Fortran 90	C	Ada
.AND.	and	&&	and
.OR.	or	||	or
.NOT.	not	!	not
			xor

- Também precisam ordem de precedência (usualmente Not, And, Or) e associatividade.
- Exemplo
  - If (True && False)

    Console.WriteLine("O resultado de True && False é False ")
  - If (a > b) || (c < d) ……..
- Característica do C
  - não possui tipo booleano
    - utiliza o tipo int com 0 para FALSO
    - diferente de zero para VERDADEIRO

Instruções de Atribuição

Atribuição Simples

- A sintaxe geral da instrução de atribuição simples é:

  <variável_alvo> <operador_de_atribuição> <expressão>

  - Exemplo: ***a*** **=** ***10***;
- Operadores de atribuição:
  - (=) → FORTRAN, BASIC, PL/I, C, C++, Java:
    - Pode ser ruim se sobrecarregado para o operador relacional de igualdade:
  - (:=) → ALGOL, Pascal, Modula-2, ADA.

Alvos Múltiplos

- Permitir a atribuição do valor da expressão a mais de uma localização.
- Exemplos
  - **PL/I:** A, B = 10;
  - **C, C++, C#, Java:** A = B = C = 10;

Alvos Condicionais

- Permite atribuir um valor dependendo do resultado de uma condição.
- Exemplo em C e java:

```
(flag) ? count1 : count2 = 0;
```

Usou-se o operador ? : com 3 operandos

  - Equivalente a

```
if(flag) cont1=0;
else cont2=0;
```

- Exemplo:
  - int b = (a >= 10)? 20: 30; // O valor de b depende do resultado da condição entre parenteses.

Operadores de atribuição compostos

- Método abreviado de especificar uma forma de atribuição comumente necessária.
  - soma + = valor;

  Realmente quer dizer:
  - Soma = soma + valor;
- Sintaxe: concatenação do operador binário desejado com o operador =.
- (C, C++ e Java):

Operadores Unários

- Linguagens baseadas em C combinam operações de incremento e de decremento com atribuição
  - Exemplos
    - soma = ++ cont (o valor de cont é incremento, e depois atribuído a soma)
    - soma = cont++ (atribui-se cont a soma, e depois cont é incrementado)
    - cont++ (cont é incrementado)
    - -cont++ (cont é incrementado e depois é transformado em negativos
      - Não (-cont)++

Referências Bibliográficas

* Sebesta, Robert. Concepts of Programming Languages, Novena Edição.

* Varejão, Flavio. Linguagens de Programação. Campus, 2004.