

Linguagens de Programação

Expressões e Declaração de Atribuição

Baseado em Conceitos de Linguagens de Programação – Robert W. Sebesta

Prof. Lucas Ismaily

Universidade Federal do Ceará
Campus Quixadá

Roteiro

1. Introdução
2. Expressões Aritméticas
3. Sobrecarga de Operadores
4. Conversão de Tipos
5. Expressões Relacionais e Booleanas
6. Avaliação Curto-Circuito
7. Instruções de Atribuição
8. Atribuição de Modo Misto

Introdução

- Qual o resultado das seguintes expressões em C, C++ e Java:

`int a = 1, b = 2, c = 3;`

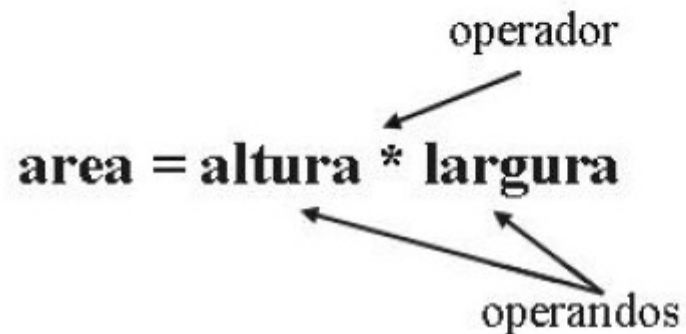
- 1) `-a*-b+-c;`
- 2) `++a+a++;`
- 3) `+a++++ ++b+---c;`

- Serão equivalentes as seguintes expressões?

- 1) `-a-b` \Leftrightarrow `-(a-b)` \Leftrightarrow `(-a) - (b)` ?
- 2) `-a*-b+-c` \Leftrightarrow `-(a*(-b))+ (-c)` ?

Introdução

- **Semântica das expressões** – significado de cada operação e operando numa expressão, tendo em atenção a ordem pela qual as operações são executadas e suas afectações.
- Operandos e operadores:



Expressões Aritméticas

- Expressões Aritméticas:
 - A avaliação de expressões aritméticas foi uma das principais motivação para o desenvolvimento da primeira linguagem de programação de alto nível;
 - Expressões aritméticas consistem de operadores, operandos, parênteses e invocações de funções.

Expressões Aritméticas

- Questões de projecto:
 - Quais são as regras de prioridade de operadores?
 - Quais as regras de associatividade dos operadores?
 - Qual a ordem de avaliação dos operandos?
 - Existem restrições nos efeitos colaterais da avaliação dos operandos?
 - A linguagem permite definição de sobrecarga de operadores pelo programador?
 - Que modos mistos (em termos de operadores) existem para expressões?

Expressões Aritméticas

- Tipos de Operadores:
 - Unário - possui apenas um operando;
 - Binário - possui dois operandos;
 - Ternário - possui três operandos;
- Regras de precedência dos operadores - definem a ordem pela qual os operadores adjacentes, de diferentes precedências, são avaliados. (Adjacentes - separados no máximo por um operando).

Expressões Aritméticas

- Níveis de precedência típicos:
 - Parênteses
 - Operadores unários
 - $**$ (se a linguagem suportar a exponenciação)
 - $*$, $/$
 - $+$, $-$
- Regra de associatividade de operadores -
Define a ordem em que operadores adjacentes, com o mesmo nível de precedência, são avaliados.

Expressões Aritméticas

- Regras de associatividade típicas:
 - Da esquerda para a direita, excepto **, que é da direita para a esquerda;
 - Alguns operadores unários associam-se da direita para esquerda (Ex. FORTRAN);
 - APL é diferente - todos operadores possuem igual precedência e todos são associados da direita para esquerda;
 - Precedência e associatividade podem ser alteradas através da utilização de parênteses.

Expressões Aritméticas

- Ordem de avaliação de operandos:
 - Variáveis: é necessário obter o seu valor;
 - Constantes: algumas vezes é necessário buscá-las à memória; algumas vezes esta encontra-se na própria instrução máquina.
 - Expressões com parênteses: avaliar-se primeiro todos os operandos e operadores dentro dos parênteses.
 - Referência a funções: caso muito importante! A ordem de avaliação é fundamental.

Expressões Aritméticas

- Efeito colateral de funções - quando uma função altera um parâmetro de E/S ou uma variável não local.
- O problema do Efeito Colateral de funções :
 - Como avaliar os operandos de uma expressão quando uma função referenciada altera outro operando da expressão ex. seu parâmetro.
a = 10;
b = a + fun(&a);
/* Assumindo que fun retorna o parâmetro dividido por 2 e modifica o parâmetro para o valor 20 */
Valor de b: 15 se "a" for avaliado primeiro
 25 se fun for avaliado primeiro

Expressões Aritméticas

- Duas possíveis soluções:
 - I) Escrever a definição da linguagem para não permitir efeitos colaterais de funções:
 - Funções sem parâmetros de entrada/saída;
 - Permitir somente referências locais em funções;
 - Vantagem: Funciona!!
 - Desvantagem: Programadores querem a flexibilidade de parâmetros de entrada/saída.
 - II) Escrever a definição da linguagem exigindo a avaliação dos operandos em ordem fixa.
 - Desvantagem: limita algumas otimizações do compilador.

Expressões Aritméticas

- Expressões Condicionais

- Ex. em C, C++, e Java

```
average = (count == 0) ? 0 : sum / count;
```

É equivalente a:

```
if(count==0)
```

```
{ average = 0; }
```

```
else
```

```
{ average = sum / count; }
```

Sobrecarga de Operadores

- Sobrecarga de Operadores
 - Comum para alguns (ex.: '/' para *int* e *float*)
 - Alguns problemas potenciais:
 - Ex. C: '*' //multiplicação ou desreferência?
 - Ex. C: `result = * soma;` // desreferência ou falta de operando?
 - Ex. C: `media = soma / cont;` // divisão '/' é inteira ou real? Estes problema pode ser evitado pela introdução de novos símbolos (Ex. Pascal: `result := soma div cont;`)

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.
- A sobrecarga de operadores foi um dos recursos do C++ não copiado para o Java.

Conversão de Tipos

- Conversão de tipos:
 - Efectuada de forma implícita – coerção (*Coercion*)
 - Efectuada de forma explícita – (*cast*)
- Conversão de estreitamento - converte um objecto para um tipo que não inclui todos os valores do tipo original (ex.: *double* para *int*).
- Conversão de alargamento – converte um objecto para um tipo que inclui pelo menos aproximações para todos os valores do tipo original (*int* para *double*).
- Expressão de modo misto – expressão que inclui operandos de vários tipos diferentes.

Conversão de Tipos

- Desvantagem de coerções:
 - Diminuem a capacidade de detecção de erro por parte do compilador.
- Na maioria das linguagens todos os tipos numéricos permitem coerção em expressões, usando conversão de alargamento.
- Modula-2 e Ada não permitem coerção de tipos em expressões.

Conversão de Tipos

- Conversão Explícita ("*cast*"): Conversão dada por uma instrução.
 - Ex. em Ada:
 `FLOAT(INDEX) -- INDEX is INTEGER type`
 - Ex. em C:
 `(int) speed /* speed is float type */`

Conversão de Tipos

- Erros em Expressões (causados por):
 - Limitações aritméticas:
 - Ex. Divisão por zero
 - Limitações da aritmética do computador:
 - Ex. overflow de inteiros
overflow, underflow de virgula flutuante
- Vários erros somente são detectados em tempo de execução e caso a linguagem não possua tratamento de exceções haverá um final *anormal* do programa.

Expressões Relacionais

- Expressões Relacionais
 - Expressão composta por um operador relacional e dois operandos de vários tipos.
 - As expressões são avaliadas para alguma representação lógica (resultado é booleano);
 - Os símbolos dos operadores relacionais variam entre as diversas linguagens.

Ex.: !=, / =, .NE., <>, #

Expressões Booleanas

- Expressões Booleanas

- Operandos são booleanos e resultado é booleano.

- Operadores:

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

- C não possui tipo booleano – utiliza o tipo *int* com 0 para FALSO e diferente de zero para VERDADEIRO.
- Em C a expressão: $a < b < c$ é correcta e equivalente a: $(a < b) < c$

Precedência dos Operadores

- Precedência dos operadores:

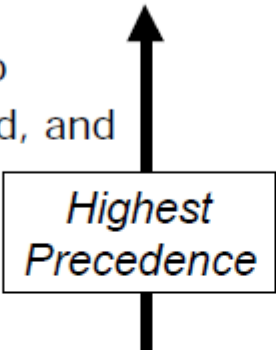
- Pascal:

- not, - unário
- *, /, div, mod, and
- +, -, or
- relops

- Ada:

- **
- *, /, mod, rem
- - unário , not
- +, -, &
- relops
- and, or, xor

*Highest
Precedence*



- C, C++, e Java possuem mais de 50 operadores

- **+ de 17 níveis diferente de precedências**

Avaliação Curto-circuito

- Avaliação Curto-circuito é uma expressão que tem um resultado determinado sem avaliar todos os seus operandos.

Ex. C: `if(a>0 && b<50) //se a<0 --> b<50 não é avaliado.`

- Problema de avaliação curto-circuito:
utilização intensa da tabela de símbolos do compilador.

- Avaliação curto-circuito expõe o potencial problema de efeito colateral em expressões

Ex. C: `(a > b) || (b++ / 3) // pode não ser executado`

Avaliação Curto-circuito

- C, C++, e Java: utilizam avaliação curto-circuito para operadores booleanos (&& e ||), e não curto-circuito para operações de bit (& e |)
- Ada: programadores podem especificar o modo da avaliação.
- Pascal: não utiliza avaliação curto-circuito.
- FORTRAN 77: possui curto-circuito mas efeitos colaterais são deixados em estado indefinido.

Instruções de atribuição

- Instruções de atribuição – mecanismo que permite modificar dinamicamente as vinculações de valores a variáveis.
 - Operador de atribuição:
 - FORTRAN, BASIC, PL/I, C, C++, Java --> '='
 - ALGOLs, Pascal, Modula-2, Ada --> ':='
- '=' ruim se possui sobrecarga com operador relacional de igualdade.
- Ex. PL/I: A=B=C; //igualdade ou atribuição?
- C, C++ não possui este problema

Instruções de atribuição

- Atribuição mais Complexa:

- Alvos múltiplos

Ex. em PL/I: A, B = 10

- Alvos condicionais

Ex. em C, C++ e Java:

`(first = true) ? total : subtotal = 0`

equivalente a: `if(first=true) total=0 else subtotal=0`

- Operador de atribuição composto - atribuição com operação

Ex. em C, C++ e Java:

`sum += next; /* sum = sum + next */`

Instruções de atribuição

- Atribuição mais Complexa (cont.):

- Operadores de atribuição unários

Ex. em C, C++ e Java:

```
a++;    /* a = a+1 */
```

- Operador aritmético binário '='

Ex. em C, C++ e Java:

```
a = b * (c = d * 2 + 1) + 1;
```

// equivalente a:

```
c = d * 2; a = b * (c + 1) + 1;
```

Instruções de atribuição

- Atribuição como uma Expressão
 - Em C, C++, Java, o comando de atribuição produz um resultado, portanto, pode ser utilizado como um operador em expressões.
 - Ex.:
`while ((ch = getchar()) != EOF) { ... }`
 - Desvantagem:
 - Outro tipo de efeito colateral em expressões.

Atribuição de Modo Misto

- Atribuição de Modo Misto:
 - Em FORTRAN, C/C++ qualquer valor numérico pode ser atribuído a qualquer variável escalar. Conversão necessária será efectuada.
Ex. C: `int x = 10;`
`double y = x; // conversão de int para double`
 - Em Pascal inteiros podem ser atribuídos a reais mas, reais não podem ser atribuídos a inteiros.

Atribuição de Modo Misto

- Em Java somente atribuição com conversão de alargamento é realizada.

Ex. em Java:

```
int x = 10;  
double y = x; // O.K.  
x = y; // inválido
```

- Em Ada não existe atribuição com coesão.