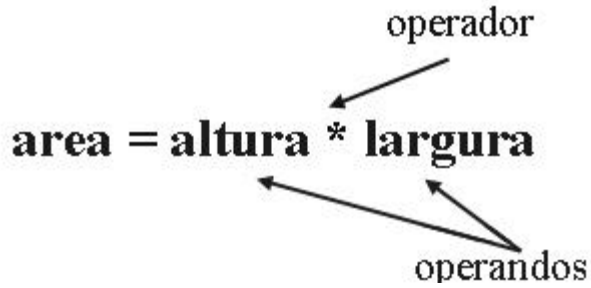
Expressões e Instruções de Atribuição (referência: Cap. 7 do livro de Sebesta)

Introdução

- Qual o resultado da expressão seguinte int a = 1, b = 2, c = 3;
  - -a*-b+-c;
- Expressões:
  - meio fundamental de especificar computações em uma linguagem de programação
- Para entender a avaliação de expressões é necessário estar familiarizado com a ordem de avaliação de operadores e de operandos
- A essência das linguagens de programação imperativas é dada pelo papel predominante das instruções de atribuição, cujo objetivo é mudar o valor de uma variável
- **Semântica das expressões** – significado de cada operação e operando numa expressão, tendo em atenção a ordem pela qual as operações são executadas e suas afetações.
- Operandos e operadores:




Expressões Aritméticas

- A avaliação de expressões aritméticas foi uma das principais motivações para o desenvolvimento da primeira linguagem de programação de alto nível;
- Expressões aritméticas consistem de operadores, operandos, parênteses e invocações de funções.
- Questões de projeto
  - Regras de precedência de operadores
  - Regras de associatividade dos operadores
  - Ordem de avaliação dos operandos
  - Efeitos colaterais da avaliação dos operandos
  - Sobrecarga de operadores
  - Mesclagem de modos (em termos de operadores) nas expressões

- Tipos de Operadores:
  - Unário - possui apenas um operando;
  - Binário - possui dois operandos;
  - Ternário - possui três operandos;

Precedência e Associatividade

- Como deve ser avaliada a seguinte expressão: a+b*c**d**e/f ?
  - Obs.: ** é a exponenciação
- Deveria ser
  - ((((a+b)*c)**d)**e)/f
  - a+(((b*c)**d)**(e/f))
  - a+((b*(c**(d**e)))/f)
- A última opção é usada por Fortran
- Ordem de avaliação dos operadores
  - a + b * c (a=3, b=4, c=5)
    - avaliada da esquerda ara a direita → resultado = 35
    - avaliada da direita para esquerda → resultado = 23
    - em vez de avaliar somente da esquerda para direita ou da direita para esquerda, os matemáticos desenvolveram o conceito de colocar operadores em uma hierarquia de prioridades de avaliação, levando ao conceito de precedência de operadores
  - As regras de precedência para avaliação de expressões definem a ordem na qual os operadores adjacentes de diferentes níveis de precedência são avaliados
  - Níveis de precedência comuns
    - parênteses
    - operadores unários
    - ** ( exponenciação, se a linguagem o suporta)
    - *, /
    - +, -

Regras de associatividade

- Regras de associatividade para avaliação de expressões definem em qual ordem operadores adjacentes de mesma precedência são avaliados
- Regras de associatividade comuns
  - Da esquerda para a direita, exceto **, o qual é da direita para a esquerda

- Em APL todos os operadores têm precedência iguais e são associativos da direita para a esquerda
- Regras de precedência e de associatividade podem ser substituídas pelo uso de parênteses

Expressões Condicionais

- Operador ternário → ?
  - disponível em C, C++ e Java
  - Usado para formar expressões condicionais
  - Sintaxe: expressão_1 ? expressão_2 : expressão_3
    - expressão_1 é interpretada como uma expressão boolena
    - se expressão_1 for verdadeira, o valor da expressão inteira será expressão_2
    - caso contrário será o valor de expressão_3
  - Um exemplo:
    - average = (count == 0)? 0 : sum/count
    - if-then-else equivalente

```
if (count == 0)
then average = 0
else average = sum /count
```

Ordem de avaliação dos operandos

- 1. Variáveis
  - São avaliadas buscando seus valores na memória
- 2. Constantes
  - Algumas vezes são avaliadas da mesma forma;
  - Outras vezes podem fazer parte da instrução em linguagem de máquina e não exigir uma busca na memória
- 3. Expressões entre parênteses
  - Todos os operadores devem ser avaliados antes que seu valor possa ser usado como um operando

Ordem de avaliação dos operandos:
Efeitos Colaterais

- 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 divido
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
```

- Duas soluções possíveis
  - 1. O projetista da linguagem poderia impedir que a avaliação da função afetasse o valor das expressões
    - Vantagem: funciona (rejeita os efeitos colaterais)
    - Desvantagem: inflexibilidade (Programadores querem a flexibilidade de parâmetros de entrada/saída.)
- 2. Declarar, na definição da linguagem, que os operandos devem ser avaliados em uma ordem particular
  - Desvantagem: limita alguns procedimentos de otimização dos compiladores

Sobrecarga de Operadores

- Usar um operador para mais do que um propósito
- Alguns são comuns
  - + para int e para float (adição de quaisquer operandos do tipo numérico)
  - JAVA: + para concatenação
- Alguns representam problemas em potencial
  - Perda da capacidade de detectar erros
    - Omissão de um operador
      - 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 podem ser evitados pela introdução de novos símbolos
    - **div** para divisão de inteiros no Pascal
      - (Ex. Pascal: result := soma div cont;)
- C++ e Ada permitem que usuários definam sobrecarga de operadores
- Problemas:
  - Usuários podem definir operadores sem significado lógico

    - A capacidade de leitura pode ser prejudicada
- A sobrecarga de operadores foi um dos recursos do C++ não copiado para o Java.

Conversões de Tipo

- Uma conversão de estreitamento transforma um valor para um tipo que não pode armazenar todos os valores do tipo original
    - float para int
- Uma conversão de alargamento transforma um valor para um tipo que pode incluir, pelo menos, aproximações de todos os valores do original
    - int para float
- As conversões de alargamento são quase sempre seguras, ao passo que as de estreitamento não

Coerção para Expressões

- Uma expressão de modo misto é aquela que possui operandos de tipos diferentes
- Uma coerção é uma conversão de tipo implícita iniciada pelo compilador
    - Desvantagem de coerção:
        - Diminui poder do compilador na detecção de erros
        - Exemplo:

```
int a, b, c;
float d;
a = b * d;
```

suponha que o segundo operando tivesse que ser c no lugar de d... uma vez que expressões de modo misto são legais, o compilador não detectaria isso como um erro e converteria inclusive b para float.

- Na maioria das linguagens, todos os tipos numéricos são convertidos (coerced) em expressões, usando coerção de alargamento
- Em Ada, praticamente, não é permitida coerção em expressões

Conversão de Tipo Explícita

- Chamada de casting em linguagens baseadas em C
- Exemplos
    - C: (int) angle

- Ada: Float (sum)
- Note que a sintaxe em Ada é similar a chamada de funções

Conversões de Tipo: Erros em Expressões

- Causas
  - Limitações inerentes à aritmética
    - Divisão por zero
- Limitações da aritmética computacional
  - Overflow e underflow (resultado não pode ser armazenado na célula de memória onde ele deve estar armazenado, dependendo se ele foi muito grande ou muito pequeno)

Expressões Relacionais e Booleanas

- 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 símbolo dos operadores relacionais variam entre as diversas linguagens.
    - Ex.: !=, /=, .NE., <>, #
- Expressões Booleanas
  - Operandos são booleanos e seu resultado é booleano
  - Exemplos de operadores

| FORTRAN 77 | FORTRAN 90 | C | Ada |
|---|---|---|---|
| .AND. | and | && | and |
| .OR. | or | \|\| | or |
| .NOT. | not | ! | not |

  - 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 correta e equivalente a: (a<b) < c
      - o resultado não é o esperado
      - Operador mais a esquerda é avaliado produzindo 0 ou 1
      - O resultado da avaliação é então comparado com o terceiro operando (i.e., c)

Avaliação Curto-Circuito

- Uma expressão que tem seu resultado determinado sem avaliar todos os operandos e/ou operadores
  - Exemplo: (13*a) * (b/13–1)
    - Se a é zero, não existe necessidade de avaliar (b/13-1)
  - Exemplo:
    - C: if(a>0 && b<50) //se a<0 --> b<50 não é avaliado.

- Problema se a avaliação não for realizada com curto-circuito

```
index = 0;
while (index < length) && (LIST[index] != value)
index++;
```

  - Quando index=length, LIST [index] causará um problema de indexação (assumindo que LIST tem length -1 elementos)

- Avaliação curto-circuito expõe o potencial problema de efeito colateral em expressões
  - Ex. C: (a > b) || (b++ / 3) // pode não ser executado
    - Incrementa b (b++) e depois avalia (a > b) ou avalia e depois incrementa

- C, C++ e Java
  - Usam avaliação curto-circuito para operadores booleanos comuns (&& e ||)
  - Mas os operadores booleanos bitwise (bit a bit) não são avaliados curto-circuito (& e |)
    - Operadores bitwise são usados quando precisamos fazer operações a nível de bits. Funcionam de forma semelhante aos operadores lógicos, exceto por trabalharem com representação binária de dados.
      - val1 & val2 : operador AND, compara 2 bits, se os dois tiverem o valor 1, retorna 1, senão retorna 0.
        - Ex: val1 & val2
        - 0011 : val1
        - 1011 : val2
        - 0011: Resultado

- Ada:
  - O programador pode especificar (curto-circuito é especificado com and then e or else)

Instruções de Atribuição

- Instruções de atribuição – mecanismo que permite modificar dinamicamente as vinculações de valores a variáveis.
- Sintaxe geral
  - <variável_alvo> <operador_de_atribuição> <expressão>
- Operadores de atribuição
  - = FORTRAN, BASIC, PL/I, C, C++, Java
  - := ALGOLs, Pascal, Ada
- = pode ser inadequado quando é sobrecarregado para o operador relacional de igualdade
  - Ex. PL/I: A=B=C; //igualdade ou atribuição?
    - Em PL/I define A para o valor booleano da expressão relacional B=C

Atribuição mais Complexa:

- Alvos múltiplos
  - Ex. em PL/I: A, B = 10
- Alvos condicionais
  - Ex. em C, C++ e Java:
    - (flag) ? total : subtotal = 0
    - equivalente a: if(flag=true) total=0 else subtotal=0
- Operador de atribuição composto - atribuição com operação
  - É um método abreviado de especificar uma forma de atribuição
  - Introduzido em ALGOL; adotado por C
  - Ex. em C, C++ e Java:
    - sum += next; /* equivalente a  sum = sum + next */
- Em Algol é possível fazer:
  - a := if b <> 0 then a/b else 0;
- Operadores unários
  - Linguagens baseadas em C combinam operações de incremento e de decremento com atribuição
  - Exemplos
    - sum = ++count (count incrementado, atribuído a sum)
    - sum = count++ (atribuído a sum, count incrementado)
    - count++ (count incrementado)
    - -count++ (count incrementado depois negado)

A Atribuição como uma Expressão

- Em C, C++ e Java, a instrução de atribuição produz um resultado, portanto, pode ser utilizado como um operador em expressões.
- Um exemplo:

  while ((ch = getchar())!= EOF){…}

  ch = getchar() é obtido; o resultado (atribuído a ch) é usado como um valor condicional na instrução while

Atribuição de Modo Misto

- Instruções de atribuição podem ser de modo misto, por exemplo

  int a, b;
  float c;
  c = a / b;
- Em Pascal, variáveis inteiras podem ser atribuídas a variáveis reais, mas variáveis reais não podem ser atribuídas a variáveis inteiras
- Em FORTRAN, C/C++ qualquer valor numérico pode ser atribuído a qualquer variável escalar. Conversão necessária será efetuada.

  Ex. C: int x = 10:
  double y = x; // conversão de int para double
- Em Java, apenas conversão de alargamento são permitidas

  Ex. em Java:
  int x = 10;
  double y = x; // O.K.
  x = y; // inválido
- Em Ada não existe atribuição com coesão.