Universidade Federal do Espírito Santo
Centro Norte do Espírito Santo - CEUNES

Linguagens de Programação

Parte 03

Prof. Francisco de Assis S. Santos, Dr.

2021/2.

Expressões e Comandos

Expressões: Conceituação

- **Uma expressão é uma frase do programa que necessita ser avaliada e produz como resultado um valor**
- Elementos
  - Operadores
  - Operandos
  - Resultado

Expressões: Conceituação

- **Podem ser**
  - Simples (apenas um operador)
  - Compostas (mais de um operador)
- Notação
  - Prefixada: operador antes dos operandos

    ```
    !b ; ++c ; sizeof(x)
    ```
  - Infixada: operador entre operandos

    ```
    a + b;
    ```
  - Posfixada: operador após operandos

    ```
    b++;
    ```

Operadores

- **`Aridade`**
  - unários (**`!a`**), binários (**`a+b`**), ternários (**`a?b:c`**)
  - eneários: aridade variável
    - número de parâmetros variável em funções C e C++

- **Origem**
  - Pré-existentes
    - normalmente unários e binários
  - Definidos pelo Programador
    - normalmente funções com qualquer aridade

Tipos de Expressões

- Literais
- Agregação
- Aritméticas
- Relacionais
- Booleanas ou lógicas
- Condicionais
- Chamadas de Funções
- Com Efeitos Colaterais
- Referenciamento
- Categóricas

Tipos de Expressões

- **Literais**

  2.72 99 0143 'c'

- **Agregação (em C)**

```
int c[ ] = {1, 2, 3};
struct data d = {1, 7, 1999};
char x [4] = {'a','b','c', '\0'};
int b[6] = {0};  // inicializa os 6 elementos (só funciona com zero)
char *y = "abc";
```

- **Agregação Estática e Dinâmica**

```
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}; }
```

Tipos de Expressões

- **Aritméticas**
  - operadores **+** e **–** unários, e **+**, **-**, *****, **/**, **%** binários

- **Relacionais (>, <, >=, <=, !=)**
  - usadas para comparar os valores de seus operandos, retornando verdadeiro ou falso

- **Booleanas ou lógicas (&&, ||, !)**
  - realizam as operações de negação, conjunção e disjunção da álgebra de Boole

Tipos de Expressões: Aritmética (Alerta!)

- **Aritméticas**
  - Exemplo:
    Em C, int x, y; x=1;
    x++;
    ++x; isoladamente produzem o mesmo efeito, mas em uma atribuição?
    - y = ++x (O que acontece?)
    - y = x++ (O que acontece?)

Tipos de Expressões

- **Condicionais**
  - Em Pascal

```
val c = if a > b  then a - 3 else b + 5
val k = case i of
  1 => if j > 5 then j - 8 else j + 5
| 2 => 2*j
| 3 => 3*j
| _ => j
```

  - Em JAVA/C/C++

```
max = x > y ? x : y;
par  = z % 2 == 0 ? true : false
```

  - Obs.: algumas LPs (tal como ADA) não oferecem expressões condicionais – forçam o uso de comandos condicionais

```
if x > y then max := x; else max := y; end if;
```

Atividade em Sala: Análise do Código Correntista

Tipos de Expressões

- **Chamadas de Funções**
  - Operador => nome da função
  - Operandos => parâmetros
  - Resultado => retorno da função

Tipos de Expressões

- **Chamadas de Funções**
  - Operador => nome da função
  - Operandos => parâmetros
  - Resultado => retorno da função

Tipos de Expressões: Chamadas de Funções

- Algumas Assinaturas de Operadores em JAVA

Operador	Assinatura da Função
!	**[boolean → boolean]**
&&	**[boolean x boolean → boolean]**
*****	**[int x int → int]** **[float x float → float]**

Tipos de Expressões

- Com Efeitos Colaterais
  (o objetivo principal de uma expressão é retornar um valor …)

  ```
  x = 3.2 * ++c;
  ```

  - Podem gerar indeterminismo

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

  - Funções possibilitam a ocorrência de efeitos colaterais

    ```
    getc();
    ```
    // além de retornar um caractere, também movimenta o ponteiro interno de leitura

  - Expressões cujo único objetivo é produzir efeitos colaterais

    ```
    free(p);
    ```
    // não tem retorno, apenas desaloca a memória associada a p

Tipos de Expressões

- **Referenciamento**
  - Usadas para acessar o conteúdo ou retornar referência para variáveis ou constantes

// os elementos da esquerda fazem referência ao elemento

// já os da direita, retornam o conteúdo

```
*q = *q + 3;
const float pi = 3.1416;
int raio = 3;
float perimetro = 2*pi*raio;
p[i] = p[i + 1];
r.ano = r.ano + 1;
s->dia = s->dia +1;

t = &m;
```

Tipos de Expressões: Referenciamento

Operador	Significado
[ ]	Acesso a valor de elemento de vetor
*	Acesso a valor de variável ou constante apontada por ponteiro
.	Acesso a valor de elemento de estrutura
->	Acesso a valor de elemento de estrutura apontada por ponteiro
&	(em C++) Retorno de referência a qualquer tipo de variável ou constante

Tipos de Expressões: Referenciamento

- **Exemplo:**

```
int a=2, b=3,c;
int *p, *q;  // p e q são ponteiros para um inteiro
p = &a;  // p aponta para a
q = &b;
c = *p + *q;
printf("\n Resultado : %d \n", c);
```

Tipos de Expressões

- **Categóricas**
  - **Realizam operações sobre tipos de dados**
  - **Tamanho do Tipo**
    - `float *p = (float*) malloc(10 * sizeof (float));`
    - `int c [] = {1, 2, 3, 4, 5};`
    - `for (i = 0; i < sizeof c / sizeof *c; i++)`
      `c[i]++;`
  - **Conversão de Tipo**

```
float f;
int num = 9, den = 5;
f = (float)num/den;
```

Tipos de Expressões: Categórica

- **`Identificação de Tipo`**

(em Java)

```
Profissao p;
    ...
    p = new Engenheiro ( );
    ...
if (p instanceof Medico)
  System.out.println (“Registre-se no CRM”);
if (p instanceof Engenheiro)
  System.out.println (“Registre-se no CREA”);
```

Avaliação de Expressões Compostas

- **Associatividade de Operadores**
  - Operadores de mesma Precedência
  - Normalmente da esquerda para a direita

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

  - Podem existir exceções a essa regra (dir p/ esq)

```
x = **p;
if (!!x) y = 3;
a = b = c;
```

  - APL não tem precedência e sempre associa da direita para a esquerda

```
X = Y ÷ W – Z
```

(fora do padrão!)

Avaliação de Expressões Compostas

- **`Associatividade de Operadores`**
  - Compiladores podem otimizar, mas isso pode causar problemas

```
x = f() + g() + h(); // quem é executado primeiro?
```

- **Precedência de Operandos**
  - Não determinismo em expressões

```
a[i] = i++; // Mingw/GCC (teste feito no Dev C)
            // e Java consideram i antigo
```

  - JAVA resolve adotando precedência de operandos da esquerda para direita
  - Garante portabilidade, mas pode comprometer a eficiência

Avaliação de Expressões Compostas

- **Curto Circuito**

Avaliação de curto-circuito ou avaliação mínima o especifica a semântica de alguns operadores booleanos em algumas linguagens de programação na qual o segundo argumento é apenas executado ou avaliado se o primeiro argumento não for suficiente para determinar o valor da expressão: quando o primeiro argumento de uma função AND é avaliado como falso, o valor global deve ser falso e quando o primeiro argumento da função OR for avaliado como verdadeiro, o valor global deve ser verdadeiro. Em algumas linguagens de programação (Lisp), os operadores booleanos usuais são de curto-circuito. Em outras (Java, Ada), os operadores booleanos padrões e de curto-circuito estão disponíveis.

  - O operador de curto-circuito x Sand y é equivalente à expressão condicional if x then y else false. x Sor y é equivalente à if x then true else y.

Avaliação de Expressões Compostas

- **Curto Circuito**

```
int n = 10; i = 0;
int[] a = new int [n];
while (i < n && a[i] != v) i++;
```

  - **C/C++/JAVA possuem operador lógico sem curto circuito: & e |**
  - Curto Circuito com efeito colateral: reduz legibilidade

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

Avaliação de Expressões Compostas
Qual será o retorno?

```
public class Exemplo {                                  //   em Java:
    public static void main(String[] args) {
        int a = 2, b = 3, c = 4;
        if(a < b || funcao1() == funcao2()) // ??
            System.out.println("ok 1");
        if(a < b | funcao1() == funcao2()) // ??
            System.out.println("ok 2");
    }
    static int funcao1() {
        System.out.println("entrou em funcao 1");      return 5;
    }
    static int funcao2() {
        System.out.println("entrou em funcao 2");      return 5;
    }
}
```

Avaliação de Expressões Compostas
Vejamos a análise!

```
public class Exemplo {                                    //   em Java:
    public static void main(String[] args) {
        int a = 2, b = 3, c = 4;
        if(a < b || funcao1() == funcao2()) // com curto circuíto
            System.out.println("ok 1");
        if(a < b | funcao1() == funcao2()) // sem curto circuíto
            System.out.println("ok 2");
    }
    static int funcao1() {
        System.out.println("entrou em funcao 1");      return 5;
    }
    static int funcao2() {
        System.out.println("entrou em funcao 2");      return 5;
    }
}
```

Exibe:

```
ok 1
entrou em
    funcao 1
entrou em
    funcao 2
ok 2
```

Avaliação de Código em C para Polinômio $f(x_o)$

Considerar o código impresso entregue em aula!

Comandos

Tipos de Comandos

- Atribuição
- Sequenciais
- Colaterais
- Condicionais
- Iterativos
- Chamadas de Procedimentos
- Desvios Incondicionais

Tipos de Comandos: Atribuição

- **= versus :=**

```
i := i + 1;  (em Pascal)
i = i + 1;   (em C)
i := !i + 1; (em ML - obriga o uso do
                operador ! para dereferenciar
                a variável i e não confundir
                com o primeiro i)
```

- **Simples**

```
a = b + 3 * c;
```

- **Múltipla (em C)**

```
a = b = 0;
```

- **Múltipla (em Python)**

```
a,b = 2,3;
  a,b = b,a;  // troca sem o uso de var auxiliar
```

Tipos de Comandos: Atribuição

- **Atribuição (continuação...)**
  - **Condicional**

```
(if a < b then a else b) := 2;   // em ML
```

  - **Composta**

```
a *= 3;
a += 3;
```

  - **Unária**

```
++a;
a++;
--a;
```

  - **Expressão**

```
while (( ch = getchar ( ) ) != EOF ) { printf("%c", ch); }
```

Tipos de Comandos: Sequenciais

- **Blocos**

```
{
        n = 1;
        n += 3;
        if (n < 5) {
           n = 10;
        m = n * 2;
    }
}
```

Tipos de Comandos: Colaterais

- Vírgula (linguagem C/C++ consideram um operador):

```
c = (2,3);  // c recebe 3
c = 2,3;    // c recebe 2, pois ele possui baixa precedência

b = (a = 3, a = a + 1); // b recebe 4
b = a = 3, a = a + 1; // Qual valor de b? Qual valor de a?
```

Útil no comando for:

```
for(i=0, n=5; i < 10; i++, n++)
```

Obs.: Java aceita, mas somente em expressões no comando for.

Tipos de Comandos: Colaterais

- **Em ML:**

```
val altura = 2
    and largura = 3
    and comprimento = 5
    and volume = altura * largura * comprimento
```

Tipos de Comandos: Condicionais

- Seleção de Caminho Condicionado

```
if (x < 0) { x = y + 2; x++; }
```

- Seleção de Caminho Duplo

```
if (x < 0) { x = y + 2; x++; } else { x = y; x--;
  }
```

- **Problema com Marcadores**

```
if ( x == 7 )
    if ( y == 11) {
        z = 13;
        w = 2;
    }
else z = 17;        // erro de identação
```

Tipos de Comandos: Condicionais

- **Seleção de Caminho Condicionado**

```
if (x < 0) { x = y + 2; x++; }
```

- **Seleção de Caminho Duplo**

```
if (x < 0) { x = y + 2; x++; } else { x = y; x--;
  }
```

  - **Problema com Marcadores**

```
if ( x == 7 )
    if ( y == 11) {
        z = 13;
        w = 2;
    }
else z = 17;        // erro de identação
```

Tipos de Comandos: Condicionais

- **Seleção de Caminho Duplo (continuação...)**
  - ADA requer marcador de final de comando: maior legibilidade

```
if x > 0 then
    if y > 0 then
        z := 0;
    end if;
else
    z := 1;
end if;
```

Tipos de Comandos: Condicionais

- **Seleção de Caminhos Múltiplos**

```
switch (nota) {
case 10:
case 9: printf (“Muito Bom!!!”);
        break;
case 8:
case 7: printf (“Bom!”);
        break;
case 6:
case 5: printf (“Passou…”);
        break;
default: printf (“Estudar mais!”);
}
```

Tipos de Comandos: Condicionais

- **Caminhos Múltiplos com ifs Aninhados**

```
if (rendaMes < 1000)
   iR = 0;
else if (rendaMes < 2000)
   iR = 0.15 * (2000 - rendaMes);
else
   iR = 0.275 * (rendaMes - 2000) +
       0.15 * (2000 – rendaMes);
```

  - Python tem `elif`
  - Ruby, Perl, Modula-2, ADA e FORTRAN-90 têm `elsif`