Expressões com inteiros

Aula 5

Diego Padilha Rubert

Faculdade de Computação
Universidade Federal de Mato Grosso do Sul

Algoritmos e Programação

Conteúdo da aula

Expressões aritméticas

- **expressão aritmética** é qualquer sequência de símbolos formada exclusivamente por constantes numéricas, variáveis numéricas, operadores aritméticos e parênteses
  - **constante numérica do tipo inteiro** é qualquer número inteiro descrito em nosso programa
  - **variável com valor inteiro** é aquela que para a qual foi atribuído um valor inteiro
  - **operadores aritméticos** são divididos em duas classes: operadores aritméticos unários e operadores aritméticos binários

Expressões aritméticas

- **expressão aritmética** é qualquer sequência de símbolos formada exclusivamente por constantes numéricas, variáveis numéricas, operadores aritméticos e parênteses
  - **constante numérica do tipo inteiro** é qualquer número inteiro descrito em nosso programa
  - **variável com valor inteiro** é aquela que para a qual foi atribuído um valor inteiro
  - **operadores aritméticos** são divididos em duas classes: operadores aritméticos unários e operadores aritméticos binários

Expressões aritméticas

- **expressão aritmética** é qualquer sequência de símbolos formada exclusivamente por constantes numéricas, variáveis numéricas, operadores aritméticos e parênteses
  - **constante numérica do tipo inteiro** é qualquer número inteiro descrito em nosso programa
  - **variável com valor inteiro** é aquela que para a qual foi atribuído um valor inteiro
  - **operadores aritméticos** são divididos em duas classes: operadores aritméticos unários e operadores aritméticos binários

Expressões aritméticas

- **expressão aritmética** é qualquer sequência de símbolos formada exclusivamente por constantes numéricas, variáveis numéricas, operadores aritméticos e parênteses
  - **constante numérica do tipo inteiro** é qualquer número inteiro descrito em nosso programa
  - **variável com valor inteiro** é aquela que para a qual foi atribuído um valor inteiro
  - **operadores aritméticos** são divididos em duas classes: operadores aritméticos unários e operadores aritméticos binários

Expressões aritméticas

- **operador aritmético unário** é um operador que age sobre um único número inteiro e devolve um resultado
  - `-` : troca o sinal da expressão aritmética que o sucede
- **operador aritmético binário** é aquele que realiza uma operação básica sobre dois números inteiros
  - `+` : adição
  - `-` : subtração
  - `*` : multiplicação
  - `/` : quociente da divisão
  - `//` : quociente inteiro da divisão
  - `%` : resto da divisão
  - `**` : exponenciação

Expressões aritméticas

- **operador aritmético unário** é um operador que age sobre um único número inteiro e devolve um resultado
  - `-` : troca o sinal da expressão aritmética que o sucede
- **operador aritmético binário** é aquele que realiza uma operação básica sobre dois números inteiros
  - `+` : adição
  - `-` : subtração
  - `*` : multiplicação
  - `/` : quociente da divisão
  - `//` : quociente inteiro da divisão
  - `%` : resto da divisão
  - `**` : exponenciação

Expressões aritméticas

- **operador aritmético unário** é um operador que age sobre um único número inteiro e devolve um resultado
  - `-` : troca o sinal da expressão aritmética que o sucede
- **operador aritmético binário** é aquele que realiza uma operação básica sobre dois números inteiros
  - `+` : adição
  - `-` : subtração
  - `*` : multiplicação
  - `/` : quociente da divisão
  - `//` : quociente inteiro da divisão
  - `%` : resto da divisão
  - `**` : exponenciação

Expressões aritméticas

Precedência dos operadores:

Operadores	Tipo	Descrição	Precedência
`**`	binário	exponenciação	1 (máxima)
`-`	unários	troca de sinal	2
`*  /  %`	binários	produto, quociente e resto da divisão	3
`+  -`	binários	adição e subtração	4 (mínima)

Expressões aritméticas

Exemplos:

```
x = 1
y = 2
a = 5
soma = 100
parcela = 134

        2 * x * x + 5 * -x - 4
         soma + parcela % 3
      4 * 1002 - 4412 % 11 * -2 + a
(((204 / (3 + x)) * y) - ((y % x) + soma))
```

Expressões relacionais

- **expressão relacional** ou **relação** é uma comparação entre dois valores do mesmo tipo primitivo de dados
- valores são representados na relação através de constantes, variáveis ou expressões
- com valores inteiros a definição de relação pode ser escrita como uma comparação entre dois valores, representados por constantes numéricas, variáveis contendo números ou expressões aritméticas

Expressões relacionais

- **expressão relacional** ou **relação** é uma comparação entre dois valores do mesmo tipo primitivo de dados
- valores são representados na relação através de constantes, variáveis ou expressões
- com valores inteiros a definição de relação pode ser escrita como uma comparação entre dois valores, representados por constantes numéricas, variáveis contendo números ou expressões aritméticas

Expressões relacionais

- **expressão relacional** ou **relação** é uma comparação entre dois valores do mesmo tipo primitivo de dados
- valores são representados na relação através de constantes, variáveis ou expressões
- com valores inteiros a definição de relação pode ser escrita como uma comparação entre dois valores, representados por constantes numéricas, variáveis contendo números ou expressões aritméticas

Expressões relacionais

Operadores relacionais:

Operador	Descrição
==	igual a
!=	diferente de
<	menor que
>	maior que
<=	menor que ou igual a
>=	maior que ou igual a

O resultado da avaliação de uma expressão relacional é sempre um valor lógico, isto é, *verdadeiro* (*True*) ou *falso* (*False*)

Expressões relacionais

Operadores relacionais:

Operador	Descrição
==	igual a
!=	diferente de
<	menor que
>	maior que
<=	menor que ou igual a
>=	maior que ou igual a

O resultado da avaliação de uma expressão relacional é sempre um valor lógico, isto é, *verdadeiro* (*True*) ou *falso* (*False*)

Expressões relacionais

Exemplos:

```
a = 2
b = 3
c = 4

   a == 2
  a > b + c
b + c <= 5 - a
   b != 3
 10 < a <= 20
```

Expressões relacionais

Importante!

- valores sempre têm um tipo definido, mas as variáveis não!
- *True* e *False* são os dois únicos valores possíveis do tipo *booleano*
- o resultado da avaliação de uma expressão relacional ou lógica é, na verdade, um valor do tipo booleano
- se este valor é *True*, teremos por exemplo `if True:`, e o fluxo do programa irá executar as instruções dentro do bloco do `if`
- caso contrário, isto é, se este valor é *False*, então o fluxo do programa será desviado (para o próximo `elif`, para um `else` ou para fora do `if`)

Expressões relacionais

Importante!

- valores sempre têm um tipo definido, mas as variáveis não!
- *True* e *False* são os dois únicos valores possíveis do tipo *booleano*
- o resultado da avaliação de uma expressão relacional ou lógica é, na verdade, um valor do tipo booleano
- se este valor é *True*, teremos por exemplo `if True:`, e o fluxo do programa irá executar as instruções dentro do bloco do `if`
- caso contrário, isto é, se este valor é *False*, então o fluxo do programa será desviado (para o próximo `elif`, para um `else` ou para fora do `if`)

Expressões relacionais

Importante!

- valores sempre têm um tipo definido, mas as variáveis não!
- *True* e *False* são os dois únicos valores possíveis do tipo *booleano*
- o resultado da avaliação de uma expressão relacional ou lógica é, na verdade, um valor do tipo booleano
- se este valor é *True*, teremos por exemplo `if True:`, e o fluxo do programa irá executar as instruções dentro do bloco do `if`
- caso contrário, isto é, se este valor é *False*, então o fluxo do programa será desviado (para o próximo `elif`, para um `else` ou para fora do `if`)

Expressões relacionais

Importante!

- valores sempre têm um tipo definido, mas as variáveis não!
- *True* e *False* são os dois únicos valores possíveis do tipo *booleano*
- o resultado da avaliação de uma expressão relacional ou lógica é, na verdade, um valor do tipo booleano
- se este valor é *True*, teremos por exemplo `if True:`, e o fluxo do programa irá executar as instruções dentro do bloco do `if`
- caso contrário, isto é, se este valor é *False*, então o fluxo do programa será desviado (para o próximo `elif`, para um `else` ou para fora do `if`)

Expressões relacionais

Importante!

- valores sempre têm um tipo definido, mas as variáveis não!
- *True* e *False* são os dois únicos valores possíveis do tipo *booleano*
- o resultado da avaliação de uma expressão relacional ou lógica é, na verdade, um valor do tipo booleano
- se este valor é *True*, teremos por exemplo `if True:`, e o fluxo do programa irá executar as instruções dentro do bloco do `if`
- caso contrário, isto é, se este valor é *False*, então o fluxo do programa será desviado (para o próximo `elif`, para um `else` ou para fora do `if`)

Expressões relacionais

Importante!

- adicionalmente, ao considerarmos valores de tipos não booleanos, os seguintes valores são equivalentes a *False*: zero, *None*, ou objeto vazio (string, lista, tupla ou dicionário vazio)
- todos os outros valores de tipos não booleanos são equivalentes a *True*

Expressões relacionais

Importante!

- adicionalmente, ao considerarmos valores de tipos não booleanos, os seguintes valores são equivalentes a *False*: zero, *None*, ou objeto vazio (string, lista, tupla ou dicionário vazio)
- todos os outros valores de tipos não booleanos são equivalentes a *True*

Expressões lógicas

- **proposição** é qualquer sentença que pode ser valorada com o valor verdadeiro ou falso
- **expressão condicional** ou **lógica**, ou ainda **booleana**, é formada por uma ou mais proposições
- relacionamos as proposições através de **operadores lógicos**

Expressões lógicas

- **proposição** é qualquer sentença que pode ser valorada com o valor verdadeiro ou falso
- **expressão condicional** ou **lógica**, ou ainda **booleana**, é formada por uma ou mais proposições
- relacionamos as proposições através de **operadores lógicos**

Expressões lógicas

- **proposição** é qualquer sentença que pode ser valorada com o valor verdadeiro ou falso
- **expressão condicional** ou **lógica**, ou ainda **booleana**, é formada por uma ou mais proposições
- relacionamos as proposições através de **operadores lógicos**

Expressões lógicas

Operadores lógicos da linguagem Python:

Operador	Descrição
`and`	conjunção
`or`	disjunção
`not`	negação

Expressões lógicas

- duas proposições `p` e `q` podem ser combinadas pelo conectivo `and` para formar uma única proposição denominada **conjunção** das proposições originais: `p and q` e lemos "`p` e `q`"
- o resultado da avaliação da conjunção de duas proposições é verdadeiro se e somente se ambas as proposições têm valor verdadeiro

p	q	p and q
1	1	1
1	0	0
0	1	0
0	0	0

Expressões lógicas

- duas proposições `p` e `q` podem ser combinadas pelo conectivo `and` para formar uma única proposição denominada **conjunção** das proposições originais: `p and q` e lemos "`p` e `q`"
- o resultado da avaliação da conjunção de duas proposições é verdadeiro se e somente se ambas as proposições têm valor verdadeiro

p	q	p and q
1	1	1
1	0	0
0	1	0
0	0	0

Expressões lógicas

- duas proposições `p` e `q` podem ser combinadas pelo conectivo `and` para formar uma única proposição denominada **conjunção** das proposições originais: `p and q` e lemos "`p` e `q`"
- o resultado da avaliação da conjunção de duas proposições é verdadeiro se e somente se ambas as proposições têm valor verdadeiro

p	q	p and q
1	1	1
1	0	0
0	1	0
0	0	0

Expressões lógicas

- duas proposições `p` e `q` podem ser combinadas pelo conectivo `or` para formar uma única proposição denominada **disjunção** das proposições originais: `p or q` e lemos "`p` ou `q`"
- o resultado da avaliação da disjunção de duas proposições é verdadeiro quando ao menos uma das proposições tem valor verdadeiro

p	q	p or q
1	1	1
1	0	1
0	1	1
0	0	0

Expressões lógicas

- duas proposições `p` e `q` podem ser combinadas pelo conectivo `or` para formar uma única proposição denominada **disjunção** das proposições originais: `p or q` e lemos "`p` ou `q`"
- o resultado da avaliação da disjunção de duas proposições é verdadeiro quando ao menos uma das proposições tem valor verdadeiro

p	q	p or q
1	1	1
1	0	1
0	1	1
0	0	0

Expressões lógicas

- duas proposições `p` e `q` podem ser combinadas pelo conectivo `or` para formar uma única proposição denominada **disjunção** das proposições originais: `p or q` e lemos "`p` ou `q`"
- o resultado da avaliação da disjunção de duas proposições é verdadeiro quando ao menos uma das proposições tem valor verdadeiro

p	q	p or q
1	1	1
1	0	1
0	1	1
0	0	0

Expressões lógicas

- dada uma proposição `p`, uma outra proposição, chamada **negação** de `p`, pode ser obtida através da inserção da palavra `not` antes da proposição: `not p` e lemos "não `p`"
- se a proposição `p` tem valor verdadeiro, então `not p` tem valor falso e se `p` tem valor falso, então a proposição `not p` tem valor verdadeiro

p	not p
1	0
0	1

Expressões lógicas

- dada uma proposição `p`, uma outra proposição, chamada **negação** de `p`, pode ser obtida através da inserção da palavra `not` antes da proposição: `not p` e lemos "não `p`"
- se a proposição `p` tem valor verdadeiro, então `not p` tem valor falso e se `p` tem valor falso, então a proposição `not p` tem valor verdadeiro

p	not p
1	0
0	1

Expressões lógicas

- dada uma proposição `p`, uma outra proposição, chamada **negação** de `p`, pode ser obtida através da inserção da palavra `not` antes da proposição: `not p` e lemos "não `p`"
- se a proposição `p` tem valor verdadeiro, então `not p` tem valor falso e se `p` tem valor falso, então a proposição `not p` tem valor verdadeiro

`p`	`not p`
1	0
0	1

Expressões lógicas

Exemplos de expressões lógicas:

```
a = 2
b = 3
c = 4
x = 1

                5
                a
                not x
      a == 2 and a < b + c
    b + c >= 5 - a or b != 3
a + b > c or 2 * x == b and 4 < x
```

Expressões lógicas

- avaliamos as expressões lógicas em uma ordem: primeiro avaliamos as expressões aritméticas, depois as expressões relacionais e, por fim, as expressões lógicas em si

```
a == 2 and a + x > b + c
  2 == 2 and 3 > 3 + 4
    2 == 2 and 3 > 7
      True and False
          False
```

Expressões lógicas

- avaliamos as expressões lógicas em uma ordem: primeiro avaliamos as expressões aritméticas, depois as expressões relacionais e, por fim, as expressões lógicas em si

```
a == 2 and a + x > b + c
  2 == 2 and 3 > 3 + 4
    2 == 2 and 3 > 7
      True and False
          False
```

Expressões lógicas

- avaliamos as expressões lógicas em uma ordem: primeiro avaliamos as expressões aritméticas, depois as expressões relacionais e, por fim, as expressões lógicas em si

```
a == 2 and a + x > b + c
  2 == 2 and 3 > 3 + 4
    2 == 2 and 3 > 7
      True and False
          False
```

Expressões lógicas

- avaliamos as expressões lógicas em uma ordem: primeiro avaliamos as expressões aritméticas, depois as expressões relacionais e, por fim, as expressões lógicas em si

```
a == 2 and a + x > b + c
  2 == 2 and 3 > 3 + 4
    2 == 2 and 3 > 7
      True and False
          False
```

Expressões lógicas

- avaliamos as expressões lógicas em uma ordem: primeiro avaliamos as expressões aritméticas, depois as expressões relacionais e, por fim, as expressões lógicas em si

```
a == 2 and a + x > b + c
  2 == 2 and 3 > 3 + 4
    2 == 2 and 3 > 7
      True and False
          False
```

Expressões lógicas

- avaliamos as expressões lógicas em uma ordem: primeiro avaliamos as expressões aritméticas, depois as expressões relacionais e, por fim, as expressões lógicas em si

```
a == 2 and a + x > b + c
  2 == 2 and 3 > 3 + 4
    2 == 2 and 3 > 7
      True and False
          False
```

Expressões lógicas

Operador	Tipo	Precedência
`**`	unário	1 (máxima)
`-`	unários	2
`* / %`	binários	3
`+ -`	binários	4
`== != >= <= > <`	binários	5
`not`	unário	6
`and`	binário	7
`or`	binário	8 (mínima)

Expressões lógicas

Exemplo:

```
x + c >= a + b or 2 * x + x < b and a > b + x
1 + 4 >= 2 + 3 or 2 * 1 + 1 < 3 and 2 > 3 + 1
    5 >= 5     or       3 < 3 and 2 > 4
    True       or     False and False
    True       or         False
               True
```

Expressões lógicas

Exemplo:

```
x + c >= a + b or 2 * x + x < b and a > b + x
1 + 4 >= 2 + 3 or 2 * 1 + 1 < 3 and 2 > 3 + 1
    5 >= 5     or         3 < 3 and 2 > 4
    True       or         False and False
    True       or             False
               True
```

Expressões lógicas

Exemplo:

```
x + c >= a + b or 2 * x + x < b and a > b + x
1 + 4 >= 2 + 3 or 2 * 1 + 1 < 3 and 2 > 3 + 1
    5 >= 5     or         3 < 3 and 2 > 4
    True       or         False and False
    True       or             False
               True
```

Expressões lógicas

Exemplo:

```
x + c >= a + b or 2 * x + x < b and a > b + x
1 + 4 >= 2 + 3 or 2 * 1 + 1 < 3 and 2 > 3 + 1
    5 >= 5     or         3 < 3 and 2 > 4
    True       or         False and False
    True       or             False
               True
```

Expressões lógicas

Exemplo:

```
x + c >= a + b or 2 * x + x < b and a > b + x
1 + 4 >= 2 + 3 or 2 * 1 + 1 < 3 and 2 > 3 + 1
    5 >= 5     or         3 < 3 and 2 > 4
    True       or         False and False
    True       or              False
                  True
```

Expressões lógicas

Exemplo:

```
x + c >= a + b or 2 * x + x < b and a > b + x
1 + 4 >= 2 + 3 or 2 * 1 + 1 < 3 and 2 > 3 + 1
    5 >= 5     or         3 < 3 and 2 > 4
    True       or       False and False
    True       or             False
               True
```

Exercícios

1. Pesquise sobre os operadores aritméticos *in-place* (auto-atribuição) `+=`, `-=`, `*=`, `/=`, `//=`, `%=` e `**=`
2. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo par e positivo. Caso contrário, escreva "NÃO".
3. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo ímpar, múltiplo de 3 e múltiplo de 7. Caso contrário, escreva "NÃO".
4. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo ímpar e positivo, OU ao mesmo tempo par e negativo. Caso contrário, escreva "NÃO".

Exercícios

1. Pesquise sobre os operadores aritméticos *in-place* (auto-atribuição) `+=`, `-=`, `*=`, `/=`, `//=`, `%=` e `**=`
2. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo par e positivo. Caso contrário, escreva "NÃO".
3. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo ímpar, múltiplo de 3 e múltiplo de 7. Caso contrário, escreva "NÃO".
4. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo ímpar e positivo, OU ao mesmo tempo par e negativo. Caso contrário, escreva "NÃO".

Exercícios

1. Pesquise sobre os operadores aritméticos *in-place* (auto-atribuição) `+=`, `-=`, `*=`, `/=`, `//=`, `%=` e `**=`
2. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo par e positivo. Caso contrário, escreva "NÃO".
3. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo ímpar, múltiplo de 3 e múltiplo de 7. Caso contrário, escreva "NÃO".
4. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo ímpar e positivo, OU ao mesmo tempo par e negativo. Caso contrário, escreva "NÃO".

Exercícios

1. Pesquise sobre os operadores aritméticos *in-place* (auto-atribuição) `+=`, `-=`, `*=`, `/=`, `//=`, `%=` e `**=`
2. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo par e positivo. Caso contrário, escreva "NÃO".
3. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo ímpar, múltiplo de 3 e múltiplo de 7. Caso contrário, escreva "NÃO".
4. Dado um número inteiro n, escreva "SIM" se ele é ao mesmo tempo ímpar e positivo, OU ao mesmo tempo par e negativo. Caso contrário, escreva "NÃO".

Exercícios

5. Dados dois números inteiros a e b, escreva “SIM” se a é ao mesmo tempo par, positivo e múltiplo de b, OU se b é ao mesmo tempo ímpar, múltiplo de a e diferente de a. Caso contrário, escreva “NÃO”.
6. Dados dois números inteiros a e b, escreva “SIM” se a é ao mesmo tempo ímpar, não positivo, e menor que b, OU se a é menor que -3, OU se o dobro de a é múltiplo de b, OU se b é ao mesmo tempo ímpar, múltiplo de a e diferente de a, OU se a diferença entre ambos é maior que 10. Caso contrário, escreva “NÃO”.

Exercícios

5. Dados dois números inteiros a e b, escreva “SIM” se a é ao mesmo tempo par, positivo e múltiplo de b, OU se b é ao mesmo tempo ímpar, múltiplo de a e diferente de a. Caso contrário, escreva “NÃO”.
6. Dados dois números inteiros a e b, escreva “SIM” se a é ao mesmo tempo ímpar, não positivo, e menor que b, OU se a é menor que -3, OU se o dobro de a é múltiplo de b, OU se b é ao mesmo tempo ímpar, múltiplo de a e diferente de a, OU se a diferença entre ambos é maior que 10. Caso contrário, escreva “NÃO”.