Expressões e definições

- Constantes
- Aplicação de função
- Nomeação
- Definindo variáveis e funções

expressões

- Por definição, uma expressão é uma entidade computacional que tem um valor
- Uma expressão em pode ser:
  - **Uma constante**
  - **Uma combinação**
  - **Um nome**

constantes

- Números (inteiros e reais):
  - **0 1.5 5. 3.14 -5 -5.3E-7**
- Strings:
  - **"Fulano de Tal"**
- Valores booleanos:
  - **True False**
  - **Ex.: (3 < 4) resulta em True**

constantes

- Quando se tecla uma *expressão* em um terminal de computador,
- o interpretador responde apresentando o resultado da *avaliação* da expressão

constantes

- Sempre que é fornecida uma constante, o interpretador devolve a constante como resultado da avaliação

- O interpretador mostra a forma canônica (representação externa) da representação interna da constante

constantes

- Em Haskell:

Prelude> 486

- A resposta será

486

constantes

- Valores lógicos em Haskell:

Prelude>True

True

Prelude>False

False

constantes

- Cadeias de caracteres (strings) em Haskell:

Prelude>“bom dia”

“bom dia”

constantes

descrição			exemplo
literais numéricos	inteiros	em decimal	`8743`
		em octal	`0o7464`
			`0O103`
		em hexadecimal	`0x5A0FF`
			`0xE0F2`
	fracionários	em decimal	`140.58`
			`8.04e7`
			`0.347E+12`
			`5.47E-12`
			`47e22`
literais caracter			`'H'`
			`'\n'`
			`'\x65'`
literais string			`"bom dia"`
			`"ouro preto\nmg"`
construtores booleanos			`False`
			`True`

Aplicação de função

- Aplicação de função é uma das formas de expressões mais comuns na programação funcional,
- uma vez que os programas são organizados em funções.
- f(x) = x + 4 definição da função
- f(2) = 6 aplicação da função

Aplicação de função

- uma aplicação de função em **notação prefixa** consiste em escrever a função seguida dos argumentos, se necessário separados por caracteres brancos (espaços, tabuladores, mudança de linha, etc.).
- Prelude> sqrt 25
  - 5.0
- Prelude> cos 0
  - 1.0

Aplicação de função

- Mais exemplos:
- Prelude> tan pi
- -1.2246467991473532e-16
- Prelude> exp 1
- 2.718281828459045
- Prelude> logBase 3 81
- 4.0

Aplicação de função

- Observe que, diferentemente de várias outras linguagens de programação,
- **os argumentos não são escritos entre parênteses e nem separados por vírgula.**
- Prelude> logBase 3 81
- 4.0

Aplicação de função

- Parênteses podem ser usados para agrupar subexpressões (geram um valor).
- Por exemplo:
- Prelude> sqrt (logBase 3 81)
- 2.0
- Prelude> logBase (sqrt 9) 81
- 4.0

Aplicação de função

- Aplicações de função também podem ser escritas em **notação infixa**, onde a função é escrita entre os seus argumentos.
- Neste caso dizemos que as funções são operadores infixos.

Aplicação de função

- Exemplos:
- Prelude> 2 + 3
- 5
- Prelude> 10 / 4
- 2.5
- Prelude> (12 - 7) * 6
- 30
- Prelude> 5 * sqrt 36
- 30.0

Aplicação de função

- Assim como na Matemática e em outras linguagens de programação,
- os operadores possuem um nível de precedência (ou prioridade) e uma associativade.
- Precedência: Escolha do operador com maior prioridade
- Associatividade: Escolha se os operadores tiveram a mesma prioridade

Aplicação de função

- Parênteses podem ser usados para agrupar subexpressões dentro de expressões maiores quebrando a precedência ou associatividade dos operadores.

Aplicação de função

- O nível de precedência de um operador é dado por um número entre 0 e 9, inclusive.
- Se dois operadores disputam um operando, o operador de maior precedência é escolhido.
- Prelude> 2 + 3 * 4
- Qual o valor desta expressão?
- Qual o operando disputado?

Aplicação de função

- E agora…
- Prelude> 4 / 2 * 2
- Qual o valor desta expressão?
- Qual o operando disputado?

Aplicação de função

precedência	associativade	operador	descrição
9	esquerda	`!!`	índice de lista
	direita	`.`	composição de funções
8	direita	`^`	potenciação com expoente inteiro não negativo
		`^^`	potenciação com expoente inteiro
		`**`	potenciação com expoente em ponto flutuante
7	esquerda	`*`	multiplicação
		`/`	divisão fracionária
		`` `div` ``	quociente inteiro truncado em direção a $-\infty$
		`` `mod` ``	módulo inteiro satisfazendo `(div x y)*y + (mod x y) == x`
		`` `quot` ``	quociente inteiro truncado em direção a 0
		`` `rem` ``	resto inteiro satisfazendo `(quot x y)*y + (rem x y) == x`
6	esquerda	`+`	adição
		`-`	subtração
5	direita	`:`	construção de lista não vazia
		`++`	concatenção de listas
4	não associativo	`==`	igualdade
		`/=`	desigualdade
		`<`	menor que
		`<=`	menor ou igual a
		`>`	maior que
		`>=`	maior ou igual a
		`` `elem` ``	pertinência de lista
		`` `notElem` ``	negação de pertinência de lista
3	direita	`&&`	conjunção (*e* lógico)
2	direita	`||`	disjunção (*ou* lógico)
1	esquerda	`>>=`	composição de ações sequenciais
		`>>`	composição de ações sequenciais (ignora o resultado da primeira)
0	direita	`$`	aplicação de função
		`$!`	aplicação de função estrita
		`` `seq` ``	avaliação estrita

Aplicação de função

- Exemplos:
- Prelude> 2 + 3 * 4 -- * tem maior precedência que +
- 14
- Prelude> 5 ^ 2 - 10 -- ^ tem maior precedência que -
- 15
- Prelude> 4 / 2 * 2 -- ^ associa-se à esquerda
- 4.0

Aplicação de função

- Aplicações de função em notação prefixa tem prioridade maior do que todos os operadores.
- Exemplos:
- Prelude> abs 10 - 20 -- abs tem precedência maior que -
- -10
- Prelude> abs (10 - 20)
- 10

Aplicação de função

- Mais exemplos:
- Prelude> succ 9 + max 5 4 * 3 -- succ e max tem precedência maior que + e *
- 25
- Prelude> 2 * logBase (8/2) 256 + 1000
- 1008.0

Aplicação de função

- Um operador pode ser associativo à esquerda, associativo à direita, ou não-associativo.
- Quando dois operadores com a mesma precedência disputam um operando,
  - •se eles forem associativos à esquerda, o operador da esquerda é escolhido,
  - •se eles forem associativos à direita, o operador da direta é escolhido,
  - •se eles forem não associativos, a expressão é mal formada e contém um erro de sintaxe,

Aplicação de função

- Prelude> 15 - 4 - 6 -- - associa-se à esquerda
- 5
- Prelude> 15 - (4 - 6)
- 17
- Prelude> 10 - 2 + 5 -- + e - tem a mesma precedência e associam-se à esquerda
- 13
- Prelude> 10 - (2 + 5)
- 3
- Prelude> 2^3^2 -- ^ associa-se à direita
- 512

Aplicação de função

- O símbolo - merece atenção especial, pois ele pode tanto ser a função de subtração (operador infixo) como a função de inversão de sinal (operador prefixo).
- Prelude> 6 - 2
- 4
- Prelude> - 5
- -5

Aplicação de função

- A notação prefixa é usada com nomes de funções que são identificadores alfanuméricos:
- formados por uma sequência de letras, dígitos decimais, sublinhado (_) e apóstrofo (’) começando com letra minúscula ou sublinhado (e que não seja uma palavra reservada).

Aplicação de função

- Já a notação infixa é usada com nomes de funções simbólicos
- formados por uma sequência de símbolos especiais (!#$%& +./<=>?@|\^-~:) que não começa com:.(ponto)

Aplicação de função

- Qualquer operador pode ser usado em notação prefixa, bastando escrevê-lo entre parênteses. Exemplos:
- Prelude> (+) 4 5
- 9
- Prelude> (/) 18.2 2
- 9.1
- Prelude> (>=) 10 20
- False
- Prelude> sqrt ((+) 4 5)
- 3

Aplicação de função

- Qualquer função prefixa de dois argumentos pode ser usada em notação infixa, bastando escrevê-la entre apóstrofos invertidos (sinal de crase: ‘), com precedência padrão 9 e associatividade à esquerda. Exemplos:
- Prelude> 20 ‘div‘ 3
- 6
- Prelude> 20 ‘mod‘ 3
- 2
- Prelude> 20 ‘mod‘ 3 == 0
- False

Nomeando valores

- Quando uma expressão é avaliada diretamente no ambiente interativo,
- Uma variável chamada *it* é automaticamente definida para denotar o valor da expressão

Nomeando valores

- Exemplo:
- Prelude> 2 + 3 * 4

14

- Prelude> it

14

- Prelude> 7*(it - 4)

70

- Prelude> it

70

Nomeando valores

- Uma declaração **let** pode ser usada para definir uma variável no ambiente interativo.
- Por exemplo:
- Prelude> let idade = 2 + 3 * 4
- Prelude> idade

14

- Prelude> 7*(idade - 4)

70

Exercício 1

- A posição s de um corpo em movimento retilíneo uniformemente variado, em função do tempo t, é dado pela equação
- $s = s_0 + v_0t + 1/2at^2$
- onde s_0 é a posição inicial do corpo, v_0 é a sua velocidade inicial, e a é a sua aceleração.

Exercício 1

- Utilize o ambiente interativo GHCi para calcular a posição de uma bola em queda livre no instante t = 8 s, considerando que a posição inicial é $s_0 = 100$ m, a velocidade inicial é $v_0 = 15$ m/s e a aceleração da gravidade é $a = -9.81$ m/s^2 .

Exercício 1

- Dica:
- Use a declaração *let* para criar variáveis correspondentes aos dados e em seguida avalie a expressão correspondente à função horária do movimento usando estas variáveis.

Definindo variáveis e funções

- Além de poder usar as funções das bibliotecas,
- o programador também pode definir e usar suas próprias funções.

- Novas funções são definidas em arquivos texto geralmente chamados de **código fonte** ou **programa fonte** ou ainda script.

Definindo variáveis e funções

- Um programa fonte contém **definições** de
  - variáveis,
  - funções,
  - tipos,
  - etc.
- usadas para estruturar o código da aplicação.

Definindo variáveis e funções

- Por convenção, arquivos de programas fonte em Haskell normalmente tem a extensão **.hs** em seu nome.
- Isso não é obrigatório, mas é útil para fins de identificação.
- Por exemplos:
- prog1.hs, prog2.hs
- circulo.hs, temperatura.hs

Definindo variáveis e funções

- Variáveis são definidas usando equações.
- No lado esquerdo de uma equação colocamos o nome da variável
- No lado direito colocamos uma expressão cujo valor será o valor da variável
- Por exemplo:
- y = 5
- x = y + 3

Definindo variáveis e funções

- Funções também são definidas usando equações.
- No lado esquerdo de uma equação colocamos o nome da função seguido de seus **parâmetros formais**.
- No lado direito colocamos uma expressão cujo valor será o resultado da função quando a função for aplicada em seus **argumentos**

Definindo variáveis e funções

- Por exemplo:
- dobro x = x + x
- quadruplo x = dobro (dobro x)

- Nomes de funções e variáveis podem ser alfanuméricos ou simbólicos

Definindo variáveis e funções

- identificadores alfanuméricos
- começam com uma letra minúscula ou sublinhado e podem conter letras, dígitos decimais, sublinhado (_) e apóstrofo (aspa simples ’)
- são normalmente usados em notação prefixa
- exemplos:
- myFun
- fun1
- arg_2
- x’

Definindo variáveis e funções

- identificadores simbólicos
- formados por uma sequência de símbolos e não podem começar com dois pontos(:)
- são normalmente usados em notação infixa
- exemplos:

  <+>

  ===

  $*=*$

  +=

Comentários

- Comentários são usados para fazer anotações no programa que podem ajudar a entender o funcionamento do mesmo
- Os comentários são ignorados pelo compilador
- Um comentário de linha é introduzido por -- e se estende até o final da linha
- -- calcula o dobro de um número
- dobro x = x + x

Comentários

- {-

Um comentário de bloco é delimitado por {- e -}

Comentários de bloco podem ser aninhados

- -}

Definições locais em equações

- Em Haskell equações são usadas para definir variáveis e funções
- Em muitas situações é desejável poder definir valores e funções auxiliares em uma definição principal.

- Exemplo em matemática:
- f(x,y) = x + y onde x > y

Definições locais em equações

- Em Haskell, isto pode ser feito escrevendo-se uma cláusula where ao final da equação.
- Uma cláusula where é formada pela palavra chave where seguida das definições auxiliares.

Definições locais em equações

- A cláusula where faz definições que são locais à equação,
- ou seja, o escopo dos nomes definidos em uma cláusula where restringe-se à menor equação contendo a cláusula where.

Definições locais em equações

- Por exemplo, considere a fórmula de Heron
- $A^2 = s(s - a)(s - b)(s - c)$ para calcular a área de um triângulo com lados a, b e c, sendo s o semiperímetro do triângulo.
- $s=(a+b+c)/2$

Definições locais em equações

- Como s aparece várias vezes na fórmula, podemos defini-lo localmente uma única vez e usá-lo quantas vezes forem necessárias na equação.
- areaTriangulo a b c = sqrt (s * (s-a) * (s-b) * (s-c))
  - where
    - s = (a + b + c)/2

Definições locais em equações

- Exemplo de uso:
- Prelude> areaTriangulo 5 6 8
  14.981238266578634

Exercício 2

- Utilize o ambiente interativo GHCi para testar as funções dobro, quadruplo e areaTriangulo.