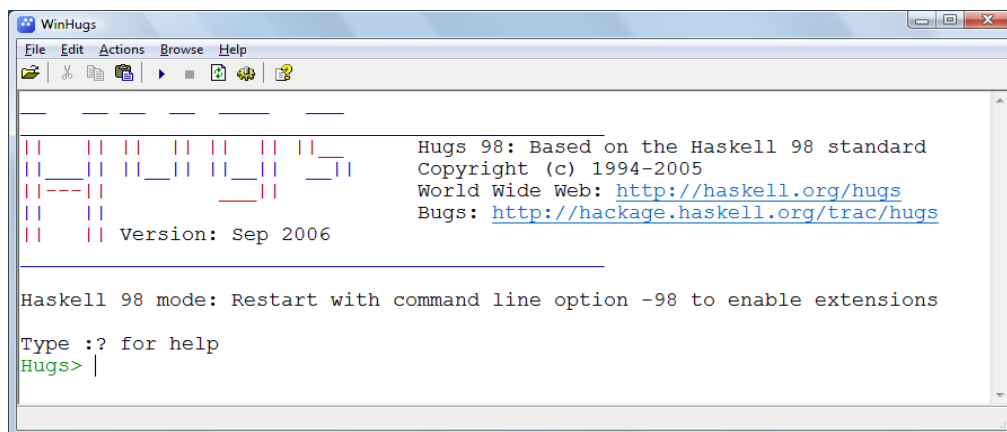




Faculdade de Computação
Programação Funcional (BCC/BSI) - 1º Período
Aula Prática: Expressões e Funções em Haskell

WinHugs

O sistema Hugs é uma implementação da linguagem Haskell que pode ser executada em PCs e sistemas Unix, incluindo Linux. Pode ser obtido gratuitamente em www.haskell.org/hugs. A figura abaixo mostra a tela inicial do WinHugs (Hugs para Windows) em que podemos carregar programas ou executar diretamente expressões.



Avaliando expressões no Hugs:

O sistema Hugs utiliza a biblioteca *Prelude.hs*, que é sempre carregada inicialmente. O prompt '>' indica que o sistema está esperando por uma expressão a ser avaliada. A biblioteca *Prelude.hs* define funções básicas para operações sobre inteiros, incluindo as 5 operações aritméticas de adição, subtração, multiplicação, divisão e exponenciação (usando os símbolos ^ ou **, para operandos positivos e negativos respectivamente).

```
Hugs> 2 * 3 - 7
-1
Hugs> 2 ^ 5
32
Hugs> 2 ** (-5)
0.03125
Hugs> 2 ^ 3 ^ 4
2417851639229258349412352
Hugs> 2 * 7 - 13 + 2 / 3
1.6666666666666667
Hugs> 2 ^ 3 * 4
32
```

Podemos notar que $2 * 3 - 7$ é o mesmo que $(2 * 3) - 7$ pois o símbolo de multiplicação tem precedência sobre adição/subtração. Em caso de operadores de mesma precedência, a associação é feita da esquerda para a direita: $2 - 1 + 4$ é visto como $(2 - 1) + 4$, com exceção do operador de exponenciação. Numa expressão como $2 ^ 3 ^ 4$ teremos $2 ^ (3 ^ 4)$.

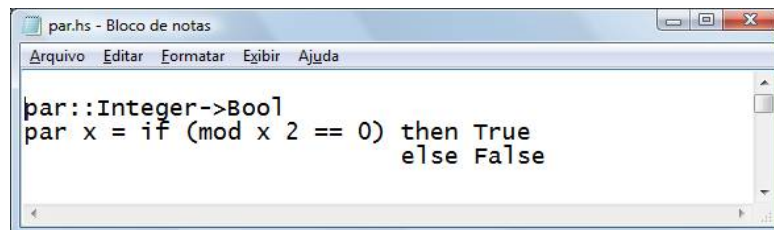


Faculdade de Computação
Programação Funcional (BCC/BSI) - 1º Período
Aula Prática: Expressões e Funções em Haskell

Na prática de programação podemos adotar o uso de parênteses para tornar mais claras as expressões e explicitar as convenções citadas acima.

Editando um programa:

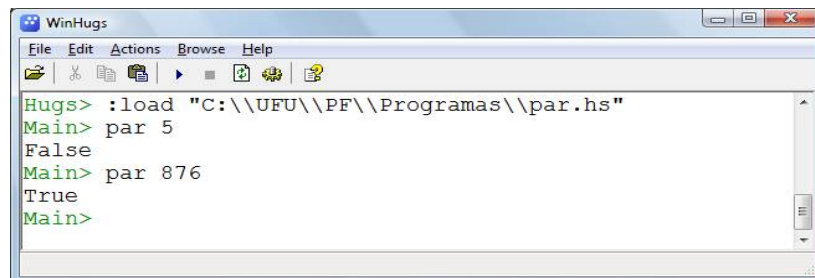
Para escrever e editar um programa em Haskell podemos utilizar o *Bloco de Notas*. Vários programas podem estar num mesmo arquivo que deve ser armazenado com a extensão (.hs). A figura abaixo mostra o exemplo da função **par**, que retorna verdadeiro se um número inteiro for par, e falso, caso contrário.



```
par.hs - Bloco de notas
Arquivo  Editar  Formatar  Exibir  Ajuda

par::Integer->Bool
par x = if (mod x 2 == 0) then True
        else False
```

A janela abaixo exemplifica a carga e a execução da função **par** no ambiente WinHugs:



```
WinHugs
File  Edit  Actions  Browse  Help
-----
Hugs> :load "C:\\UFU\\PF\\Programas\\par.hs"
Main> par 5
False
Main> par 876
True
Main>
```

Alguns comandos úteis no Hugs:

Comando	Significado
:load "arq.ext"	carregar o arquivo
:reload	recarregar o arquivo atual
:edit "arq.ext"	editar o arquivo pedido
:edit	editar o arquivo atual
:type expr	mostrar o tipo de uma expressão
:?	mostrar todos os comandos
:quit	encerrar o Hugs



Exercícios:

1) Teste as expressões abaixo e explique cada resultado.

```
> 7 `div` 2
> 7 `div` 2 == div 7 2
> x * 3 where x = 10
> 3 > 5
> False == False
> if (mod 5 2 == 0) then 1 else 2
> 'a' > 'b'
> "abc" == "abc"
> "elegante" < "elefante"
> [1, 2, 3] < [1, 2]
> :type (+)
```

2) Teste as expressões abaixo e defina parênteses para as mesmas:

```
2 ^ 3 * 4
2 * 3 + 4 * 5
2 + 3 * 4 ^ 5
```

3) Escreva uma função para calcular a multiplicação de três números inteiros, como no exemplo abaixo:

```
> mult 3 4 5
60
```

4) Escreva uma função que recebe um valor numérico e devolva o valor 1 se o valor for maior que zero, -1 se for negativo, 0 se for zero.

5) Faça uma função que, dados três parâmetros de entrada, se o primeiro for um asterisco, os outros dois serão multiplicados; se for uma barra, o segundo deve ser dividido pelo terceiro; se não for nenhum dos dois, imprima uma mensagem de erro.

6) Seja a função `ehDigito` definida abaixo para um valor do tipo `Char`. Faça uma função que verifique se um `Char` é uma letra.

```
ehDigito :: Char -> Bool
ehDigito c = c >= '0' && c <= '9'
```

7) Escreva funções para calcular:

- (a) Uma equação do primeiro grau ($ax + b$)
- (b) Uma equação do segundo grau ($ax^2 + bx + c$)