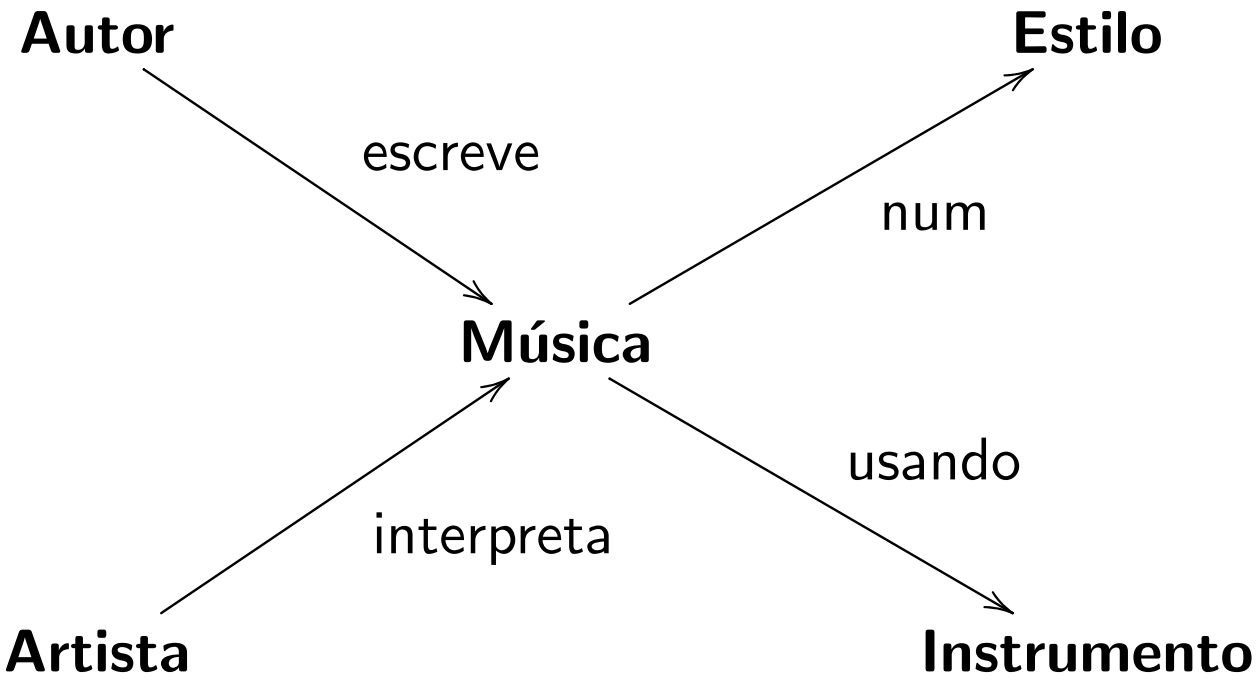
Computação sem fronteiras

Dept. Informática,
Universidade do Minho
Braga, Portugal

18 a 22 de Julho de 2011

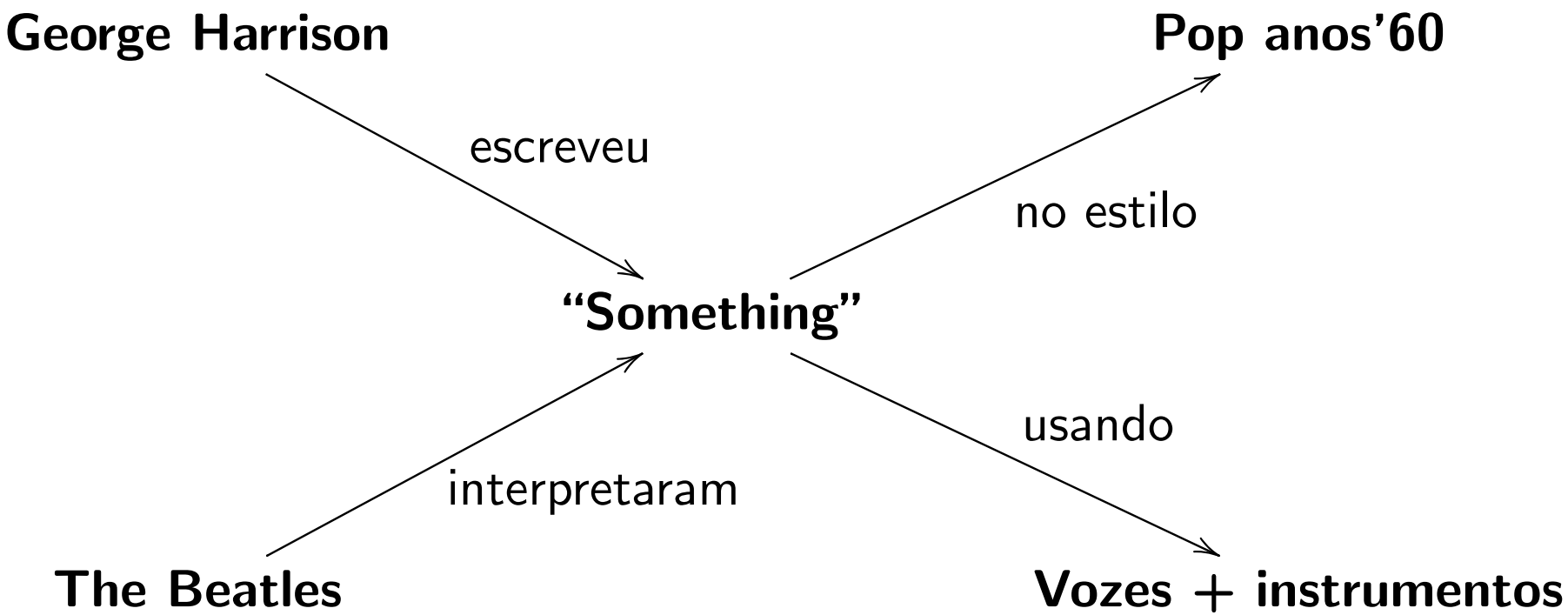
O que é programar?

Analogia:



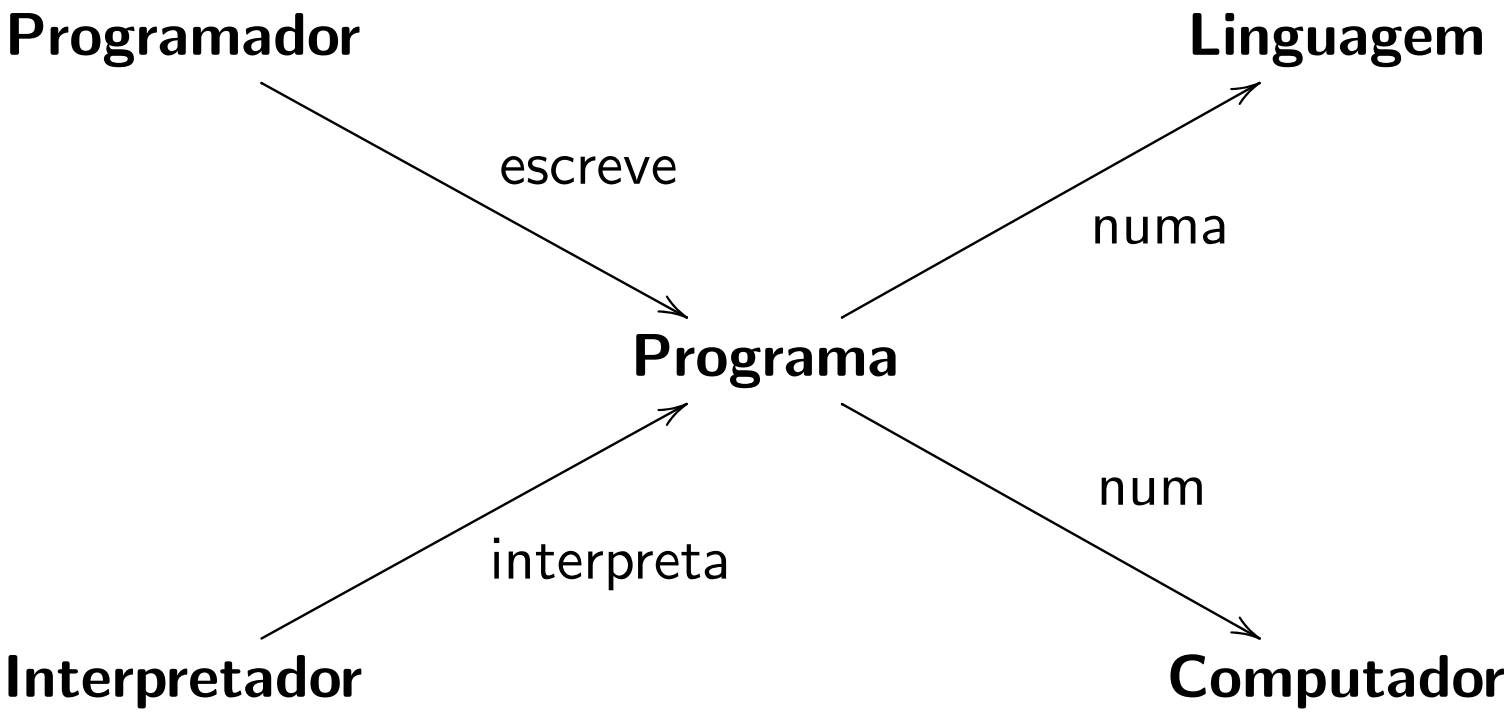
Programar computadores

Exemplo:



Programar computadores

Programação:



Programar computadores

Editar? Interpretar? Então vamos precisar de

- Um editor de texto — AQUAMACS

- Um interpretador dos programas — GHCI

- Um computador — MAC

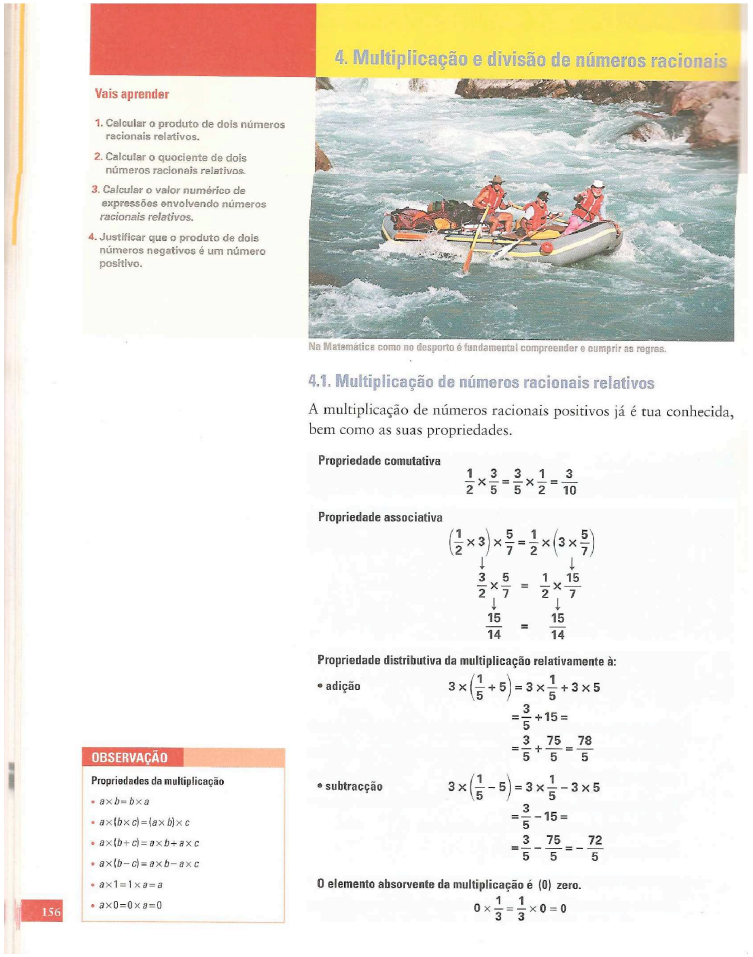
(mas podia ser outro qualquer)

Programar computadores

- Será suficiente?
- Não — para começar **bem**, precisamos de um livro de **matemática**...

Onde começa a programação?

Primeira página do capítulo sobre multiplicação e divisão de números racionais (7.º ano):



4. Multiplicação e divisão de números racionais

Vais aprender

1. Calcular o produto de dois números racionais relativos.
2. Calcular o quociente de dois números racionais relativos.
3. Calcular o valor numérico de expressões envolvendo números racionais relativos.
4. Justificar que o produto de dois números negativos é um número positivo.

Na Matemática como no desporto é fundamental compreender e cumprir as regras.

4.1. Multiplicação de números racionais relativos

A multiplicação de números racionais positivos já é tua conhecida, bem como as suas propriedades.

Propriedade comutativa

$$\frac{1}{2}\times\frac{3}{5}=\frac{3}{5}\times\frac{1}{2}=\frac{3}{10}$$

Propriedade associativa

$$\left(\frac{1}{2}\times 3\right)\times\frac{5}{7}=\frac{1}{2}\times\left(3\times\frac{5}{7}\right)$$
$$\frac{3}{2}\times\frac{5}{7}=\frac{1}{2}\times\frac{15}{7}$$
$$\frac{15}{14}=\frac{15}{14}$$

Propriedade distributiva da multiplicação relativamente à:

- adição $3\times\left(\frac{1}{5}+5\right)=3\times\frac{1}{5}+3\times 5$
$=\frac{3}{5}+15=$
$=\frac{3}{5}+\frac{75}{5}=\frac{78}{5}$
- subtracção $3\times\left(\frac{1}{5}-5\right)=3\times\frac{1}{5}-3\times 5$
$=\frac{3}{5}-15=$
$=\frac{3}{5}-\frac{75}{5}=-\frac{72}{5}$

O elemento absorvente da multiplicação é (0) zero.

$$0\times\frac{1}{3}=\frac{1}{3}\times 0=0$$

OBSERVAÇÃO

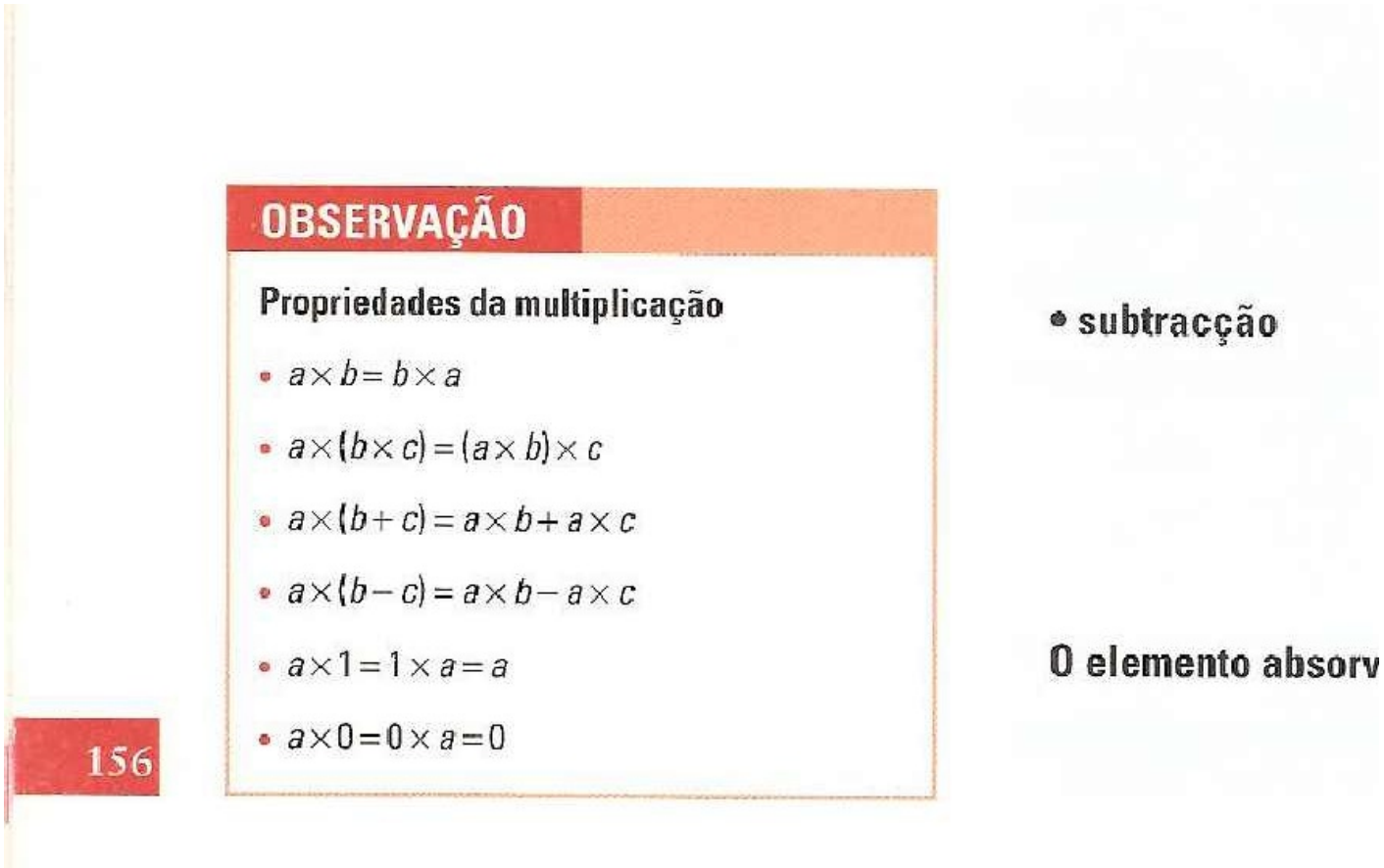
Propriedades da multiplicação

- $a\times b=b\times a$
- $a\times(b\times c)=(a\times b)\times c$
- $a\times(b+c)=a\times b+a\times c$
- $a\times(b-c)=a\times b-a\times c$
- $a\times 1=1\times a=a$
- $a\times 0=0\times a=0$

156

Multiplicação de racionais

Prestem atenção à caixa ao fundo, à esquerda:



OBSERVAÇÃO

Propriedades da multiplicação

- $a \times b = b \times a$
- $a \times (b \times c) = (a \times b) \times c$
- $a \times (b + c) = a \times b + a \times c$
- $a \times (b - c) = a \times b - a \times c$
- $a \times 1 = 1 \times a = a$
- $a \times 0 = 0 \times a = 0$

156

• subtracção

0 elemento absorv

Primeiro programa: multiplicação

As seguintes **propriedades** da multiplicação, que aparecem nessa caixa,

$$a \times 0 = 0$$
$$a \times 1 = a$$
$$a \times (b + c) = (a \times b) + (a \times c)$$

vão ser suficientes para começarmos a escrever programas envolvendo adição e multiplicação.

Primeiro programa: multiplicação

Vejamos como: fazendo $c = 1$ na terceira linha de

$$a \times 0 = 0$$
$$a \times 1 = a$$
$$a \times (b + c) = (a \times b) + (a \times c)$$

ter-se-á

$$a \times 0 = 0$$
$$a \times 1 = a$$
$$a \times (b + 1) = (a \times b) + (a \times 1)$$

Primeiro programa: multiplicação

Ainda podemos simplificar

$$a \times 0 = 0$$
$$a \times 1 = a$$
$$a \times (b + 1) = (a \times b) + (a \times 1)$$

para

$$a \times 0 = 0$$
$$a \times 1 = a$$
$$a \times (b + 1) = (a \times b) + a$$

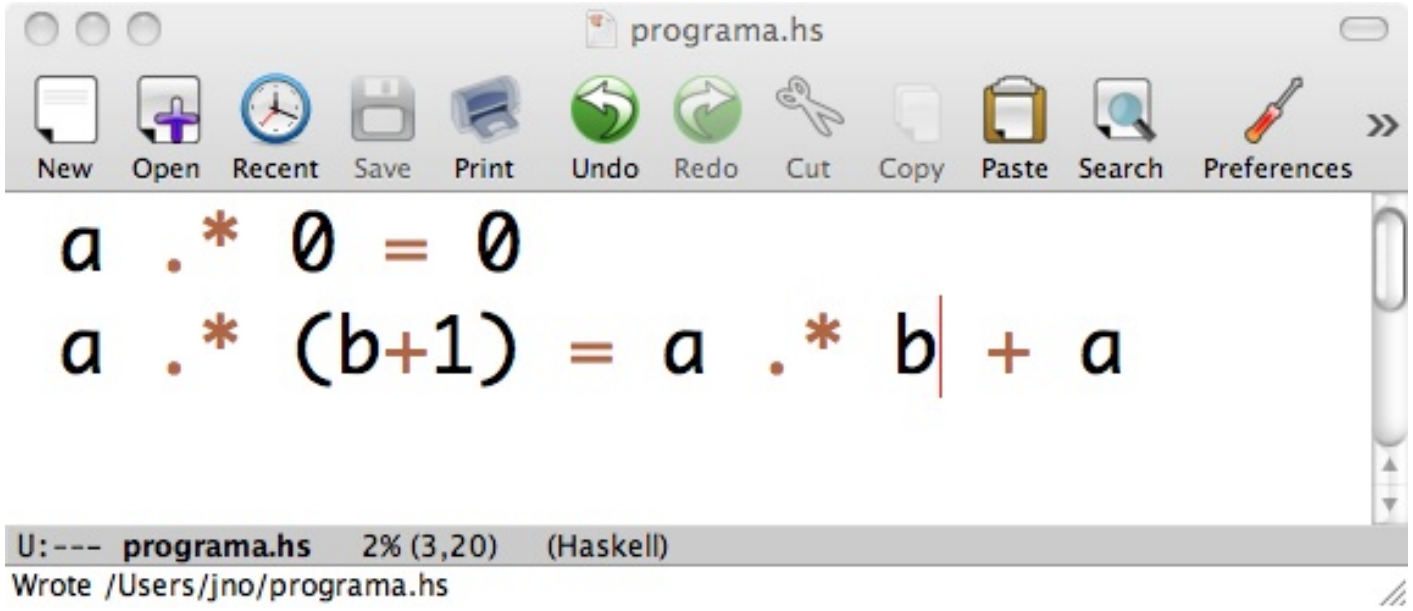
pois $a \times 1 = a$. E esta linha, a do meio, pode sair, pois corresponde à terceira para $b = 0$.

Primeiro programa: multiplicação

Em suma, temos:

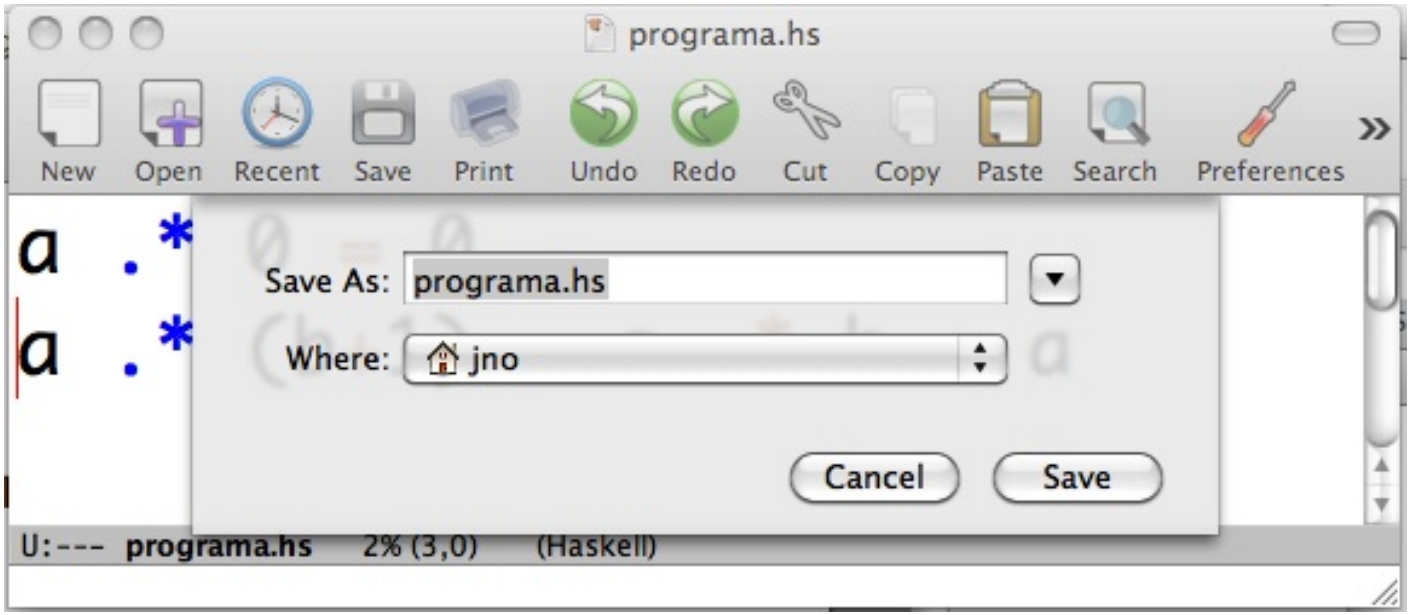
$$a \times 0 = 0$$
$$a \times (b + 1) = (a \times b) + a$$

— o nosso **programa** para multiplicar números naturais. Vamos **editá-lo** no AQUAMACS:



Primeiro programa: multiplicação

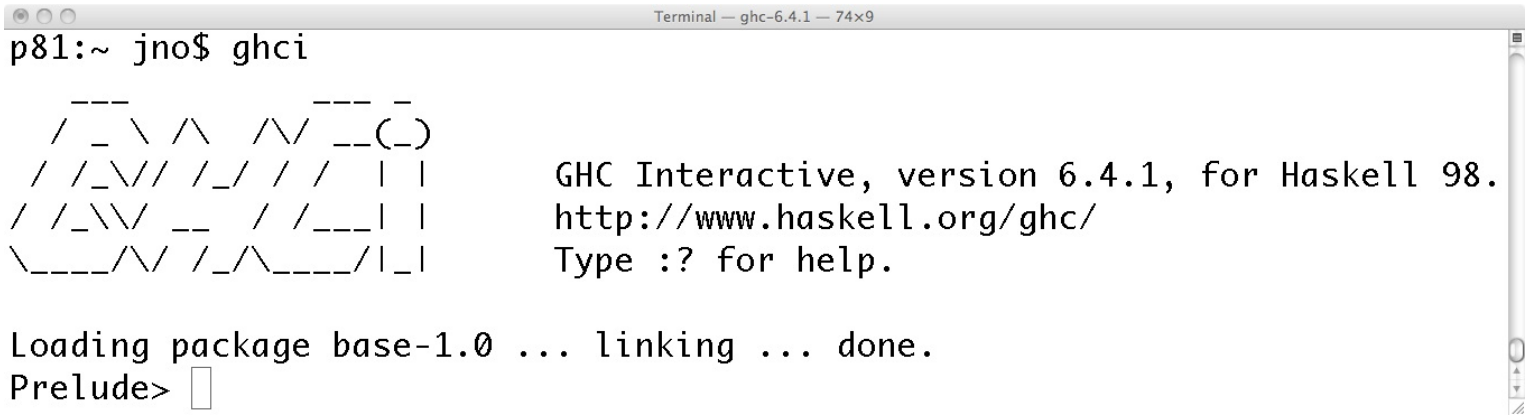
De seguida, é melhor gravá-lo para que se não perca a nossa "obra":



Escrita que está a "nossa música", vamos passá-la a alguém que a saiba "tocar".

Primeiro programa: multiplicação

É o que sabe fazer o **interpretar** GHCi, que entretanto é preciso acordar, noutra janela:



Este interpretador precisa de "ler a partitura" antes de a começar a tocar:

Primeiro programa: multiplicação

O interpretador entende o nosso comando `load programa.hs` ("l" abrevia "load"):

```
   ___         ___ _
  / _ \ /\  /\/ __(_)
 / /_\// /_/ / /   | |      GHC Interactive, version 6.4
/ /_\\/ __  / /___| |      http://www.haskell.org/ghc/
\____/\/ /_/\____/|_|      Type :? for help.

Loading package base-1.0 ... linking ... done.
Prelude> :l programa.hs
Compiling Main             ( programa.hs, interpreted )
Ok, modules loaded: Main.
*Main>
```

Uups! Esquecemos de dar um nome à "música". O GHCi não perdeu tempo e resolveu chamar-lhe `Main`.

Primeiro programa: multiplicação

Agora é só exercitarmos os dedos no "instrumento", pedindo-lhe que nos multiplique números:

```
/ /_\\/ __  / /___| |      http://www.has
\____/\/ /_/\____/|_|      Type :? for he

Loading package base-1.0 ... linking ...
Prelude> :l programa.hs
Compiling Main             ( programa.hs,
Ok, modules loaded: Main.
*Main> 23 .* 45
1035
*Main> 23 .* 0
0
*Main> 1 .* 45
45
*Main>
```

Importante: programa **necessariamente certo**, pois baseou-se apenas em leis conhecidas da matemática.

Segundo programa: quadrados

Queremos agora um programa qua responda ao seguinte

Desafio: *calcular o* **quadrado** *de um número*

$$quad\ x = x^2$$

sem fazer multiplicações.

Que fazer?

Fácil: vamos investigar como anteriormente as propriedades de $quad$.

Vejam o próximo slide.

Segundo programa: quadrados

Sabemos que

$$quad\ x = x^2$$

Ter-se-á:

$$quad\ 0 = 0^2 = 0$$
$$quad\ 1 = 1^2 = 1$$
$$quad\ (a + b) = a^2 + 2\ ab + b^2$$

(cf. binómio de Newton). Fazendo $b = 1$ e simplificando:

$$quad\ 0 = 0$$
$$quad\ 1 = 1$$
$$quad\ (a + 1) = a^2 + 2\ a + 1$$

Segundo programa: quadrados

Ora $2\ a = a + a$ e $quad\ a = a^2$; logo podemos fazer essas substituições em

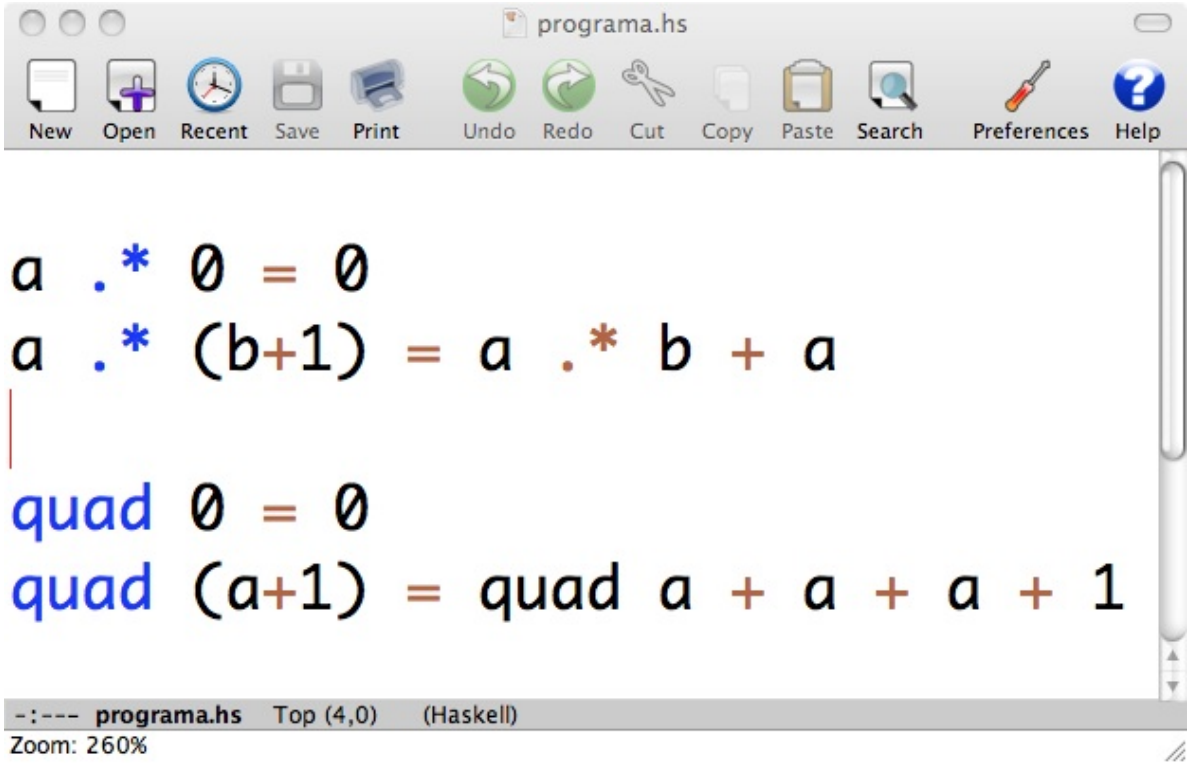
$$quad\ 0 = 0^2 = 0$$
$$quad\ 1 = 1^2 = 1$$
$$quad\ (a + 1) = a^2 + 2\ a + 1$$

obtendo (de novo removendo a linha do meio):

$$quad\ 0 = 0$$
$$quad\ (a + 1) = quad\ a + a + a + 1$$

Segundo programa: quadrados

Agora é só repetir o que fizemos atrás para a multiplicação: acrescenta-se *quad* à nossa "música", recorrendo de novo ao AQUAMACS,



grava-se de novo e carrega-se no interpretador (próximo slide):

Segundo programa: quadrados

Como o interpretador já está aberto, basta "refrescar" a versão que ele tem do programa e exercitar o que fizemos:

```
*Main> :r
Ok, modules loaded: Main.
*Main> quad 3
9
*Main> quad 5
25
*Main> quad 5 .* 2
50
*Main>
```

e por aí fora...

Sequências

- Os computadores não trabalham só com **números**
- Entendem outras coisas como, por exemplo, **palavras**

  $$\mathit{comp}\ \texttt{"ola"} = 3$$

  que são sequências de **letras** (`"ola"` tem 3 letras) e mesmo outras sequências, por exemplo

  $$\mathit{inverte}\ [1,2,3] = [3,2,1]$$
  $$[1,2,3] +\!\!+ [3,2,1] = [1,2,3,3,2,1]$$

- Quando se está a ver um **filme**, o que está a acontecer é que o computador está a ler uma **sequência enorme** de "pontos coloridos" que está a mostrar tão depressa no écran que nós temos a **ilusão** do movimento...

Programar com sequências

Questão:

Será muito difícil escrever programas com sequências?

Resposta:

Não, é muito parecido com o que fizemos para programnação com números.

Basta investigar as propriedades matemáticas daquilo que queremos fazer. O nosso exemplo será a programação da função *comp* que já apareceu em

comp `"ola"` $= 3$

e que calcula o **comprimento** de uma qualquer sequência (incluindo palavras).

Programar com sequências

Vai haver na mesma três casos a considerar:

$$comp\ [\,] = \ldots$$
$$comp\ [a] = \ldots$$
$$comp\ (x +\!\!+\ y) = \ldots$$

em que

- $[\,]$ — sequência vazia (corresponde ao 0 dos programas com números)
- $[a]$ — sequência com apenas um elemento (corresponde ao 1 dos programas com números)
- $x +\!\!+\ y$ — sequência é junção de duas outras quaisquer sequências x e y (corresponde ao $x + y$ dos programas com números).

Programar com sequências

Essa correspondência vê-se bem no caso que temos em mão — calcular o comprimento *comp* de uma sequência —; de facto, as propriedades

$$comp\ [\,] = 0$$
$$comp\ [a] = 1$$
$$comp\ (x +\!\!+\ y) = (comp\ x) + (comp\ y)$$

são fáceis de encontrar.

E o que se faz a seguir?

Muito parecido com o que fizemos atrás para números — ora vejam:

Programar com sequências

Na terceira linha, começamos por substituir x por $[a]$:

$$comp\ [\,] = 0$$
$$comp\ [a] = 1$$
$$comp\ ([a] +\!\!+\ y) = (comp\ [a]) + (comp\ y)$$

Agora vamos (tal como anteriormente) usar a segunda linha (que pode desaparecer) para simplificar a terceira:

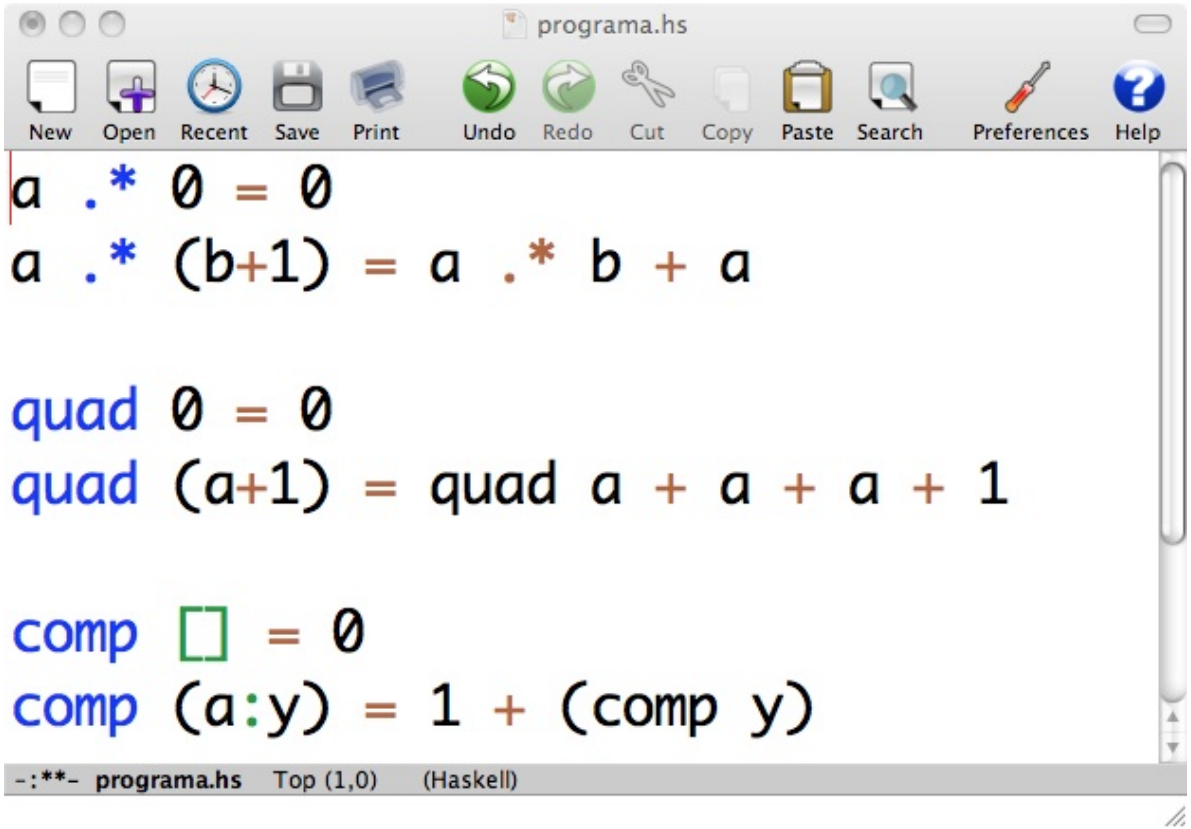
$$comp\ [\,] = 0$$
$$comp\ ([a] +\!\!+\ y) = 1 + (comp\ y)$$

Finalmente, simplificamos a expressão $[a] +\!\!+\ y$ para $a : y$, querendo dizer *"a sequência que começa por a e continua como y* :

$$comp\ [\,] = 0$$
$$comp\ (a : y) = 1 + (comp\ y)$$

Programar com sequências

Concebida esta nova operação, sobre sequências, vamos acrescentá-la ao nosso programa, usando mais uma vez o AQUAMACS,



não esquecendo gravar esta nova versão.

Programar com sequências

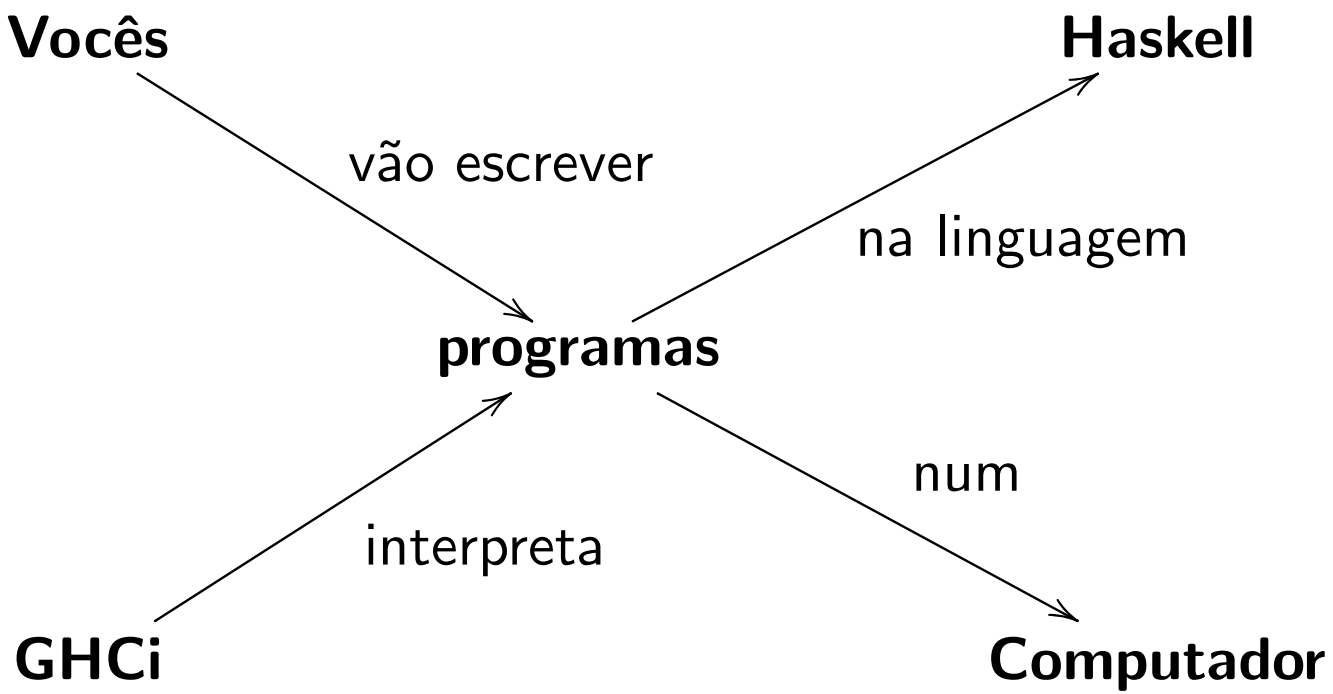
De novo, é só carregar no interpretador e testar a nova função:

```
*Main> comp "Maria"
5
*Main> comp "22-Julho-2011"
13
*Main> comp [12,23,34]
3
*Main>
```

Ok?

- Agora vão ser vocês a fazer o resto — cf. folha de exercícios que vos vou entregar para resolverem.
- Não se esqueçam: análise dos 3 casos, simplificação a seguir, edição, teste, etc.

Em suma


Vocês
Haskell
vão escrever
na linguagem
programas
num
interpreta
GHCi
Computador

Nota final

Antes de se passar à prática:

- Há MUITO MAIS a aprender, mas já perceberam o mais importante:
  - que a programação está ligada à matemática
  - que segue um processo sistemático para garantir que, no final, tudo funciona bem.
- (Quem assim não fizer acaba por não saber programar... mesmo que pense o contrário!)
- Na Universidade do Minho o nosso lema é ensinar a construir programas que funcionam "à primeira" :-)

Contacto: ✉ jno@di.uminho.pt