02

Casting

Casting de tipos primitivos

Não podemos atribuir a uma variável de um tipo um valor que não é compatível com ela:

```
double d = 3.14;
int i = d;
```

COPIAR CÓDIGO

Só podemos fazer essas atribuições se os valores forem ::compatíveis::. Compatível é quando um tipo cabe em outros, ou seja, ele só cabe se o ::range:: (alcance) dele for mais amplo que o do outro.

byte -> short -> int -> long -> float -> double

char -> int

Se estivermos convertendo de um tipo que vai da esquerda para a direita nessa tabelinha, não precisamos de casting, a **autopromoção** fará o serviço por nós.

Se estamos indo da direita para a esquerda, precisamos do ::casting:: e não importam os valores que estão dentro. Exemplo:

```
double d = 0;
float  f = d;
```

COPIAR CÓDIGO

Esse código não compila sem um casting! O casting é a maneira que usamos para moldar uma variável de um tipo em outro. Nós estamos avisando o compilador que sabemos da possibilidade de perda de precisão ou truncamento, mas nós realmente queremos fazer isso:

```
float  f = (float) d;
```

COPIAR CÓDIGO

Podemos fazer casting entre ponto flutuante e inteiro, o resultado será o número truncado, sem as casas decimais:

```
double d = 3.1415;
int i = (int) d; // 3
```

COPIAR CÓDIGO

Dica

Não é preciso decorar a sequência int->long->float etc. Basta lembrar os alcances das variáveis. Por exemplo, o `char` tem dois bytes e guarda um número positivo. Será então que posso atribuir um `char` a um `short`? Não, pois um `short` tem 2 bytes, e usa meio a meio entre os números positivos e negativos.