基本型の縮小変換

前のプログラムでは、int 型の値を double 型として取り出すために、キャスト演算子を利用しました。逆の変換について、以下のコードで考えましょう。

```
   int a;
1  a = 10.0;              // エラー
2  a = (int)10.0;         // ＯＫ
```

このコードをコンパイルすると、1はエラーとなります。そのため、2のようにキャストが必須です。

その理由を Fig.5-17 を見ながら考えていきましょう。

図aに示すように、代入元の値が 10.0 であれば、int 型の表現範囲に収まっているため、代入は可能なはずです。

また、図bに示すように、代入元の値が 10000000000.0 であれば、int 型の表現範囲を超えていますので、代入は不可能なはずです（値が容器からあふれ出ますから）。

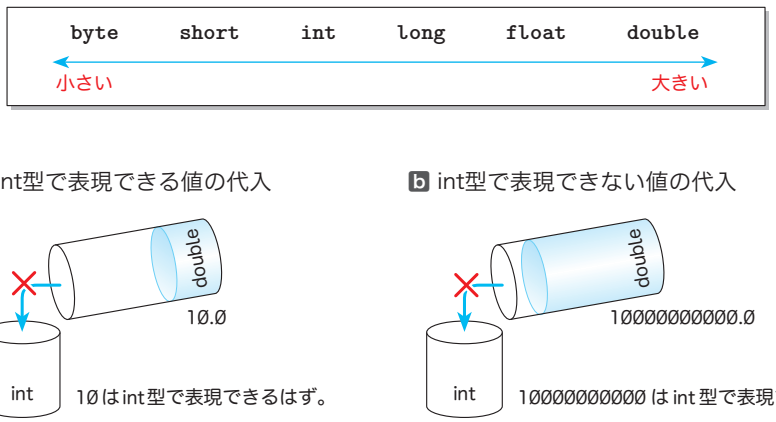

Fig.5-17　より小さい型への値の代入

とはいえ、代入元の値が、代入先の型で表現できる範囲内であるかどうかを、代入のたびにチェックしているのでは、プログラムが大きくなり、実行速度も低下します。

そのため、double 型の値を int 型に代入することは、（代入される値とは関係なく、たとえ 10.0 であっても 10000000000.0 であっても）許されないことになっています。

そのため、より小さい型への値の代入は、そのままでは行えず、キャストが必須となるのです。

重要　より小さい型への値の代入の際は、キャストが必要である。

次の 22 個の変換を**基本型の縮小変換**（*narrowing primitive conversion*）と呼びます。

- **short** から **byte**、**char** への変換。
- **char** から **byte**、**short** への変換。
- **int** から **byte**、**short**、**char** への変換。
- **long** から **byte**、**short**、**char**、**int** への変換。
- **float** から **byte**、**short**、**char**、**int**、**long** への変換。
- **double** から **byte**、**short**、**char**、**int**、**long**、**float** への変換。

縮小変換では、原則としてキャストが必須です。変換に伴って、数値の《大きさ》の情報や《精度》を失うことがあります。

定数の代入・定数による初期化

基本型の縮小変換では、原則としてキャストが必須であるとはいえ、例外があります。それは、以下のような例です。

```
byte a = 0;          // OK
a = 5;               // OK
short b = 53;        // OK
```

すでに学習したように、整数接尾語 **l** や **L** の付かない整数リテラルは **int** 型です。したがって、**byte** 型や **short** 型に対して、**int** 型である 0 や 5 や 53 の値を入れる際は、キャストが必要なはずです。しかし、キャストしなくてもコンパイルエラーとならないのは、以下の規則があるからです。

代入の右辺の式や初期化子が **byte**、**short**、**char**、**int** 型の**定数式**で、代入先あるいは初期化先の変数の型が **byte**、**short**、**char** であって、定数式の値が変数の型で表現できる場合は、基本型の縮小変換が自動的に行われる（キャストは不要である）。

キャストせずに入れることができるのが、**定数式に限られる**ことに注意しましょう。変数であれば、必ずキャストが必要です。

```
short a = 1;         // OK
byte b = a;          // エラー
```

また、浮動小数点型に対しては、このような規則はありません。したがって、**float** 型の変数に対して、キャストすることなく **double** 型の定数値を入れることはできません。以下に例を示します。

```
float a = 3.14;          // エラー：3.14はdouble型
float b = (float)3.14;   // OK
float c = 3.14f;         // OK：3.14fはfloat型
```