## *Приведение типов(Introduction)*

Значения любого базового типа данных занимают фиксированное количество байт в памяти. Такое хранение влияет на результат выполнения операторов с различными типами данных. Чтобы избежать ошибки, нужно правильно определять тип переменной, в которую будет помещен результат выражения.

Существует ряд правил, которые **определяют тип результирующего выражения:**

* **Правило1:** Если в выражении участвуют только целочисленные литералы и/или переменные до типа **int** включительно, то результат выражения будет **int**. Другими словами, даже если в выражении не будет переменных типа **int**, результат выражения все равно будет **int**.
* **Правило2:**Если в выражении есть переменная или литерал типа **long**, то результат выражения будет **long**.
* **Правило3:** Если в выражении есть переменная или литерал типа **float**, то результат выражения будет **float**.
* **Правило4:** Если в выражении есть переменная или литерал типа **double**, то результат выражения будет **double**.

Если в выражении используются данные различных типов, то для его выполнения необходимо привести все данные к одному типу. Существует два вида приведения для примитивных типов – неявное и явное.

* **Неявное**: Неявно выполняется только расширяющее приведение типов. Оно не позволяет потерять информацию о величине числового значения. Выполняется автоматически Java.
* **Явное**: Явно выполняется сужающее приведение типов. При этом возможна потеря информации о величине числового значения, а также точности и диапазона. Выполняется программистом.

**Расширяющее приведение примитивных типов** – это 19 конкретных преобразований:

* byte->short, int, long, float, double
* short->int, long, float, double
* char->int, long, float, double
* int->long, float, double
* long->float, double
* float->double.

**При сужающем приведении одного** целого числа к другому целому типу отбрасываются все биты, кроме n младших разрядов (где n – это число битов для представления типа, к которому приводим). При этом **возможны**:

* потеря данных о величине значения
* изменение знака результирующего значения.

**Сужающее преобразование из типа** ***double*** в тип ***float*** регулируется правилами округления IEEE 754. При этом может произойти:

* потеря точности
* потеря диапазона (например, нулевое значение типа **float** может получиться из ненулевого **double**, а бесконечное значение типа **float** – из конечного **double**).

**Компиляция java-приложения** отличается от компиляции приложений С-подобных языков программирования. Статический компилятор преобразует исходный код в машинные инструкции, которые могут быть выполнены на целевой платформе. Но java-компилятор преобразует исходный код в переносимые байткоды JVM, которые для JVM являются "инструкциями виртуальной машины". При этом компилятор javac выполняет незначительную оптимизацию, в отличие от других статических компиляторов, которые проводят оптимизацию во время исполнения программы.