ПЕРЕМЕННЫЕ И ТИПЫ ДАННЫХ

Представление информации в компьютере

Компьютер использует для хранения информации **двоичный** (**бинарный**) код, состоящий из двух цифр: 0 и 1. Этот код является основой внутреннего языка, на котором "говорит" компьютер.

Для корректной интерпретации информации важно знание её **типа**, будь то число, буква или другой символ. Это помогает правильно обрабатывать и представлять данные.

Бит - это базовая единица информации в компьютере. Он может принимать значение 0 (ложь) или 1 (истина). Биты используются для представления различных состояний и данных.

В повседневной жизни мы привыкли к **десятичной системе счислени**я, но компьютер оперирует д**воичной системой**. Это означает, что числа представляются с использованием только 0 и 1.

$$\begin{matrix} 7\ 6\ 5\ 4\ 3\ 2\ 1\ 0 \\ 10001001_2 = 1\cdot2^7+0\cdot2^6+0\cdot2^5+0\cdot2^4+1\cdot2^3+0\cdot2^2+0\cdot2^1+1\cdot2^0 = 137_{10} \end{matrix}$$

Для того, чтобы представить десятичное число в двоичной системе счисления, можно воспользоваться вычислением остатков от деления на 2 и “прочесть” полученные значения снизу вверх

137 / 2 = 68 **остаток: 1**
68 / 2 = 34 **остаток: 0**
34 / 2 = 17 **остаток: 0**
17 / 2 = 8 **остаток: 1**
8 / 2 = 4 **остаток: 0**
4 / 2 = 2 **остаток: 0**
2 / 2 = 1 **остаток: 0**
1 / 2 = 0 **остаток: 1**

итог: 10001001_2

Группа из 8 бит называется **байтом**. Байты служат "кирпичиками" информации и используются для представления чисел, символов и других данных. Максимальное число, которое можно представить 1 байтом, - 255

Операции сложения и вычитания для двоичной системы счисления аналогичны этим операциям в десятичной системе счисления:

$0_2 + 0_2 = 0_2 = 0_{10}$

$0_2 + 1_2 = 1_2 = 1_{10}$

$1_2 + 1_2 = 10_2 = 2_{10}$

$1_2 + 1_2 + 1_2 = 11_2 = 3_{10}$

Сложение “столбиком”:

$$\begin{array}{r} {\scriptstyle 1\,1\quad 1\,1\,1} \\ 11100111 \\ +\ 11100111 \\ \hline 111001110 \end{array}$$

Прямой код - это обычное бинарное представление числа. В прямом коде старший бит несет весовой знак числа. Например, для положительного числа 5 в 4-битном прямом коде: **0101**.

Обратный код получается инвертированием всех битов прямого кода. Это значит, что 0 заменяется на 1, а 1 - на 0. Например, для числа 5 его обратный код будет: **1010**.

Дополнительный код получается добавлением 1 к обратному коду числа. Таким образом, дополнительный код - это обратный код с добавленной единицей. Например, для числа 5 его дополнительный код: **1011**.

Представление -5_{10}

$5_{10} = 00000101_2$ (прямой код)

обратный код: 11111010_2
дополнительный код: 11111011_2

В системе дополнительного кода разряд числа, отвечающий за знак, обычно называется **"старшим разрядом"** или **"знаковым битом"**. Этот бит указывает на знак числа: 0 обозначает положительное число, а 1 — отрицательное.

В 8-битной системе дополнительного кода, например, старший бит (самый левый бит) отвечает за знак числа. Если старший бит равен 0, это положительное число, и значение остальных битов следует интерпретировать как модуль этого числа. Если старший бит равен 1, это отрицательное число, и значение остальных битов представляет собой дополнение к обратному коду положительного числа.

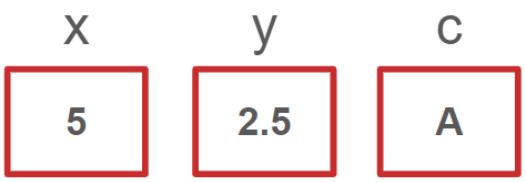
$$11111011_2$$

Переменные

Переменная в программировании представляет собой именованный участок памяти, который можно использовать для хранения и обработки данных. Для того чтобы воспользоваться переменной, необходимо явно объявить ее, указав имя и тип данных, которые она будет хранить. Каждая переменная содержит данные определенного типа, такие как целые числа, вещественные числа, символы и т.д.

Чтобы использовать переменную, первым шагом является ее **объявление**, которое предполагает указание типа данных и выбор уникального имени. Например, `int age;` объявляет переменную с именем `age`, которая будет содержать целочисленные значения. Затем переменной можно **присвоить** конкретное значение, соответствующее ее типу, например, `age = 25;`. Можно также объявить и присвоить значение одновременно: `int count = 10;`

Тип данных переменной важен для правильной интерпретации и использования данных. Например, переменная типа `double` может хранить вещественные числа с плавающей запятой, в то время как переменная типа `char` будет предназначена для символьных данных. Важно соблюдать соответствие типа данных и значения, чтобы избежать ошибок при выполнении программы.



int

В языке программирования Java тип данных **int** представляет собой целочисленный тип, который используется для хранения целых чисел. Тип int занимает **4 байта** в памяти компьютера, обеспечивая диапазон значений от **-2147483648** до **2147483647**

Над переменными типа int можно выполнять основные арифметические операции, такие как сложение, вычитание, умножение и деление.

Операция взятия остатка (**%**) позволяет получить остаток от деления двух целых чисел. Например, `13 % 3` даст в результате 1, так как 13 деленное на 3 равно 4 и остаток равен 1. Операция целочисленного деления (**/**) возвращает только целую часть от результата деления, отбрасывая дробную часть. Например, `13 / 3` возвращает 4, так как результат деления 13 на 3 равен 3.25, но целочисленное деление вернет только 3.

Тип данных int является одним из наиболее часто используемых целочисленных типов в Java, обеспечивая достаточный диапазон значений для большинства приложений. При использовании этого типа данных важно учитывать его ограничения, особенности операций и выбирать его в зависимости от требуемого диапазона значений.

double

В языке программирования Java тип данных **double** представляет собой числовой тип с плавающей точкой, используемый для хранения вещественных чисел. Этот тип данных занимает **8 байт** в памяти компьютера, что обеспечивает расширенный диапазон значений по сравнению с целочисленными типами данных.

double подходит для работы с очень маленькими и очень большими числами, предоставляя большой диапазон значений. Минимальное положительное - **$4.9 * 10^{-324}$**, максимальное положительное значение - **$1.7976931348623157 * 10^{308}$**. Вы можете выполнять все стандартные арифметические операции над переменными типа double, включая сложение, вычитание, умножение и деление.

Пример присвоения значения переменной типа double: `double pi = 3.14159;`.

Тип данных double предоставляет точное представление вещественных чисел и широко используется в программировании, особенно там, где требуется высокая точность при

работе с десятичными дробями или большими числами. При использовании double важно учесть возможные потери точности из-за представления чисел с плавающей точкой.

char

В языке программирования Java, тип данных char используется для хранения одного символа. Тип char занимает 2 байта в памяти компьютера, что позволяет представлять символы с использованием стандарта Unicode. Unicode – это международный стандарт, который позволяет компьютерам представлять и обрабатывать текст на большинстве письменных языков мира, поддерживая огромное количество символов и специальных знаков.

Каждый символ в Unicode имеет уникальный код, называемый кодовой точкой. Например, символ 'A' имеет кодовую точку U+0041 (65 - в десятичной системе счисления), а русская буква 'Я' – U+042F (1071 - в десятичной системе счисления). Тип данных char в Java может представлять любой символ из стандарта Unicode, используя эти кодовые точки.

Пример присвоения значения переменной типа char: `char letter = 'A';`

Тип char широко используется в программировании для работы с отдельными символами, строками текста и при выполнении различных операций с текстовыми данными. Например, он может быть использован для проверки, является ли символ буквой, цифрой, знаком пунктуации и т.д. Важно помнить, что char в Java представляет собой беззнаковый тип, и его значения находятся в диапазоне от **0** до **65,535**, что соответствует диапазону кодовых точек Unicode.

Примитивные типы данных

Примитивные типы данных в Java – это встроенные типы данных, которые определены в самом языке программирования. Они используются для хранения простых значений, таких как числа, символы и логические значения. В Java существует восемь примитивных типов данных:

byte: Целочисленный тип, занимает 1 байт памяти. Хранит значения от -128 до 127.

short: Целочисленный тип, занимает 2 байта памяти. Хранит значения от -32,768 до 32,767.

int: Стандартный целочисленный тип, занимает 4 байта памяти. Хранит значения от -2,147,483,648 до 2,147,483,647.

long: Большой целочисленный тип, занимает 8 байт памяти. Хранит значения от -9,223,372,036,854,775,808 до 9,223,372,036,854,775,807.

float: Тип с плавающей точкой, занимает 4 байта памяти. Используется для представления вещественных чисел с меньшей точностью.

double: Тип с плавающей точкой, занимает 8 байт памяти. Используется для представления вещественных чисел с большей точностью.

char: Символьный тип, занимает 2 байта памяти. Используется для хранения отдельных символов (в Unicode).

boolean: Логический тип, занимает 1 бит (но в реализации может занимать больше из-за выравнивания памяти). Используется для хранения логических значений true (истина) и false (ложь).

Эти типы данных являются основой для работы с числовыми, символьными и логическими значениями в Java. Они отличаются от ссылочных типов данных, таких как объекты и массивы, тем, что хранят данные непосредственно, а не ссылки на объекты в памяти.

Литеральные значения

Литеральные значения – это фиксированные значения, которые непосредственно встроены в код программы. Они используются для представления постоянных значений различных типов данных. В Java литералы могут быть представлены для различных типов данных:

Целочисленные литералы: Могут быть представлены в десятичной (базовая 10), шестнадцатеричной (базовая 16, например 0x1A), восьмеричной (базовая 8, начинается с 0, например 034) и двоичной (базовая 2, начинается с 0b или 0B, например 0b1010) системах счисления.

Литералы с плавающей точкой: Представляют вещественные числа, могут быть записаны в обычном (например, 0.75) или научном (например, 1.5e-4) форматах.

Символьные литералы: Представляют символы Unicode и заключаются в одинарные кавычки, например 'a', '\u0041' (код символа 'A').

Строковые литералы: Последовательность символов, заключенных в двойные кавычки, например "Hello World".

Логические литералы: true и false для представления логических значений.

Переполнение

Переполнение происходит, когда значение, присваиваемое переменной, выходит за пределы её допустимого диапазона значений. Это часто происходит с целочисленными типами данных из-за их ограниченного размера:

В случае **целочисленного** переполнения, если значение выходит за верхний предел, оно "оборачивается" к началу диапазона. Например, для типа byte после значения 127 следует -128. Это происходит происходит из-за способа представления чисел в двоичной форме и ограниченного количества битов, выделенных для хранения этих чисел.

Рассмотрим это на примере:

Двоичное представление: В компьютерах все данные, включая числа, представлены в виде двоичных чисел (наборов 0 и 1). Количество битов, выделенных для каждого типа данных, определяет максимальное число, которое может быть представлено.

Ограниченное количество битов: Когда значение превышает максимально возможное для данного количества битов, происходит переполнение. Например, для 8-битного целого числа максимальное значение – 255 (в двоичной системе 11111111). При добавлении 1 к этому числу, двоичное значение должно было бы стать 100000000, но из-за ограничения в 8 битов, старший бит (9-й бит) "отбрасывается", и значение становится 00000000 (то есть 0).

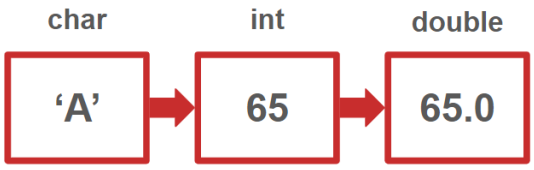
Циклическое переполнение: Этот процесс можно представить как циклическое возвращение к началу диапазона. В примере с 8-битным числом, после достижения 255 (максимального значения), счет продолжается с начала диапазона, то есть с 0.

В случае со знаковыми целыми числами, такими как int или byte в Java, принцип аналогичен, но в дополнение к значениям, которые могут быть представлены, учитывается также знак числа (положительный или отрицательный), что влияет на интерпретацию двоичных данных.

Это поведение является фундаментальным для двоичных систем и присутствует во всех языках программирования и компьютерных архитектурах, где используются ограниченные по размеру типы данных.

Понимание переполнения важно для предотвращения неожиданных ошибок в вычислениях, особенно в случаях, когда работа ведется с большими числами или в условиях, где точность вычислений критична. В Java необходимо тщательно управлять размерами данных и типами переменных, чтобы избежать таких проблем.

Неявное преобразование типов



Неявное преобразование типов в Java - это процесс, при котором компилятор автоматически преобразует один тип данных в другой. Это происходит, когда операнды в выражении имеют разные типы данных. Примером такого преобразования является преобразование из **char** в **int** и в **double**.

Когда переменная типа `char` используется в контексте, требующем `int` (например, в арифметической операции), происходит неявное преобразование типа. В Java тип `char` представляет символ в кодировке Unicode и занимает 2 байта. Каждый символ `char` имеет соответствующее целочисленное значение в Unicode.

```
char ch = 'A'; // Unicode для 'A' - это 65
int num = ch;  // неявное преобразование char в int
System.out.println(num); // Выведет 65
```

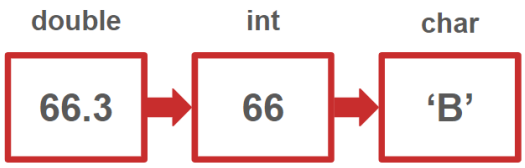
В этом примере, когда мы присваиваем значение типа `char` переменной типа `int`, символ `'A'` автоматически преобразуется в его целочисленный эквивалент в Unicode, который равен 65.

В случае преобразования из `int` в `double`, `int` - это 32-битный целочисленный тип данных, а `double` - это 64-битный тип данных с плавающей точкой.

Поскольку `double` может содержать все значения `int` и дополнительно вещественные числа, преобразование происходит автоматически без потери информации. При этом целочисленное значение 65 преобразуется в значение с плавающей точкой 65.0.

Неявное преобразование в таких случаях удобно, поскольку избавляет программиста от необходимости писать дополнительный код для явного приведения типов.

Явное преобразование типов



Явное преобразование типов данных в Java, также известное как приведение типов, происходит, когда программист явно указывает конвертацию из одного типа данных в другой. Это особенно важно, когда преобразуемый тип имеет больший размер или диапазон значений, чем целевой тип, или когда изменяется тип данных (например, с числового на символьный).

При преобразовании из `double` в `int` происходит усечение десятичной части числа. То есть, число округляется до ближайшего целого вниз.

```
double d = 66.3;
int i = (int) d; // Явное преобразование из double в int
System.out.println(i); // Выведет 66
```

Далее, преобразуем целочисленное значение `int` в символьный тип `char`. В Java символы char представлены их числовыми значениями в таблице Unicode.

```
int i = 66;
char c = (char) i; // Явное преобразование из int в char
System.out.println(c); // Выведет 'B'
```

В Unicode символу 'B' соответствует числовое значение 66. Таким образом, при преобразовании числа 66 в тип char мы получаем символ 'B'.

Оба шага в вашем примере демонстрируют, как явное преобразование типов используется для конвертации значений между различными типами данных в Java. Это важная часть языка программирования, позволяющая контролировать и управлять данными разных типов. Явное указание типов при приведении требует от программиста осознанного решения о таком преобразовании, тем самым снижая риск непреднамеренной потери информации.