Программирование на языке Java

6. Форматный вывод

Программирование на языке Java

Тема 6. Форматный вывод

Сложение двух чисел

Задача. Ввести два целых числа и вывести на экран их сумму.

Простейшее решение:

```
int a, b, c;
a = in.nextInt();
b = in.nextInt();
c = a + b;
System.out.print(a+"+"+b+"="+c);
```

Форматный вывод

Форматный вывод – вывод в различные потоки значений **разных типов**, отформатированных согласно заданному формату (шаблону).

Формат определяется составленной по специальным правилам строкой.

Форматный вывод в Java

```
System.out.printf(<форматная строка>,
                  <список аргументов>);
```

Форматная строка – символьная строка, которая задает шаблон вывода аргументов.

Форматная строка состоит из:

- символов, которые копируются в выходной поток;
- спецификаторов формата, определяющих способ, в соответствии с которым должны отображаться последующие аргументы.

Спецификатор формата

Спецификатор формата начинается со знака процента с последующим спецификатором преобразования.

Пример. Спецификатор формата для десятичного целого числа – **%d**.

Форматный вывод. Пример

printf – форматный вывод

Форматная строка

Целое число подставляется из переменной a

```
System.out.printf("%d", a);
```

a – имя переменной

```
a
```

```
12
```

12 – значение переменной **a**

Спецификаторы формата – 1

Спецификатор	Применяемое преобразование
`%a, %A`	Шестнадцатеричное с плавающей точкой
`%b, %B`	Булевское
`%c`	Символ
`%d`	Десятичное целое число
`%e, %E`	Научная нотация
`%f`	Десятичное с плавающей точкой
`%g, %G`	Либо %e, либо %f, смотря что короче
`%o`	Восьмеричное целое
`%n`	Символ перевода строки
`%s, %S`	Строка
`%x, %X`	Шестнадцатеричное целое
`%%`	Вставляет символ %

Спецификаторы формата – 2

Некоторые спецификаторы имеют заглавную и прописную формы. При использовании заглавной формы буквы отображаются в верхнем регистре.

Форматирование целых чисел – 1

вывести целое число и перевод строки

это число взять из переменной **c**

```
System.out.printf ("%d%n", c);
```

```
System.out.printf ("Результат: %d",
c);
```

```
System.out.printf("%d+%d=%d%n", a, b, c
);
```

форматная строка

список аргументов

```
System.out.printf("%d+%d=%d%n",a, b,
a+b);
```

арифметическое выражение

Форматирование целых чисел – 2

```
System.out.printf ("16-ая с/с %x", 196);
```

```
16-ая с/с c4
```

```
System.out.printf ("16-ая с/с %X", 196);
```

```
16-ая с/с C4
```

```
System.out.printf ("8-ая с/с %o", 196);
```

```
8-ая с/с 304
```

Форматирование вещественных чисел

```
double x = 12345.6789;
System.out.printf ("%f",
x);
```

```
12345,678900
```

минимальное число позиций, **6 цифр** в дробной части

```
System.out.printf ("%e",
x);
```

```
1.234568e+04
```

Научная нотация: $1{,}23456 \cdot 10^4$

```
System.out.printf ("%E",
x);
```

```
1.234568E+04
```

Спецификаторы %n и %%

Отличаются от других тем, что они не соответствуют аргументу. Представляют собой управляющие последовательности:

%n – вставляет перевод строки

%% – вставляет знак процента.

```
System.out.printf ("Копирование файла
%nПеремещение на %d%% завершено",
88);
```

```
Копирование файла
Перемещение на 88% завершено
```

Указание минимальной ширины поля – 1

Спецификатор минимальной ширины – целое число, помещенное между символом `%` и кодом преобразования формата.

Спецификатор минимальной ширины дополняет вывод пробелами, обеспечивая заданную минимальную длину.

Если строка или число получаются длиннее, чем заданный минимум, то число выводится полностью.

По умолчанию, дополнение осуществляется пробелами.

Указание минимальной ширины поля – 2

```
int x = 1234;
System.out.printf ("%d\n", x);
```

```
1234
```

минимальное число позиций

```
System.out.printf ("%9d\n",
x);
```

```
     1234
```

всего 9 позиций

Указание минимальной ширины поля – 3

Чтобы дополнить число лидирующими нулями, нужно поместить **0** перед спецификатором ширины поля.

```
System.out.printf ("%09d\n",
x);
```

```
000001234
```

всего 9 позиций, пустые заполнены нулями

Указание минимальной ширины поля – 4

```
double x = 10.12345;
System.out.printf("|%f|%n|%12f|%n|%012f|",
x, x, x);
```

```
|10,123450|
|   10,123450|
|00010,123450|
```

Указание точности – 1

Спецификатор точности может быть применен к спецификаторам формата **`%f`**, **`%e`**, **`%g`** и **`%s`**.

Спецификатор точности следует за спецификатором минимальной ширины поля (если таковой имеется) и состоит из точки с последующим целым числом.

Указание точности – 2

Спецификатор точности для данных с плавающей точкой (**%f** или **%e**) определяет количество отображаемых десятичных разрядов.

%10.4f – число в 10 символов шириной с 4 разрядами после запятой.

При использовании **%g** точность определяется количеством значащих десятичных разрядов.

Точность по умолчанию – 6 знаков после запятой.

Указание точности – 3

```
double x = 12345.6789;
System.out.printf ("%10.3f", x);
```

```
 12345,679
```

всего 10 позиций, **3 цифры** в дробной части

```
System.out.printf ("%10.2e", x);
```

```
  1.23e+04
```

всего 10 позиций, **2 цифры** в дробной части мантиссы

Вопрос. Как вывести `00012345,68`

```
System.out.printf ("%011.2f", x);
```

Указание точности – 4

Для строк спецификатор точности задает максимальную ширину поля. Если строка длиннее максимальной ширины, конечные символы усекаются.

```
System.out.printf ("%.15s%n",
"Форматировать в Java очень просто");
```

```
Форматировать в
```

15 символов

Флаги формата – 1

Флаги формата позволяют управлять различными аспектами преобразования.

Все флаги формата – одиночные символы, которые следуют за знаком `%` в спецификаторе формата.

Флаги формата – 2

Флаг	Эффект
-	Выравнивание влево
0	Вывод дополняется нулями вместо пробелов
Пробел	Положительным числам предшествует пробел
+	Положительным числам предшествует знак +
,	Числовые значения, включающие групповые разделители
(	Отрицательные числовые значения заключены в скобки

Внимание! Не все флаги применимы ко всем спецификаторам формата.

Выравнивание вывода

По умолчанию весь вывод выравнивается вправо.

Для выравнивания по левому краю, нужно поставить знак минус после сразу после %.

```
System.out.printf ("|%10.2f|%n",123.123);
System.out.printf ("|%-10.2f|%n",123.123);
```

```
|    123,12|
|123,12    |
```

Флаги пробела, +, 0 и (– 1

Данные флаги работаю со знаком числа:

```
System.out.printf ("%d%n",100);
System.out.printf ("% d%n",100);
System.out.printf ("%+d%n",100);
System.out.printf ("%05d%n",100);
System.out.printf ("%(d%n",100);
```

```
100
 100
+100
00100
100
```

Флаги пробела, +, 0 и (– 2

Данные флаги работаю со знаком числа:

```
System.out.printf ("%d%n",-100);
System.out.printf ("% d%n",-100);
System.out.printf ("%+d%n",-100);
System.out.printf ("%05d%n",-100);
System.out.printf ("%(d%n",-100);
```

```
-100
-100
-100
-0100
(100)
```

Флаг запятая

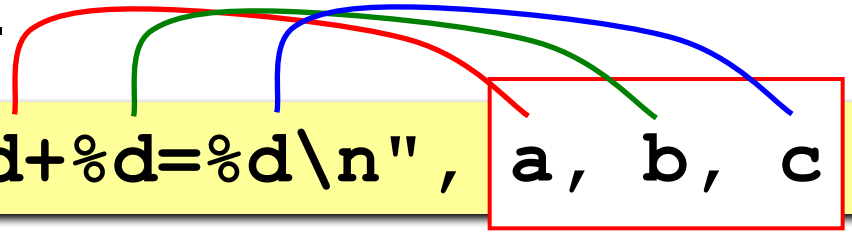
При отображении больших чисел удобно использовать разделители групп. Например 1234567 читается легче в виде 1 234 567. Для добавления спецификаторов группирования служит флаг запятая.

```
System.out.printf ("%,.2f", 4356783497.34);
```

```
4 356 783 497,34
```

Использование индекса аргументов – 1

Обычно порядок аргументов и спецификаторов совпадает (слева направо), т.е. первый спецификатор относится к первому аргументу, второй – ко второму и т.д.



```
System.out.printf("%d+%d=%d\n", a, b, c
);
```

Используя индексы аргументов, можно управлять тем, к какому из аргументов относится спецификатор формата.

Использование индекса аргументов – 2

Индекс аргумента следует за `%` в спецификаторе формата и имеет вид **`n$`**, где **`n`** – индекс нужного аргумента, начиная с 1.

```
System.out.printf("%3$d %1$d %2$d",
                  10, 20, 30);
```

```
30 10 20
```

Использование индекса аргументов – 3

Преимущество индексирования аргументов: повторное использование аргумента.

```
System.out.printf("%1$d в шестнадцатеричном
        формате это %1$X%n", 255);
```

```
255 в шестнадцатеричном формате это FF
```

Полное решение

Задача. Ввести два целых числа и вывести на экран их сумму.

```
int a, b, c;
  a = in.nextInt();
  b = in.nextInt();
  c = a + b;
  System.out.printf("%d+%d=%d", a, b, c);
}
```