

Структуры и алгоритмы компьютерной обработки данных

Типы данных

Типы данных языка Java

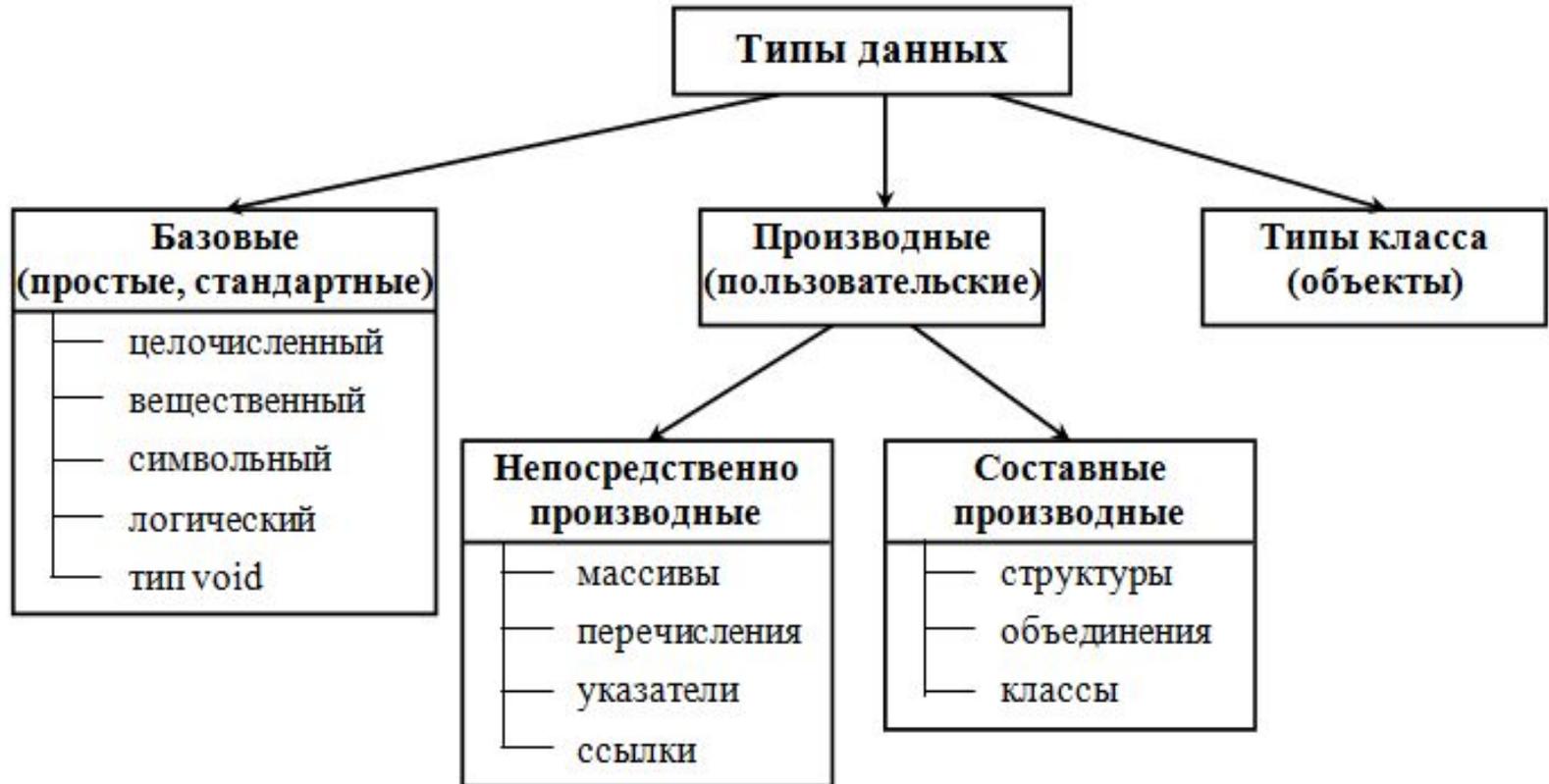
Простые (примитивные)

- **целочисленные**
 - **byte**
 - **short**
 - **int**
 - **long**
 - **char**
- **дробные**
 - **float**
 - **double**
- **булево**
 - **boolean**

Объектные (Ссылочные)

Классы
Интерфейсы
Массивы

Типы данных языка C++



Переменные используются в программе для хранения данных.

Любая переменная имеет три базовых характеристики:

имя;

тип;

значение.

Имя уникально идентифицирует переменную и позволяет обращаться к ней в программе.

Тип описывает, какие величины может хранить переменная.

Значение – текущая величина, хранящаяся в переменной на данный момент.

Некоторые примеры объявления переменных примитивного типа int :

```
int a;
```

```
int b = 0, c = 3+2;
```

```
int d = b+c;
```

```
int e = a = 5;
```

Таблица 4.1. Целочисленные типы данных.

Название типа	Длина (байты)	Область значений
byte	1	-128 .. 127
short	2	-32.768 .. 32.767
int	4	-2.147.483.648 .. 2.147.483.647
long	8	-9.223.372.036.854.775.808 .. 9.223.372.036.854.775.807 (примерно 10^{19})
char	2	'\u0000' .. '\uffff', или 0 .. 65.535

Над целочисленными аргументами можно производить следующие операции:

- ❖ операции сравнения (возвращают булево значение)

<, <=, >, >=

==, !=

- ❖ числовые операции (возвращают числовое значение)

все числовые операторы возвращают результат типа int или long!

унарные операции + и -

арифметические операции +, -, *, /, %

операции инкремента и декремента (в префиксной и постфиксной форме): ++ и --

операции битового сдвига <<, >>, >>>

битовые операции ~, &, |, ^

- ❖ оператор с условием ? :

- ❖ оператор приведения типов

- ❖ оператор конкатенации со строкой +

Операторы + и - могут быть как бинарными (иметь два операнда), так и унарными (иметь один операнд).

Бинарные операнды являются операторами сложения и вычитания, соответственно.

Унарный оператор + возвращает значение, равное аргументу (+x всегда равно x).

Унарный оператор -, примененный к значению x, возвращает результат, равный 0-x.

Неожиданный эффект имеет место в том случае, если аргумент равен наименьшему возможному значению примитивного типа.

```
int x=-2147483648; // наименьшее возможное значение типа int
int y=-x;
```

Теперь значение переменной y на самом деле равно не 2147483648, поскольку такое число не укладывается в область значений типа int, а в точности равно значению x!

Другими словами, в этом примере выражение $-x==x$ истинно!

Рассмотрим пример:

```
int i=300000;  
print(i*i); // умножение с точностью 32 бита  
long m=i;  
print(m*m); // умножение с точностью 64 бита  
print(1/(m-i)); // попробуем получить разность значений int и long
```

Результатом такого примера будет:

-194313216

900000000000

Время в Java измеряется в миллисекундах.

Попробуем вычислить, сколько миллисекунд содержится в неделе и в месяце:

```
print(1000*60*60*24*7);    // вычисление для недели  
print(1000*60*60*24*30);  // вычисление для месяца
```

Необходимо перемножить количество миллисекунд в одной секунде (1000), секунд – в минуте (60), минут – в часе (60), часов – в дне (24) и дней – в неделе и месяце (7 и 30, соответственно).

Получаем:

```
604800000  
-1702967296
```

Очевидно, во втором вычислении произошло переполнение. Достаточно сделать последний аргумент величиной типа long:

```
print(1000*60*60*24*30L);  // вычисление для месяца
```

Получаем правильный результат:

```
2592000000
```

Операторы инкрементации и декрементации.

```
byte x=5;
byte y1=x++; // на момент начала исполнения x равен 5
byte y2=x--; // на момент начала исполнения x равен 6
byte y3=++x; // на момент начала исполнения x равен 5
byte y4=--x; // на момент начала исполнения x равен 6
print(y1);
print(y2);
print(y3);
print(y4);
```

В результате получаем:

5
6
6
5

Оператор с условием ? :

Если второй и третий операнды имеют одинаковый тип, то и результат операции будет такого же типа.

```
byte x=2;
```

```
byte y=3;
```

```
byte z=(x>y) ? x : y;    // верно, x и y одинакового типа
```

```
byte abs=(x>0) ? x : -x; // неверно!
```

Последняя строка неверна, так как третий операнд содержит числовую операцию, стало быть, его тип `int`, а значит, и тип всей операции будет `int`, и присвоение некорректно. Даже если второй аргумент имеет тип `byte`, а третий – `short`, значение будет типа `int`.

Оператор конкатенации со строкой +

Оператор + может принимать в качестве аргумента строковые величины. Если одним из аргументов является строка, а вторым – целое число, то число будет преобразовано в текст и строки объединятся.

```
int x=1;
```

```
print("x="+x);
```

Результатом будет:

```
x=1
```

Обратите внимание на следующий пример:

```
print(1+2+"text");
```

```
print("text"+1+2);
```

Его результатом будет:

```
3text
```

```
text12
```

Рассмотрим работу с типом char.

Значения этого типа могут полноценно участвовать в числовых операциях:

```
char c1=10;  
char c2='A'; // латинская буква A (\u0041, код 65)  
int i=c1+c2-'B';
```

Переменная *i* получит значение 9.

Рассмотрим следующий пример:

```
char c='A';  
print(c);  
print(c+1);  
print("c="+c);  
print('c'+'+'+c);
```

Его результатом будет:

```
A  
66  
c=A  
225
```

Дробные типы

Таблица 4.2. Дробные типы данных.

Название типа	Длина (байты)	Область значений
float	4	3.40282347e+38f ; 1.40239846e-45f
double	8	1.79769313486231570e+308 ; 4.94065645841246544e-324