IT ШКОЛА SAMSUNG

Тема: *Представление данных в памяти. Поразрядные и логические операции*

Дмитрий Сергеевич Егоров

Представление целочисленных типов данных

Данные примитивных типов представляются фиксированным количеством двоичных разрядов

Название	Длина (байт/бит)	Область значений
byte	1 (8)	-128 .. 127
short	2 (16)	-32.768 .. 32.767
int	4 (32)	-2.147.483.648 .. 2.147.483.647
long	8 (64)	-9.223.372.036.854.775.808 .. 9.223.372.036.854.775.807 (примерно 10^{19})
char	2 (16)	'\u0000' .. '\uffff', или 0 .. 65.535

Представление отрицательных целых чисел

Нельзя использовать знак «минус» для представления отрицательного числа в памяти. Отрицательные числа должны состоять **только из нулей и единиц**.

???

Представление отрицательных целых чисел

Нельзя использовать знак «минус» для представления отрицательного числа в памяти. Отрицательные числа должны состоять **только из нулей и единиц**.

???

Отведем старший разряд под знак:

1 – «-»,

0 – «+»

Представление отрицательных целых чисел

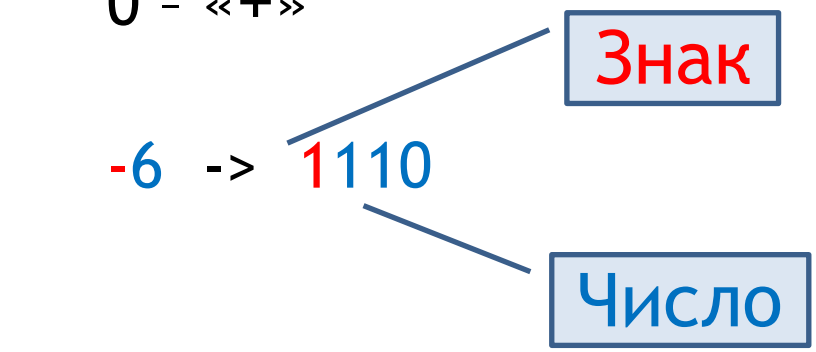
Нельзя использовать знак «минус» для представления отрицательного числа в памяти. Отрицательные числа должны состоять **только из нулей и единиц**.

???

Отведем старший разряд под знак:

1 – «-»,

0 – «+»



это **прямой** код числа

Прямой код - недостатки

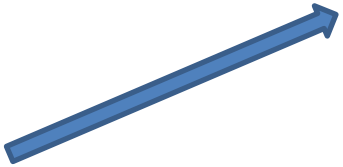
Сложно складывать отрицательные числа и числа разных знаков.

```
  01011          11
+              +
  10010          -2
-----------    -----------
  11101          -13 !!!
```

Прямой код - недостатки

Сложно складывать отрицательные числа и числа разных знаков.

```
  01011          11
+              +
  10010          -2
-----------    -----------
  11101          -13 !!!
```



???

Представление числа в так называемом **дополнительном** коде.

Дополнительный код числа

Для получения дополнительного кода отрицательного целого числа:

1) Находим прямой код
2) Инвертируем прямой код (меняем 0 на 1, а 1 на 0)
3) К инверсному коду прибавляем 1.

Для положительных чисел:

прямой код = дополнительный код

Дополнительный код числа

Получим дополнительный код для числа -4.

1) $4 = 0100_2$ - прямой код

Дополнительный код числа

Получим дополнительный код для числа -4.

1) $4 = 0100_2$ – прямой код

2) 1011_2 – инверсный код

Дополнительный код числа

Получим дополнительный код для числа -4.

1) $4 = 0100_2$ – прямой код

2) 1011_2 – инверсный код

3) 1011_2
+
1

1100 - дополнительный код

Дополнительный код числа

Получим дополнительный код для числа -4.

1) $4 = 0100_2$ – прямой код

2) 1011_2 – инверсный код

$1011_2 + 0100_2 = 1111_2$

3) 1011_2

\+

1

1100

- дополнительный код

Дополнительный код числа

Получим дополнительный код для числа -4.

1) $4 = 0100_2$ – прямой код

2) 1011_2 – инверсный код

3) 1011_2
+
1

1100 - дополнительный код

$1011_2 + 0100_2 = 1111_2$

Инверсный код **дополняет** прямой до заполнения количества разрядов единицами

Дополнительный код числа Задание

Получите дополнительный код для числа -12.

Какое число представлено **дополнительным** кодом: 1001_2?

!!!Все целочисленные типы Java хранят числа в дополнительном коде!!!

Дополнительный код числа. Эффект фиксированной разрядной сетки при переполнении

```
  0110
+
  0100
-----------
  1010
```

Пусть для хранения числа есть 4 бита.

Дополнительный код числа. Эффект фиксированной разрядной сетки при переполнении

$$
\begin{array}{r}
0110_2 \\
+ \\
0100_2 \\
\hline
1010_2
\end{array}
\qquad \Rightarrow \qquad
\begin{array}{r}
6 \\
+ \\
4 \\
\hline
-6
\end{array}
$$

Шестнадцатеричная система счисления

Чаще, чем двоичная используется **шестнадцатеричная** система счисления

Как быстро перевести число из двоичной системы в шестнадцатеричную?

Шестнадцатеричная система счисления

Чаще, чем двоичная используется **шестнадцатеричная** система счисления

Как быстро перевести число из двоичной системы в шестнадцатеричную?

0_{16}	1_{16}	2_{16}	3_{16}	4_{16}	5_{16}	6_{16}	7_{16}	8_{16}	9_{16}	A_{16}	B_{16}	C_{16}	D_{16}	E_{16}	F_{16}
0000_2	0001_2	0010_2	0011_2	0100_2	0101_2	0110_2	0111_2	1000_2	1001_2	1010_2	1011_2	1100_2	1101_2	1110_2	1111_2

Шестнадцатеричная система счисления

Пример:

100111100010_2

Шестнадцатеричная система счисления

Пример:

100111100010_2

$1001\ 1110\ 0010_2$

$9 \quad E \quad 2_{16}$

Самостоятельно: 1100001011110011_2

Самостоятельно

1. Сколько единиц содержится в двоичной записи года вашего рождения? Как год рождения записать в шестнадцатеричной системе счисления?

Задание

Предложите алгоритм вычитания одного числа из другого без операции «минус». Например: 5-3.

???

Задание

Предложите алгоритм вычитания одного числа из другого без операции «минус». Например: 5-3.

???

5+(-3); -3 представляется дополнительным кодом числа. Следовательно:

1. ***Инвертируем двоичную запись вычитаемого***
2. Прибавляем к инверсному коду 1.
3. Складываем полученный результат с уменьшаемым

Поразрядные и логические операции

Побитовое отрицание(NOT)

Построение инверсного кода - это и есть побитовое отрицание NOT, т.е. замена единиц на нули, а нулей на единицы в двоичном коде числа

В Java NOT обозначается ~ (тильда).
Например:
~a;

~ 00000000 00000000 00000000 00001011
=
11111111 11111111 11111111 11110100

Побитовое И (AND)

Результат в бите равен 1, если соответствующие биты первой и второй последовательностей тоже 1.

В Java AND обозначается **&** (амперсант).
Например:
c = a & b;

a	b	a & b
1	1	1
1	0	0
0	1	0
0	0	0

00000000 00000000 00000000 00001101 (11)
&
00000000 00000000 00000000 00010101 (21)
=
00000000 00000000 00000000 00000101 (5)

Зачем используется

Байт – это минимальная единица для доступа к памяти => нет операций доступа к отдельным битам.
Поразрядные операции позволяют решить эту задачу.
Для **AND**:
Если взять подходящее второе число (маску), то байт-результат даст информацию о конкретном бите первого числа.

Например, какое значение содержится в третьем бите числа:

```
  xxxx?xxx
&
  00001000
=
  0000?000
```

Если искомый бит (?) нулевой, то в байте-результате будут все нули, то есть ? = 0, если нет – то результат будет 4, то есть ? = 1.

Побитовое ИЛИ (OR)

Результат в бите равен 1, если хотя бы один из соответствующих бит первой и второй последовательностей тоже 1.

В Java AND обозначается |.
Например:
c = a | b;

a	b	a | b
1	1	1
1	0	1
0	1	1
0	0	0

00000000 00000000 00000000 00001101 (11)
|
00000000 00000000 00000000 00010101 (21)
=
00000000 00000000 00000000 00011101 (29)

Зачем используется OR

Позволяет гарантированно установить заданный бит числа в 1.

Например, установим 1 в третьем бите числа:

```
  xxxx?xxx
|
  00001000
=
  00001000
```

Значение в третьем бите точно будет 1

Благодаря маскам можно хранить в одной переменной несколько двоичных значений (например, истина/ложь).

Исключающее ИЛИ (XOR)

Результат в бите равен 1, если значения соответствующих бит первой и второй последовательностей совпадают.

В Java XOR обозначается ^.
Например:
c = a ^ b;

a	b	a ^ b
1	1	0
1	0	1
0	1	1
0	0	0

00000000 00000000 00000000 00001101 (11)
^
00000000 00000000 00000000 00010101 (21)
=
00000000 00000000 00000000 00011000 (24)

Зачем используется XOR

Самое известное применение – шифрование.
Если применить эту операцию к значению с каким-то ключом (другим значением) первое изменится, но при повторном применении восстанавливается.

Например: пусть ключом будет число 5 (0…0101)
Зашифруем число 8:
0..01000 ^ 0..00101 = 0..01101 (13)
0..01101 ^ 0..00101 = 0..01000 (8)

Если рассматривать символы как числовые коды, то можно шифровать тексты

Операции сдвига

Знаковый оператор сдвига влево <<

Все биты смещаются влево. Число справа дополняется нулем.

Например:

a = 0..00110

a<<;

a= 0..01100

Знаковый оператор сдвига влево <<

Все биты смещаются влево. Число справа дополняется нулем.

Например:
a = 0..00110 (6)
a<<;
a= 0..01100 (12)

Быстрое умножение на 2!!!

Задача

Пусть число имеет тип byte (8 бит).

Чему равен результат применения операции << к числу 100?

Знаковый оператор сдвига вправо >>

Все биты смещаются вправо. Число слева дополняется нулем, если число положительное и единицей, если отрицательное.

Например:
a = 0..00110 (6)
a>>;
a= 0..00011 (3)

Быстрое деление на 2!!!

Например:
a = 1..10101 (-11)
a>>;
a= 1..11010 (-6)

Зачем используются операции сдвига

Операции сдвига можно применять для умножения или деления нацело на степень двойки.

Например:

5 << 2 = 20 (соответствует $5 * 2^2$)

42 >> 3 = 5 (соответствует $42 / 2^3$)

Операции сдвига выполняются значительно быстрее обычного умножения и деления.

Задание

Решить задачи http://informatics.msk.ru

Группа 1: №№121, 123,

Группа 2: №№124, 125

Дополнительно: №127

Логические операции. Тип boolean

Операторы сравнения

> строго больше	<= меньше или равно
< строго меньше	== равно
>= больше или равно	!= не равно

Операторы сравнения возвращают true/false

Например:

```
int a = 4;
int b = 8;
boolean c = a > b; //c=false
```

Логические операции.

Используются для построения сложных условий

! – логическое НЕ (NOT)

&& - логическое И (AND)

|| - логическое ИЛИ (OR)

Операции указаны в порядке понижения приоритета

Как записать с помощью условий, что число x принадлежит отрезку [5;100]? Число x не принадлежит этому отрезку?

Логические операции.

Используются для построения сложных условий

! – логическое НЕ (NOT)

&& - логическое И (AND)

|| - логическое ИЛИ (OR)

Операции указаны в порядке понижения приоритета

Как записать с помощью условий, что число x принадлежит отрезку [5;100]? Число x не принадлежит этому отрезку?

a) x>=5 **&&** x<=100

b) x<5 **||** x>100

Тип boolean

Используется для хранения логических величин в Java. Значения типа **boolean** занимают 1 байт памяти

Переменные типа boolean могут принимать всего два специальных значения true и false (истина- ложь).

Значения этого типа можно сравнивать на равенство и производить с ними логические операции, но с ними нельзя производить вычисления.

Например:


```
int x = 5;
boolean b1 = x > 0;
boolean b2 = b1 && (x < 20);
```

Тернарная операция

Это операция с тремя операндами

?:

Синтаксис

<условие>?<значение, если условие true>:<значение, если условие false>

Например, поиск максимального из двух чисел:

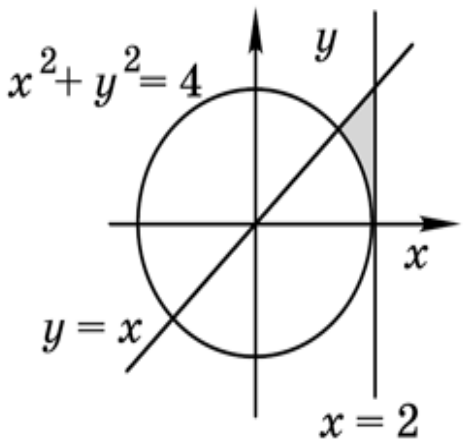
```
max = (a > b ? a : b);
```

Определить площадь квадрата по длине стороны или сообщить о некорректности данных

```
out.println (a > 0 ? a * a : "WRONG");
```

Задание

Решим задачу **112165.** Точка - 1.



Для вертикальной прямой условие очевидно: **x < 2**
Если дан график функции $y = f(x)$, то условие
«точка лежит **ниже** графика» записывается $y < f(x)$,
«точка лежит **выше** графика»**:** $y < f(x)$

Наконец, принадлежность точки кругу («**внутри окружности**») записывается $x^2 + y^2 < R^2$,
«**вне окружности**» - $x^2 + y^2 > R^2$,
в нашем случае **x * x + y * y > 4**.

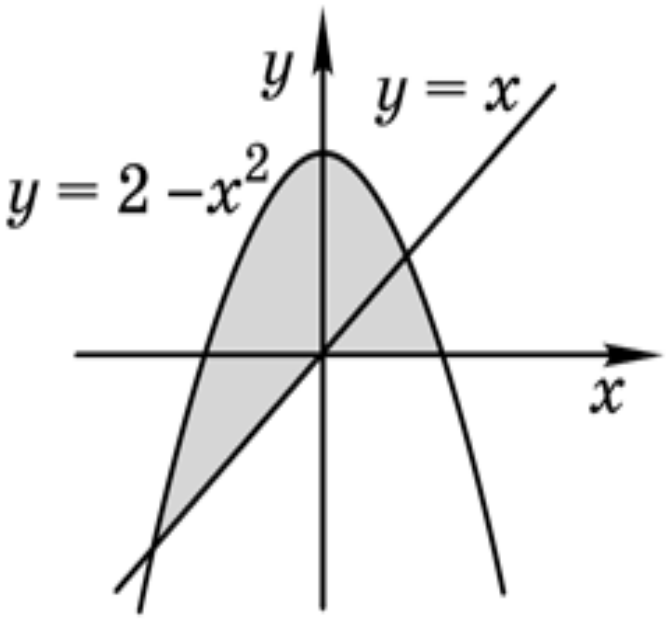
Задание

Точка принадлежит закрашенной области на рисунке, если она лежит левее вертикальной прямой **и** ниже графика функции y=x **и** вне окружности.

Запишите полное условие с использованием операторов сравнения и логических операций.

Для составления правильных условий, обычно надо разбивать сложную область на простые и связывать их логическими операциями.

Самостоятельно



1. Опишите условия попадания точки в выделенную область
2. Напишите программу, проверяющую, попадает ли точка с заданными координатами в выделенную область

Домашнее задание

1. Дорешайте задачи, предложенные на занятии.
2. Решить задачи informatics: № №112165, 112167, 112169.

3*. Решите задачу 3060 (определить, является ли число точной степенью двойки) **без использования циклов**.

Домашнее задание

1. acmp.ru -> Курсы -> Решение олимпиадных задач -> Целочисленная арифметика -> НОД и НОК -> C. Единичный НОД (№85)
2. acmp.ru -> Курсы -> Региональные олимпиады -> Полуфинал ВКОШП -> 2018-2019 -> A. Книги (№1616)