Модуль 1. Основы программирования

Тема 1.3. Представление целых чисел в памяти. Приведение типов

1 час

Оглавление

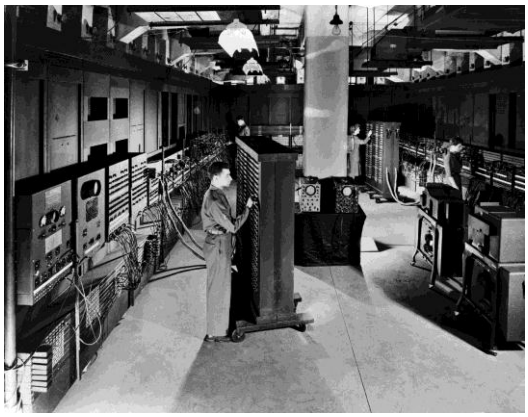
1.3. Представление данных в памяти

1.3.1. Р-ичные системы счисления

Не только обычному пользователю, но и современному программисту редко приходится обращаться к внутреннему представлению информации в компьютере. Конечно, для того и разработаны языки программирования высокого уровня, такие как Java, чтобы не задумываться о внутреннем устройстве компьютерной памяти и вычислений. Но для хорошего понимания того, как работает компьютер, иметь представление об этом необходимо, чтобы избежать некоторых ошибок, которые могут возникнуть в процессе вычислений.

Все знают, почему человек пользуется десятичной системой. Гипотеза о том, что десятичная система получила распространения из-за десяти пальцев на руках человека выглядит очень логичной. И всем, пожалуй, знакома двоичная система счисления. Достаточно открыть школьную тетрадь по информатике, чтобы увидеть ее описание на первой странице самых важных теоретических сведений. Действительно, вся информация в компьютере представлена в двоичной системе счисления – при помощи двух состояний – нуля и единицы.

Но не все знают, что это произошло не сразу. Блез Паскаль (не создатель языка Pascal, который был назван в честь него, а создатель первой вычислительной машины) считал десятичную систему счисления для автоматических вычислений волне удобной. Механическое счетное колесо он сделал десятичным: в нем было десять зубьев. Затем десятичная система перекочевала и в электромеханические счетные машины. В них был применен шаговый искатель с десятью позициями. И первые электронные вычислительные машины пользовались все теми же десятью «пальцами» - десятью триггерами.



На десятичной системе счисления работала, например, машина ЭНИАК. Но для нее требовалось столько дорогого оборудования, что конструкторы стали искать способы сокращения числа триггеров. Кроме того, десять состояний по тем временам было очень сложно разделить, часто возникали ошибки. В истории вычислительной техники были и троичные ЭВМ, в которых вычисления производились в так называемой симметрической троичной системе, кстати говоря, с определенной точки зрения она самая экономичная[1]. Система, имеющая всего два состояния (0 и 1, высокое и низкое напряжение) значительно более устойчива к помехам. И именно она получила распростренение.

Современные технические возможности волне позволяют сделать вычислительную машину, работающую в десятичной системе счисления, но это уже не целесообразно, уже очень много разработано именно под двоичную логику. Кроме, того, двоичная система счисления имеет перед десятичной определенные преимущества в использовании.

Другое распространенное заблуждение состоит в том, что двоичную систему считают современницей электронно-вычислительных машин. Она намного старше. Двоичным счислением люди интересуются давно. Особенно им увлекались с конца XVI до начала XIX века.

Лейбниц считал двоичную систему простой, удобной и красивой. Он говорил, что «вычисление с помощью двоек ... является для науки основным и порождает новые открытия ... При сведении чисел к простейшим началам, каковы 0 и 1, везде появляется чудесный порядок».

[1] Об этом можно подробно почитать, например, на странице http://5kr.mosuzedu.ru/darkblue04/troich.htm

По просьбе ученого в честь «диадической системы» – так тогда называли двоичную систему – была выбита медаль. На ней изображалась таблица с числами и простейшие действия с ними. По краю медали вилась лента с надписью: «Чтобы вывести из ничтожества все, достаточно единицы».

Основные преимущества двоичной системы перед десятичной в простоте операций, и возможность использовать булевы (логические) операции для вычислений. Об этом мы поговорим на следующем занятии.

Для того, чтобы понять что такое двоичная, троичная, шестнадцатиричная, стопятидесятиричная система счисления, нужно что, скажем, 3507 – это всего лишь *запись* числа, состоящего из трех тысяч, пяти сотен, нуля десятков и семи единиц. Мы фактически разложили число по степеням числа 10: $\mathbf{3}\cdot10^3 + \mathbf{5}\cdot10^2 + \mathbf{0}\cdot10^1 + \mathbf{7}\cdot10^0$ и записали и подряд коэффициенты. Но так же можно разложить число 3507 по степеням 8: $\mathbf{6}\cdot8^3 + \mathbf{6}\cdot8^2 + \mathbf{6}\cdot8^1 + \mathbf{3}\cdot8^0$ и записать сокращенно 6663_8. А можно взять в качестве **основания системы счисления** число 16: $\mathbf{13}\cdot16^2 + \mathbf{11}\cdot16^1 + \mathbf{3}\cdot16^0$. Тут нужно быть осторожным, нельзя записать просто 13113_{16}, потому что это будет означать уже $1\cdot16^4$ +… В этом случае используют дополнительные обозначения. В информатике принято использовать буквы латинского алфавита A – 10, B – 11, C – 12 и т.д. Таким образом можно записать наше число 3507 в 16-чной системе счисления будет выглядеть как $DB3_{16}$.

Переводить из системы с другим основанием в десятичную очень просто – по определению:

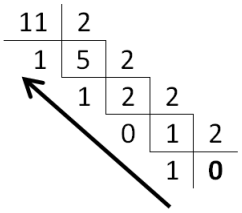
$10011_2 = \mathbf{1}\cdot2^4 + \mathbf{0}\cdot2^3 + \mathbf{0}\cdot2^2 + \mathbf{1}\cdot2^1 + \mathbf{1}\cdot2^0 = 16 + 2 + 1 = 19$

Обратно чуть сложнее. На предыдущем занятии мы решали задачу на нахождение последней цифры числа: для этого надо найти остаток от деления на 10. Соответственно, если мы найдем остаток от деления на два – мы получим последнюю цифру числа в двоичной системе. Дальше можно «скинуть» последнюю цифру - разделить число нацело на основание (например, 3507 / 10 получится 350) и опять найти последнюю цифру и так далее пока не получим ноль.

Пример 1

Обычно для расчетов на бумаге используют косой столбик:

11 / 2 = 5 (ост. 1)
5 / 2 = 2 (ост. 1)
2 / 2 = 1 (ост. 0)
1 /2 = 0 (ост. 1)
Ответ: $11 = 1011_2$



Действия в системах счисления с другим основанием выполняются так же, как и в десятичной.
Например:
$456_8 + 7467_8$:

```
  456
 7467
+
 10145
```

Обычным столбиком, только при этом нужно помнить, что 6 + 7 это 15 (в восьмеричной системе счисления один старший разряд - это 8) – «пять пишем, один переносим».

При этом таблица сложения в двоичной системе выглядит очень просто:

+	0	1
0	0	1
1	1	10

и 1 + 1 + 1 = 11

То есть фактически нужно помнить всего одну «странность» $1 + 1 = 10_2$

1.3.2. Представление целочисленных типов данных

Данные примитивных типов представляются фиксированным количеством двоичных разрядов.

Это очень важно, потому что задает ограничение на размер чисел. Например, если использовать всего три двоичных разряда, можно использовать только 8 различных значений: 000, 001, 010, 011, 100, 101, 110 и 111. То есть представлять 8 различных чисел. В реальных типах используется значительно больше разрядов.

Целочисленные типы в Java

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 1019)
char	2 (16)	'\u0000' .. '\uffff', или 0 .. 65.535

1.3.3. Приведение примитивных типов

Если в выражении участвуют операнды разных типов, происходит их приведение. Приведение в большую сторону происходит автоматически, в меньшую нужно делать вручную.

Можно написать

```
double z = 5;
```

Целое 5 при присваивании будет преобразовано в **`double`**, а вот

```
int x = 3 / 1.5; //НЕВЕРНО!
```

не скомпилируется, нужно приводить типы явно:

```
int x = (int)(3 / 1.5);
```

Правильно возвести 100 миллионов в квадрат можно так:

```
int x = 100 * 1000 * 1000;
out.println((long)x * x);
```

Достаточно при этом привести к типу ***long*** один из множителей.

1.3.4. Системы счисления на практике

Программисты редко используют двоичную систему для записи чисел, потому что она слишком объемная. Работая с двоичными числами, очень удобно использовать шестнадцатеричную систему счисления. 16 – четвертая степень двойки, поэтому перевести числа из двоичной системы в шестнадцатиричную можно очень легко.

Чтобы перевести число из двоичной системы в шестнадцатеричную, его нужно разбить на тетрады (четверки цифр), начиная с младшего разряда, в случае необходимости дополнив старшую тетраду нулями, и каждую тетраду заменить соответствующей шестнадцатеричной цифрой .

0_{16}	0000_2		8_{16}	1000_2
1_{16}	0001_2		9_{16}	1001_2
2_{16}	0010_2		A_{16} (10)	1010_2
3_{16}	0011_2		B_{16} (11)	1011_2
4_{16}	0100_2		C_{16} (12)	1100_2
5_{16}	0101_2		D_{16} (13)	1101_2
6_{16}	0110_2		E_{16} (14)	1110_2
7_{16}	0111_2		F_{16} (15)	1111_2

Например, число 1011100011_2 переведем в шестнадцатеричную систему счисления:

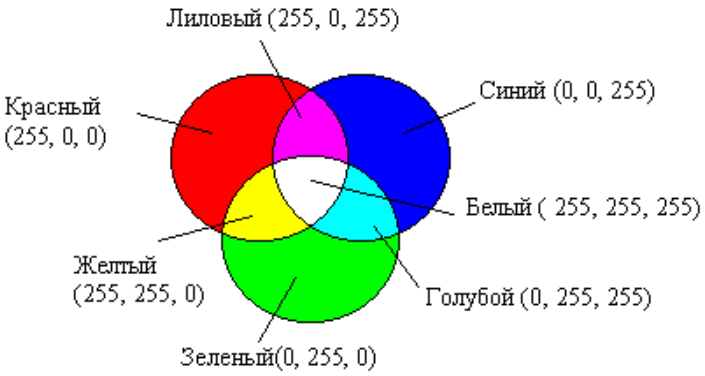
0010 1110 0011 = $2E3_{16}$

Для перевода шестнадцатеричного числа в двоичное необходимо каждую цифру заменить эквивалентной ей двоичной тетрадой:

$3E5_{16}$ = 0011 1110 0101_2

Программисты часто имеют дело с двоичными числами, но записывают их при помощи шестнадцатеричной системы счисления. Например, цвета принято кодировать тремя двухзначными 16-ричными числами (это составляет один байт).

Упражнение 1.3.1



Любой цвет можно представить в виде комбинации трех основных цветов: красного, зеленого и синего (цветовые составляющие).

В модели RGB берутся за основу красный, зеленый и синий цвета. Они удобны при воспроизведении цветов на мониторах компьютеров. Они устроены таким образом, что воспроизводят цвета путем «перемешивания» именно этих составляющих.

При кодировании цвета с помощью трех байтов первый байт является красной составляющей, второй байт – зеленой, а третий – синей составляющей. Чем больше значение байта соответствующей цветовой составляющей (в пределах от 0 до FF_{16}), тем больше ее насыщенность в итоговом цвете (см. рисунок).

Белый цвет имеет все цветовые составляющие с масимальной насыщенностью $FFFFFF_{16}$, R (red – красный) = $FF0000_{16}$, G (green зеленый) = $00FF00_{16}$, B (blue – синий) = $0000FF_{16}$.

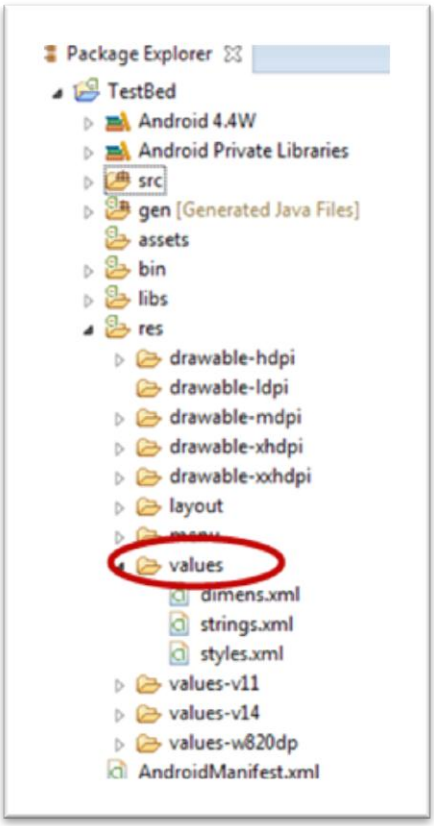
Примеры всех основных цветов ниже в таблице:

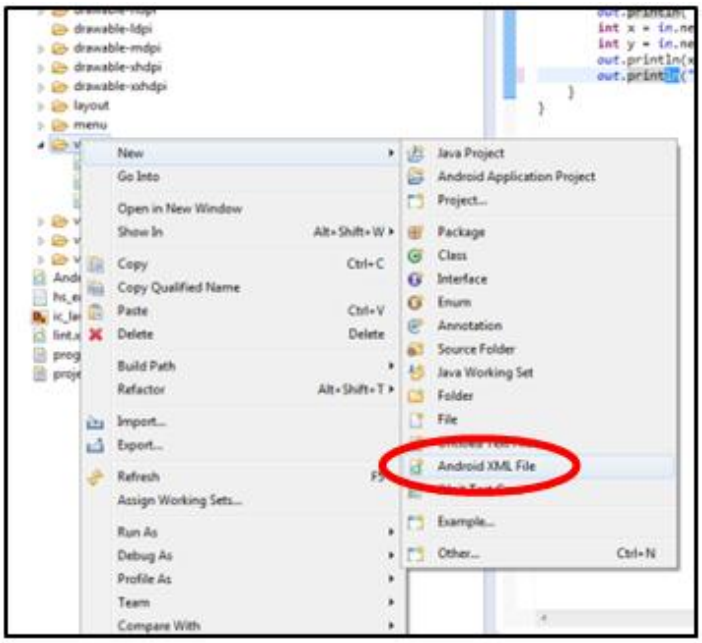
Цвет	Код HEX
	#000000
	#FF0000
	#00FF00
	#0000FF
	#FFFF00
	#00FFFF
	#FF00FF
	#C0C0C0
	#FFFFFF

При программировании хорошим тоном является отделение логики работы программы от оформления. В Android-программах, многое из того, что относится к оформлению помещается в раздел ресурсов и хранится в XML-файлах.

Идея проста: не указывать в программе конкретный цвет или размер, а описать его в файле ресурсов, присвоив ему идентификатор, а дальше использовать именно этот идентификатор (ID). Это дает возможность изменять внешний вид программы без изменения программного кода.

Обычно для цветовых ресурсов используют файл colors.xml в подкаталоге /res/values. Но можно использовать любое произвольное имя файла, или даже вставить их в файл вместе со строковыми ресурсами strings.xml. Android прочтет все файлы, а затем обработает их, присвоив им нужные ID.





Об устройстве XML детально мы поговорим на следующих занятиях, а сейчас, как обычно, будем действовать по аналогии.

Добавим цветности в TestBed.

Как видим в текущей версии цвета не используются.

Создайте новый файл XML: colors.xml (после выбора в меню задайте имя файла – colors.xml)

С созданным файлом можно работать в визуальном интерфейсе, но лучше сразу переключится в текстовый вид вкладка colors.xml

Цветовые ресурсы

Для работы с цветом используется тег **<color>**, а цвет указывается в специальных значениях.

#RGB;

#RRGGBB;

#ARGB;

#AARRGGBB;

A – это альфа-канал величина обратная прозрачности. То есть цвет #4000FF00 это почти прозрачный зеленый.

Например, в файле colors.xml можно написать

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="red">#f00</color>
    <color name="yellow">#FFFF00</color>
    <color name="transpgreen">#4000FF00</color>
</resources>
```

Мы описываем ресурсы – три цвета. После этого появится возможность использовать в других частях кода использовать красный, желтый и прозрачно-зеленый цвета.

Определенные таким образом цвета можно использовать в других xml-файлах.

Например, их можно в TestBed использовать в файле разметки основной активности (там описываются элементы главного окна приложения.

Откройте файл разметки **res ⇒layout ⇒activity_main.xml** и сразу переключитесь к текстовому виду (вкладка внизу activity_main.xml)

```
<FrameLayout xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context="com.example.testbed.MainActivity"
    tools:ignore="MergeRootFrame">

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    tools:context="com.example.testbed.MainActivity$PlaceholderFragment" >

    <EditText
        android:id="@+id/valuePrompt"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:ems="10"
        android:hint="@string/enter_value"
        android:visibility="invisible">

    </EditText>

    <TextView
        android:id="@+id/consoleWrite"
        android:layout_width="match_parent"
        android:gravity="top"
        android:layout_height="0dp"
        android:layout_weight="1"
        />

    <Button
```

```
        android:id="@+id/closeButton"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:text="@string/close"
        android:onClick="stop"
        android:visibility="gone"
        />

</LinearLayout>

</FrameLayout>
```

LinearLayout – все окно,

EditText – поле ввода,

TextView – основное поле вывода и

Button – кнопка закрытия, которая возникает после окончания работы запущенной консольной программы.

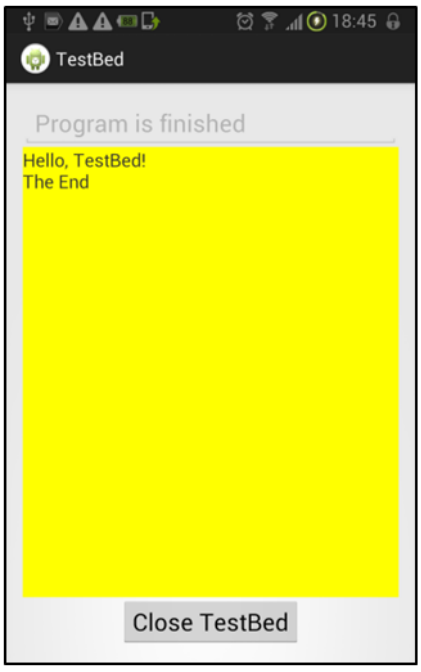
Например, зададим цвет основному полю ввода consoleWrite

Просто добавим в описание новый параметр:

```
<TextView
      android:id="@+id/consoleWrite"
      android:layout_width="match_parent"
      android:gravity="top"
      android:layout_height="0dp"
      android:layout_weight="1"
      android:background="@color/yellow" />
```

По смыслу все очень похоже на CSS. Мы задаем элементу свойство – цвет фона.

После запуска основное поле TestBed станет желтым.



Обратите внимание на использование префикса @ для того, чтобы ввести ссылку ресурса — текст после этого префикса — имя ресурса. В этом случае мы не должны были указывать пакет, потому что мы ссылаемся на ресурс в нашем собственном пакете.

Можете попробовать добавить различным элементам разметки кроме **android:background** свойства **android:textColor** и **android:TextColorHint** (это свойство применяется, когда в поле ввода еще нет текста пользователя и там отображается подсказка).

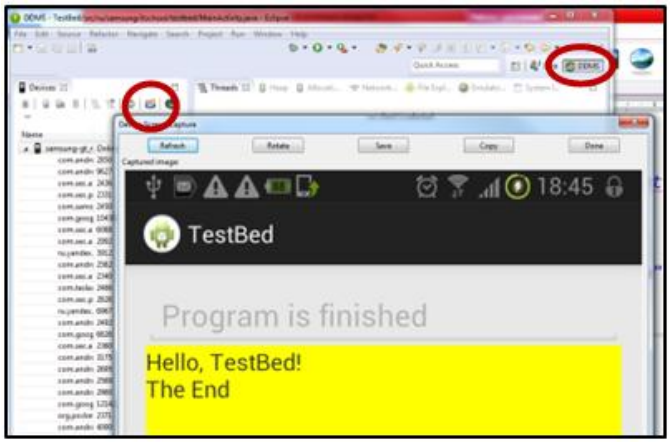
Вообще говоря, разные элементы поддерживают разные свойства, надо уточнять по документации.

Также существуют предопределенные названия цветов.

Такие ID доступны в пространстве имен **android.R.color**. Посмотреть цветовые значения цветов можно в документации http://developer.android.com/reference/android/R.color.html.

Для ссылки на системный ресурс мы должны, например, записать: **android:textColor="@android:color/black"**

Скриншот с устройства можно сделать прямо из Eclipse.



Для этого надо переключиться в перспективу DDMS (в правом верхнем углу) и нажать кнопку со значком фотоаппарата на тулбаре вкладки Devices (слева)

Задание 1.3.1

1. Сколько единиц содержится в двоичной записи года основания Samsung?

2. Попробуйте самостоятельно написать строку, задающую фиолетовый цвет. Проверьте результат на практике.

Благодарности

Компания Samsung Electronics выражает благодарность за участие в подготовке данного материала преподавателю IT ШКОЛЫ SAMSUNG Ильину Владимиру Владимировичу.