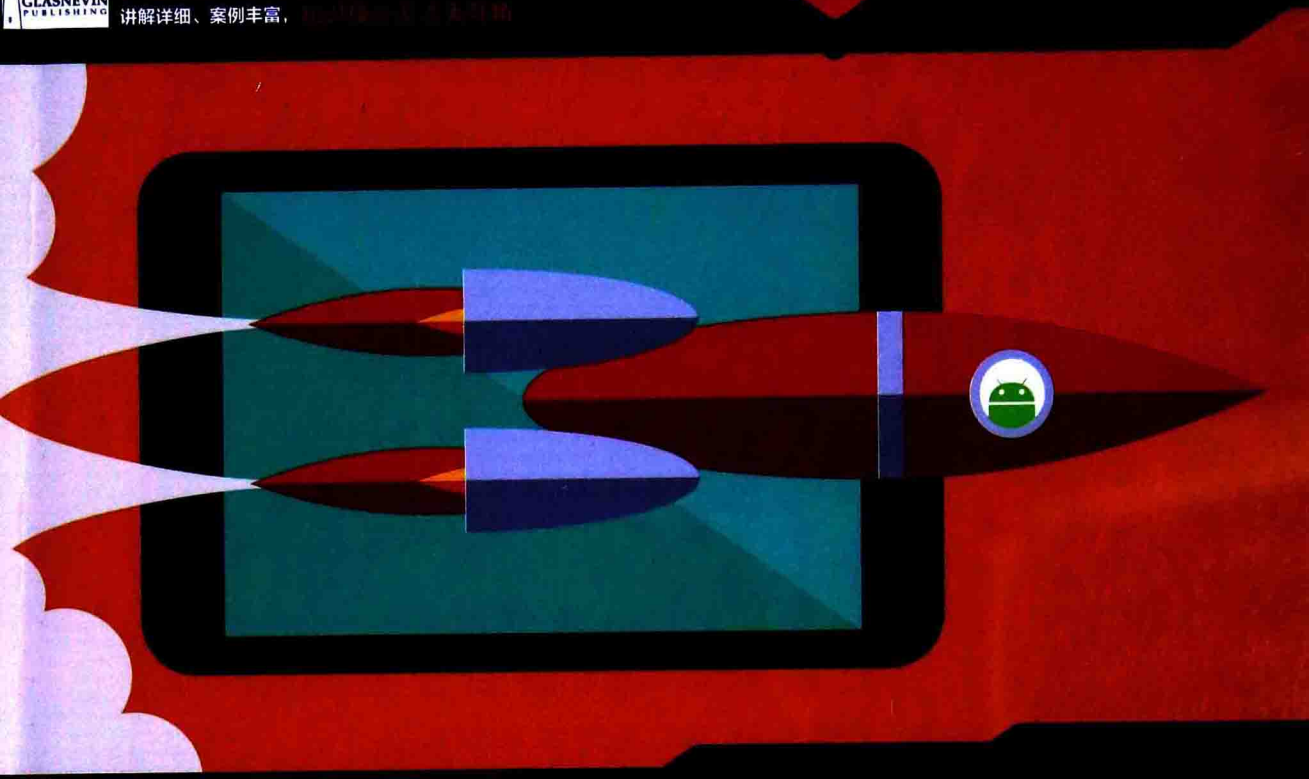
GLASNEVIN PUBLISHING

Android游戏开发教程

讲解详细、案例丰富，

游戏设计与开发



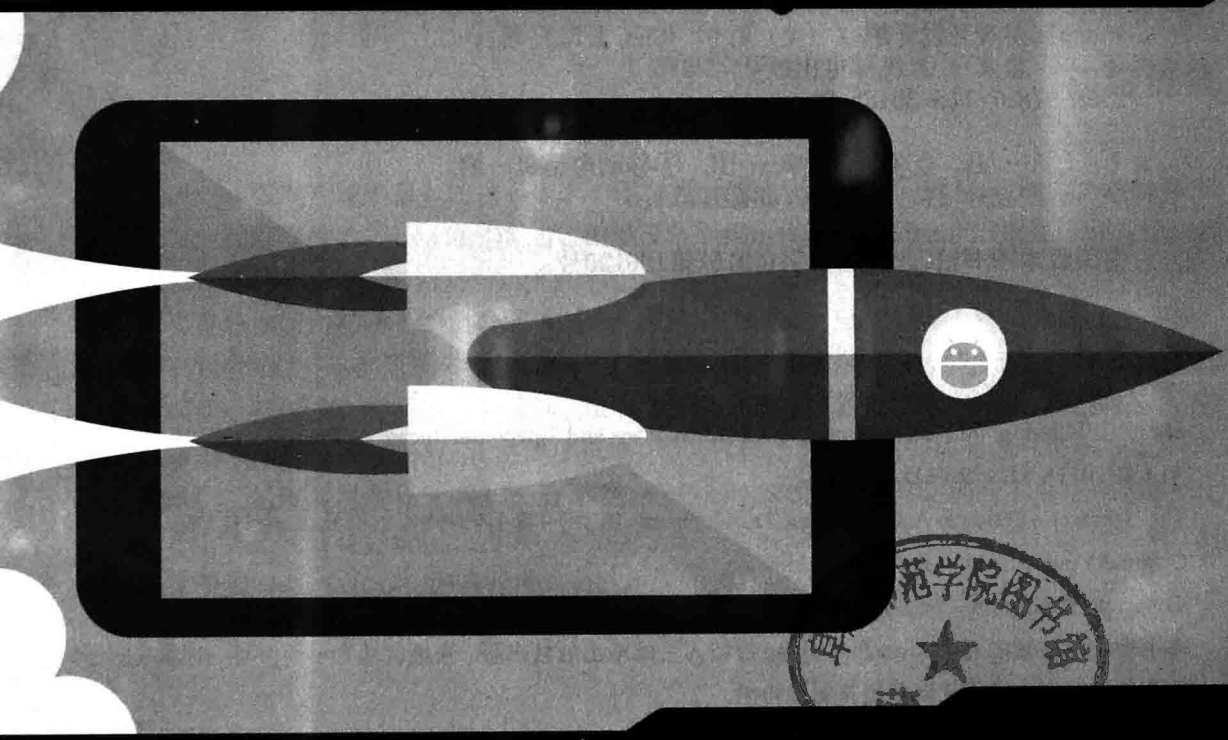
Android游戏开发详解

THE BEGINNER'S GUIDE TO ANDROID GAME DEVELOPMENT

[美] James S Cho 著　　李强 译

中国工信出版集团　人民邮电出版社 POSTS & TELECOM PRESS

游戏设计与开发



Android游戏开发详解

[美] James S Cho 著　　李强 译

人民邮电出版社
北京

图书在版编目（CIP）数据

Android游戏开发详解 / (美) 乔伊 (Cho, J.S.) 著 ; 李强译. -- 北京 : 人民邮电出版社, 2015.7
ISBN 978-7-115-39185-8

Ⅰ. ①A… Ⅱ. ①乔… ②李… Ⅲ. ①移动电话机－游戏程序－程序设计 Ⅳ. ①TN929.53②TP311.5

中国版本图书馆CIP数据核字(2015)第106130号

◆ 著　　[美] James S. Cho
译　　李　强
责任编辑　陈冀康
责任印制　张佳莹　焦志炜

◆ 人民邮电出版社出版发行　　北京市丰台区成寿寺路 11 号
邮编　100164'　　电子邮件　315@ptpress.com.cn
网址　http://www.ptpress.com.cn
北京鑫正大印刷有限公司印刷

◆ 开本：800×1000　1/16
印张：29
字数：638 千字　　2015 年 7 月第 1 版
印数：1－2 500 册　　2015 年 7 月北京第 1 次印刷
著作权合同登记号　图字：01-2014-7962 号

定价：59.00 元

读者服务热线：(010)81055410　印装质量热线：(010)81055316
反盗版热线：(010)81055315

内容提要

Android 游戏开发有很大的市场需求，但又容易给人以很简单的错觉。实际上，Android 游戏开发涉及编程基础、Java 编程语言、游戏开发、代码优化、Android 应用程序开发等众多的知识和技能。

本书是一本面向初学者的优秀的 Android 游戏开发指南。全书共 11 章，分为 4 个部分，按部就班地介绍了 Java 语言和编写面向对象的应用程序等基本知识，带领读者尝试 Android 的构建模块，并创建有趣的、交互性的、支持触摸控制的 2D 游戏。本书还通过配套站点，提供了众多的示例 Java 和 Android 游戏项目库，可供你自己继续学习并成长为一名游戏程序员。

如果你已经或者想要开发 Android 游戏，但是却不知道从何下手，那么本书是为你量身定做的。不管你是没有任何编程经验的初学者，还是一名有经验的 Java 开发者，都可以通过阅读本书成长为一名 Android 游戏开发人员。

前言

作为对编程知之甚少或者毫无所知的初学者，开始学习 Android 游戏开发，可能会觉得就像是穿越陌生的星际的旅程。有太多的事情要尝试，太多的知识要学习，令人遗憾的是，还有如此之多的方式令人陷入迷途。

究其原因之一，可能是 Android 游戏开发给人以很简单的错觉。这个术语给人的感觉是，只需要学习和掌握一个主题就够了，实际上，Android 游戏开发包括各种不同的主题，其中的一些如下所示。

- 编程基础；
- Java 编程语言；
- 面向对象设计原理；
- 游戏开发；
- 代码优化；
- Android 应用程序开发。

如果你不了解这些主题，也不必惊讶！这正是需要指南的地方。本书是为初学者而编写的，作者也曾经是初学者，不知道从何处开始学习。本书将引导你经历构建自己的 Android 游戏的每一个步骤。如果这正是你的学习目标，那么，这本书很适合你。

本书并不会对读者做太多假设。当然，我们假设你有基本的数学知识，并且知道如何在计算机上安装程序或应用，但是，并不会假设你之前编写程序，或者有物理学的学位。

如果你是第一次开始编写代码，肯定会遇到一些问题。这没事。实际上，当你遇到难处，请访问本书的配套网站并寻求帮助。无论是编辑、Kilobolt 的工作人员或者是陌生人，都会乐意帮助你解答问题或解决问题。

学习本书过程中，你将会阅读和编写很多代码。一些章节的整个篇幅都是学习如何编写代码，并且很少讨论游戏开发。其背后的思路是，如果你能够脱离游戏开发的环境去理解和编写代码，那么，在创建图形和游戏的时候，你可以很容易地应用这些知识。

通过从头到尾依次阅读，你将会从本书中获益良多。尽管如此，如果你记得对某个主题非常熟悉的话，跳过它也没问题。周期性的知识点检查，允许你下载工作项目的最新版本，并且从一个部分或一章的中间开始工作。

此外，要力图保持积极。你的学习旅程不会像穿越未知的星际那样紧张、刺激，但是，我期望它同样能够令人兴奋。有本书作为你的指导，你立刻就可以创建自己的游戏。

尽管本书的编写尽量全面，但是，一本书恐怕不足以涵盖 Android 游戏开发的主题。尽管如此，本书会随着配套网站一起完善。如果你觉得某个概念的介绍不够全面，请通过 jamescho7.com/book/feedback 反馈给我们。作者很高兴能够更详细地介绍一些重要的概念。

致谢

我想要感谢 Glasnevin 出版社的 Helen McGrath 博士，她给了我编写本书的机会。在整个过程中，她给予了巨大的帮助，没有她的话，不可能有这本书。

接下来，我要感谢 Dr. Bryan Mac Donald、Kyle Yu、Vignesh Sivashanmugam 以及所有其他不厌其烦地编辑我的书稿，使其尽可能减少错误的人们。得益于他们的努力，本书才能够成为那些想要学习 Android 游戏开发的人们的合适的指南。

感谢 Racheal Reeves 为本书封面做出的精彩设计。由于 Racheal 的努力工作，本书才如此完美。最后感谢 Ling Yang 无尽的耐心。Ling 的爱支持和激励我在很多不眠之夜努力工作以完成这本书，我希望 Ling 有一天能够读到它。

Web 资源

本书中给出的 Java 和 Android 代码，以及其他附加资源，都可以通过本书的配套站点获取：`http://www.jamescho7.com`。

对本书的任何评论、建议和勘误，欢迎通过以下电子邮件联系：

`info@glasnevinpublishing.com`。

作者简介

James 有多年的游戏开发经验。他最早在笔记本上开始了自己的游戏开发职业，最终创建了 Kilobolt，这是一家位于美国的独立游戏工作室。此外，他还教授一系列流行的编程课程，并且在杜克大学学习计算机科学的同时担任助教。

除了编写代码，阅读科学研究相关的文献，James 还是曼联球迷，并且不断探索新的美食。

目录

第 1 部分　Java 基础知识

第 2 部分　Java 游戏开发

第 1 章　程序设计基础

无论是何种情况，你离成为一名程序员都还相去甚远。本章将为你打下很好的基础，以便你能够成长为一名善于思考的、成功的 Java 程序员，从而能够编写高效的代码并构建优秀的游戏。我们从第 2 章开始，才会真正地编写程序，因此，现在你还不需要计算机。

1.1　什么是编程

从最基本的层面看，编程是让计算机执行以代码（code）的形式给出的一系列的任务。让我们来看一些示例代码，看看程序员能够提供什么样的指令。现在，还不要关心每个符号和每行代码背后的含义。我们将在本书中详细介绍这些。现在，先尝试理解其逻辑。阅读每行代码前面的注释，尝试搞清楚后面的代码的意图。

程序清单 1.1　程序员的指令

```
01 // Instruct the computer to create two integer variables called a and
02 // b, and assign values 5 and 6, respectively.
03 int a = 5;
04 int b = 6;
05 // Create another integer variable called result using a + b.
06 int result = a + b;
07 // Print the result (Outputs the value of result to the Console).
08 print("The value of a + b is " + result);
```

程序清单 1.1 展示了程序员输入到像 Notepad（Windows）或 TextEdit （Mac）这样的一个文本编辑器中的内容。计算机在控制台所产生的输出如下所示。

```
The value of a + b is 11
```

好了，我们看完了 Java 代码的一个小示例。在继续学习之前，这里有一些需要记住的关键知识点。

关键知识点

代码执行的基本规则

代码是从上到下一行接着一行地执行的。这是一个简化的说明，但是，现在很适合我们。

稍后，我们会给这条规则添加内容。

注释(//)

在 Java 中，两条斜杠后面的内容是注释。注释是为人类而编写的（在这里是我向你描述代码的方式），因此，Java 虚拟机（Java Virtual Machine，稍后详细介绍 Java 虚拟机）不会执行注释。

行号

我们可以通过行号来引用代码。在确定行号的时候，必须把注释和空行都算在内。例如，在程序清单 1.1 中，如下的代码出现在第 3 行。

```
int a = 5;
```

正如程序清单 1.1 所示，我们可以让计算机把值存储为变量，并且我们可以对这些值执行数学计算和连接（连接是将文本和整数组合起来，参见程序清单 1.1 第 8 行）。我们甚至可以在控制台显示这些运算的结果。这只是冰山一角。稍后，我们可以绘制一个视频游戏角色，并且实现它在屏幕上移动的动画，它每走一步还会发出脚步声。看上去如下所示（注意，下面只是一个示例。在学习完本书的几章之后，你将能够编写自己的代码）。

程序清单 1.2　更复杂的指令的示例

```
while (mainCharacter.isAlive()) {
   mainCharacter.updatePosition();
   mainCharacter.animate(time);
   if (mainCharacter.getFoot().collidesWith(ground)) {
     footstepSound.play(volume);
   }
   screen.render(mainCharacter);
}
```

1.2　数据类型

1.2.1　基本类型

在前面的示例中，我们看到了数据类型（*data type*）的例子。例如，在程序清单 1.1 中，我们使用了整数值（*integer value*）5 和 6，这两个都是数值数据的例子。我们来看看其他的数据类型，先介绍其他的数值类型。

- 可以使用 4 种类型来表示整数（*Integer*），每种类型都用不同的大小。在 Java 中，我们有 8 位的 byte、16 位的 short、32 位的 int 和 64 位的 long。4 种类型中的每一种，都可以保存正的和负的整数值。
- 有两种类型可以表示小数值（*Decimal*，如 3.14159）：32 位的 float 和 64 位的 double。
- 可以使用 char 表示一个单个的字符或符号。

这些是 Java 中主要的基础数据类型，我们称之为基本数据类型（*primitive type*）。在后面各章中，我们将会看到很多使用基本数据类型的例子。

1.2.2 字符串

字符串（*String*）指的是一系列的字符。正如其名称所示，我们可以使用字符串将多个字符保存到一起（而基本类型 char 只能够保存一个字符）。

```
char firstInitial= 'J';
char lastInitial = 'C';
String name = "James";
```

注意，这里关键字 String 的首字母大写了，而基本数据类型 char 的首字母没有大写。这是因为，字符串属于对象（*object*）一类，而不属于基本数据类型。我们稍后要花很多时间讨论这些对象，它们在 Java 编程中扮演重要的角色。现在，我们只需要将字符串当作基本数据类型一样对待就行了。

1.3 声明和初始化变量

所有的基本数据类型（和字符串）都可以表示为变量。它们都是使用相同的基本语法来声明（创建）的。

创建一个新的变量的时候，我们总是要声明两件事情：变量的数据类型（*data type*）和变量的名称（*variable name*）。在大多数情况下，我们还使用赋值运算符（*assignment operator*，即=）给变量指定一个初始值。有两种方法做到这点。第一种方法是指定一个字面值（*literal value*），例如，图 1-1 所示的'J'。第二种方法是，指定一个计算值的表达式（*expression*），例如，图 1-1 所示的 35 + 52（这个表达式在赋值之前计算）。

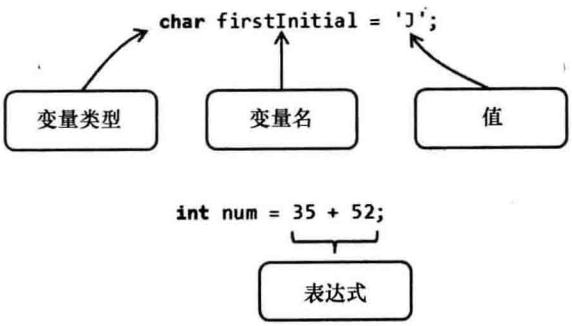


图 1-1 变量声明的示例

赋值运算符（=）不是在声明相等性。这一点很重要。正如其名称所示，我们使用赋值运算符把一个值（在等号的右边）赋给（*assign*）一个变量（在等号的左边）。例如，考虑如下两行代码。

```
int a = 5;
a = a + 10;
```

在这个例子中，我们不是要表示 a 和 a + 10 的对等关系。我们直接将 a + 10 的值赋值给一个已有的变量 a。进行区分的一种较好的方法是，将等号读作“获得”。因此，图 1-1 应该读作，“int 类型的 num 获得了表达式 35 + 52 的结果”。

作为一个练习，浏览一下程序清单 1.3 的 6 行代码中的每一行，并且尝试将其读出声，说出每行的含义。记住，要区分字面值（*literal value*）和表达式（*expression*）（如果忘记了，再回顾一下图 1-1）。第 1 行应该读作“short 类型的 num 获得了 15”。记住，这意味着，“声明了一个类型为 short、名为 num 的变量，并且将字面值 15 赋值给它”。

程序清单 1.3　声明各种变量

```
1   short numberOfLives = 15;
2   long highScore = 21135315431 - 21542156; // uses an expression;
3   float pi = 3.14159f;
4   char letter = 'J';
5   String J = "James";
6   boolean characterIsAlive = true;
```

1.3.1　变量名和字面值

注意，当我们讨论字符和字符串的时候，使用了‘’和“　”将字面值和具有相同名称的变量（*variable*）区分开来。例如，在程序清单 1.3 中，变量名 J 引用“James”，而字面值‘J’指的是它自己。

1.3.2　初始化或不初始化

在以上的每个示例中，我们在声明过程中都使用了一个初始值来初始化（*initialized*）变量。然而，正如我在本节开始所介绍的，声明一个变量的时候初始化它（分配一个初始值），这本身并不是必须要做的。例如，我们可以这么做。

```
int a, b, c;
a = 5;
```

```
b = 6;
c = 7;
```

上面的第 1 行代码声明了 3 个整数类型，分别名为 a、b 和 c。没有明确地赋给其初始值。接下来的代码行分别用值 5、6 和 7 初始化了这 3 个整数。

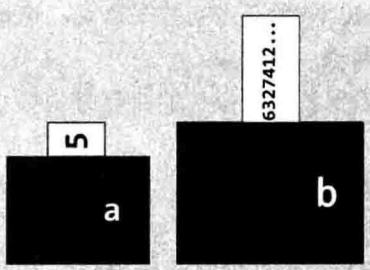
尽管这么做是允许的，我们通常也将声明变量并初始化它们，就像在前面程序清单 1.3 中所做的那样。

关键知识点

声明变量

当创建一个新的变量的时候，我们把一个值存储到计算机的内存中以便随后使用。我们可以使用该变量的名称来引用它。

打个比方，可以把变量看作一个盒子。当我们输入 int a = 5 的时候，是告诉 Java 虚拟机创建一个相应大小的盒子，并且把我们的值放进去。



引用变量

一旦创建了变量，在引用它的时候，我们不应该声明其类型。提供变量的名称就足够了。

复制值

考虑如下所示的代码。

```
int x = 5; // declare a new integer called x
int z = x;  // assign the value of x to a new integer z
z = z + 5;  // increment z by 5
[End of Program]
```

你能告诉我，在程序结束的时候，x 和 z 分别是什么值吗？如果你的回答是 5 和 10，那么你答对了！

如果不是，也不必担心。很多初学者都不能正确地理解第 2 行代码。在第 2 行代码中，我们不是说 int x 和 int z 引用相同的盒子（变量）。相反，我们创建了一个名为 int z 的新盒

子，并且将 int x 的内容复制后赋给它。

这对我们来说意味着什么呢？这意味着，当我们在第 3 行将 z 和 5 相加的时候，z 变成了 10，而 x 仍然是 5。

1.4　关于位的一切（位和字节的简单介绍）

在我们继续深入之前，值得先细致地介绍如何具体把值存储到变量中。我前面提到，不同的基本数据类型具有不同的位大小。例如，一个 int 有 32 位而一个 long 有 64 位。你可能会问，那么，到底什么是位？

位（*bit*）是一个二进制位的简称。换句话说，如果你有一个只有 0 和 1 的二进制数，每个数字就是 1 位。达到 8 位的时候，例如，（10101001），你就有了 1 字节。

对于位，你需要记住的一点是：拥有的位越多，所能表示的数值也越多。为了说明这一点，让我们问一个问题。十进制的 1 位能够表示多少个数字？当然是 10 个（0，1，2，3，4，5，6，7，8 和 9）。两位呢？100 个（00，01……99）。我们看到，每增加一个位数，都会使得我们所能表示的数值增多到原来的 10 倍。对于二进制数字来说，也是如此，只不过每次增加一位，所能表示的数值的数量是原来的两倍。

在计算中，位是很重要的，因为我们所操作的机器是由细小的电路组成，而这些电路要么是开，要么是关。数据表示的挑战，完全由此而引发。我们不能使用这些电路来直接表示“hello”这样的单词。我们必须使用任意某种系统将单词“hello”和某些电路的开关组合联系起来。

这就是我们应该了解的和变量相关的知识。通过声明一个新的变量，我们在内存中分配了特定数目的位（根据声明的类型），并且将某些数据的一个二进制表示存储起来，以便随后使用。

在数据类型之间转换

在 Java 中，可以将一种数据类型转换为另一种类型。例如，我们可以接受一个 int 值并且将其存储到一个 long 变量中。之所以能这样，是因为 long 变量保存了 64 位，可以很容易地容纳来自较小的类型 int（32 位）中的数据而不会遇到麻烦。但是，如果我们接受一个 64 位的 long 数字，并且试图将其放入到一个 32 位的 int 的“容器”中，将会发生什么呢？会有丧失精度的风险。这 64 位中的 32 位，必须删除掉，然后我们才能将数字放置到 int 变量中。

规则是这样的：如果从一个较小的类型转换为一个较大的类型，这是安全的。如果要从一个较大的类型转换为一个较小的类型，应该小心以避免丢失重要的数据。稍后，我们将详细介绍如何从一种类型转换为另一种类型。

1.5 运算

我们前面看到了，变量可以用来存储值，并且变量可以在运算中用作运算数，如图 1-2 所示。

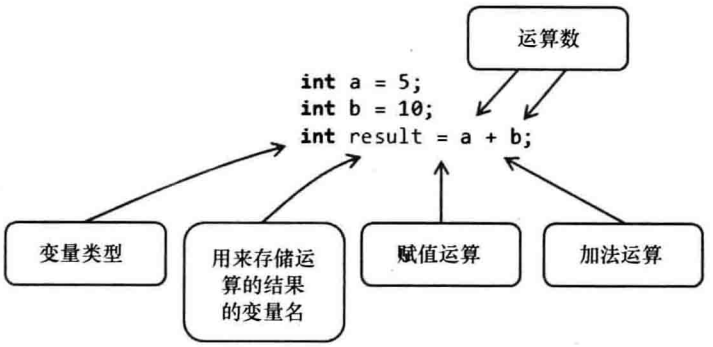


图 1-2 变量可以用来存储值，也可以用作运算数

1.5.1 算术运算

表 1-1 所列内容是你必须知道的 5 种算术运算。在了解示例的过程中，请记住如下两条规则。

规则#1 涉及两个整数的一个运算，总是会得到一个整数的结果（整型变量中不允许有小数值）。

规则#2 至少涉及一个浮点数（小数值）的运算，其结果总是浮点数。

表 1-1 必须知道的 5 种运算

运算符	说明	示例/值
+（加法）	将加号两边的运算数相加	3 + 10 = 13 4.0f + 10 = 14.0f
−（减法）	从第一个运算数中减去第二个运算数	16 − 256 = -240
*（乘法）	将乘号两边的运算数相乘	4.0f * 3 = 12.0f
/（除法）	用第一个运算数除以第二个运算数 前面提到的两条规则在这里特别重要	6/4 = 1 记住规则#1，我们将其向下舍入到最近的整数 6.0f/4 = 1.5f 记住规则#2
%（求余数、模除）	计算除法运算的余数	10 % 2 = 0 （执行 10/2，然后计算余数，也就是 0） 7 % 4 = 3

1.5.2　运算顺序

在执行运算的时候，使用标准的运算顺序。计算机将会按照如下的顺序执行运算。

1．圆括号（或方括号）。

2．指数。

3．乘法/除法/余数。

4．加法/减法。

如下的示例说明了运算顺序的重要性。

```
print(2 + 5 % 3 * 4);——输出“10”。
print((2 + 5) % 3 * 4);——输出“4”。
```

1.5.3　关系/布尔运算

现在来看看在两个值之间进行比较的关系运算符，如表 1-2 所示。注意，在下面的示例中，算术运算在关系运算之前执行。如下所有的计算，都得到一个 true 或 false 值（布尔）。

表 1-2　　关系运算符用来确定一个值与另一个值进行比较的结果

运算符	说明	示例/值
==(等于)	检查运算符两边的两个值是否相等	(3 + 10 == 13) = true (true == false) = false
!=（不等于）	检查运算符两边的两个值是否不相等	(6/4 != 6.0f/4) = true
>（大于）	判断第一个运算数是否比第二个运算数大	6 > 5 = true
<（小于）	判断第二个运算数是否比第一个运算数大	6 < 5 = false
>=（大于或等于）	含义明显	6 >= 6 = true
<=（小于或等于）	含义明显	10 <= 9 + 1 = true
!（取反）	将一个布尔值取反。这是一个一元运算符（只需要一个运算数）	!true = false !false = true