面向对象程序设计Java

江春华

电子科技大学信息与软件工程学院

内　容

第2章　Java语言基础

1. Java程序基本结构
2. Java字符集
3. Java数据类型
4. 常量与变量
5. 运算符与表达式

Java程序基本结构

UESTC

- Java源程序中可包含三个基本部分：
  - 一个包声明package语句(可选)；

    ```
    package database;
    ```
  - 任意数量的引入import语句(可选)；

    ```
    import java.applet.Applet;
    ```
  - 类和接口声明。

    ```
    class Hello{ ... }
    interface DataCollect{ ...}
    ```

Java程序基本结构

❖包声明：package语句

- 包是类和接口的集合，即为类库；
- 用类库管理类，方便对类和接口管理，减少类名、接口名之间的重名问题；
- Java的类都包含在类库中，package语句为类、接口(或者说是字节码文件)来指定所属的类库(包)。

在一个源程序中，只能有一个包声明语句，且是程序的第一条语句。

Java程序基本结构

❖引入语句：**import**语句

- 源程序中可以有任意条import引入语句；
- 当源程序在编译时，会将需要的在引入语句中的类引入到程序中。

import语句在包语句后，所有类或接口之前。

Java程序基本结构

❖类和接口声明

- 类和接口是程序的基本组成单元；
- 类是由成员变量和成员方法等组成，表示了对象的基本属性和行为；
- 接口表现了对象所具有的行为规范。

💣源程序中至少有一个类或接口创建。

Java字符集

❖符号集

- 符号是构成程序的基本单位。Java采用的是Unicode码，又称统一码字符集，使用16位存储空间，支持多种语言，更具有国际化特性；
- 当Unicode中的高8位为0时，则低8位的编码与ASCII码相同。

ASCII码是用8位存储空间。

Java字符集

- Java的符号也分为五种类型
  - 关键字(Keywords);
  - 标识符(Identifiers);
  - 常量(Literals);
  - 运算符(Operands);
  - 分隔符(Separator)。

Java字符集

❖关键字(Keywords)

- 关键字是构成编程语言本身的符号，是一种特殊的标识符，又称保留字。Java语言中关键字有40多个。

boolean	break	byte	case	catch	char	class
continue	default	double	else	extends	false	final
finally	float	for	if	implements	import	instanceof
int	interface	long	native	new	null	package
private	protected	public	return	short	static	super
switch	synchronized	this	throw	throws	true	transient
try	void	volatile	while	abstract		

Java字符集

❖关键字(Keywords)

- 对Java编译器有特殊的含义，标识数据类型或程序构造名。编译器通过对关键字的检查程序合法性；
- 注意以下有关关键字的重要事项：

➢ true、false和null为小写，不能大写。严格地讲，它们不是关键字，而是一种值。但是仍然把它们作为关键字使用。

➢ 所有类型的长度和表示是固定的，不能在程序的运行中改变它。

➢ 不能作为一般的标识符使用，即一般的标识符 (变量名、类名、方法名等)不能与其同名。

Java字符集

❖标识符(Identifiers)

- 在Java语言中，标识符取名的规则：
  - Ø 必须由字母、下划线或美元符开头的；
  - Ø 并由字母、数字、下划线和美元符组成的；
  - Ø 不能与关键字同名；
- 例如：
  - ➢ 合法标识符：

    Identifier、userName、User_Name
  - ➢ 不合法标识符：

    2mail、room#、class

Java字符集

- 标识符(Identifiers)
  - 标识符名有：
    - 类名、接口名；例：Hello、DataCollect
    - 对象名、数组名、变量名、方法名、语句标号。
      tom、font、stuName、setData()、...
  - 好的取名习惯：
    - 类名、接口名的第一个字母大写，其余小写；
    - 其它标识符第一个字母小写。
    - 在标识符中的单词第一个字母大写，其它小写。

Java字符集

❖程序中的注释

- 适当注释会大大增强程序的可读性，注释内容本身不对程序执行产生任何影响，只会使程序易读。
- 三种注解：
  - //　　--由//开始到行末为注释内容。

    例：　int stuName; //学生名
  - /* */ --在/*到*/之间为注释。

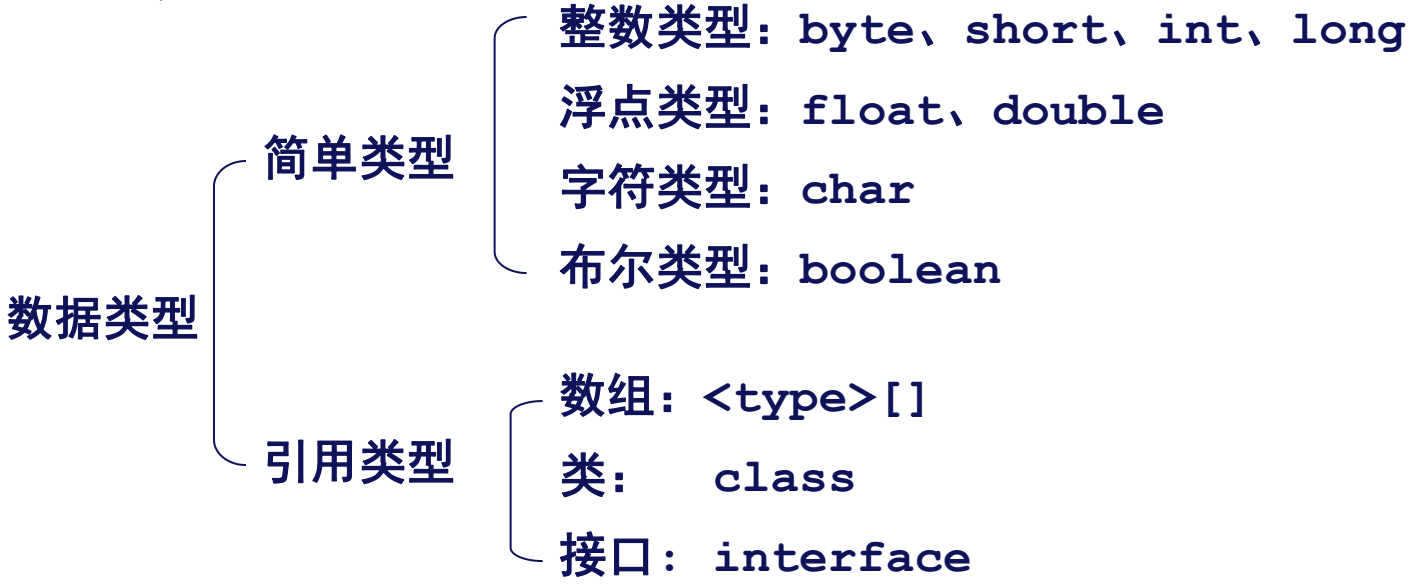
    例：　/* 源程序：Hello.java */
  - /** */--在/**到*/之间为注释,javadoc专用。

    例：　/** 初始化成员变量的值 */

Java数据类型

❖数据类型

- Java语言的数据类型有两大类：简单类型和引用类型。



Java数据类型

❖简单数据类型

- 分为整数型、浮点型、字符型和布尔型。其开销为：

类型	存储(bit)	最小值/值	最大值/值	说 明
boolean	1	false	true	布尔型
char	16	Unicode: 0	Unicode: $2^{16}-1$	字符型
byte	8	-128	+127	字节整型
short	16	-2^{15}	$+2^{15}-1$	短整型
int	32	-2^{31}	$+2^{31}-1$	整型
long	64	-2^{63}	$+2^{63}-1$	长整型
float	32	IEEE754	IEEE754	单精度浮点型
double	64	IEEE754	IEEE754	双精度浮点型
void	-	-	-	无类型

Java数据类型

❖简单数据类型

- Java语言数据中的数值类型都是有符号(正负号)的，在贮存数值类型的数据时，其最高位用来表示数据的正负号。

- 简单类型的变量被声明时，存储空间也同时被分配。该贮存空间只占用一个单一贮存单元。对简单类型变量访问则直接可以得到它的数据。

Java数据类型

❖引用数据类型

- 引用类型(数组、class或interface)声明变量时，是不会为变量(即对象)分配存储空间。它们声明的变量不是数据本身，而是数据的引用(reference)，需用new运算符来为引用类型的变量分配贮存空间；
- 引用:类似C/C++中的指针，但又不同于C/C++中的指针，它的引用必须由Java的虚拟机创建和管理。Java语言本身不支持指针；
- 引用类型变量的值是一个数据的引用(即地址)。它是对占有由多个贮存单元构成的贮存空间的引用。引用类型的变量通过点“.”运算符访问它的成员。

Java常量

- 常量是指直接用于放入程序中的固定不变的值。
- 它的表现形式有两种：数值和字符。
- 每一种简单数据类型常量都有固定的表现形式。

Java常量

❖整数型常量

- Java整数类型常量有三种形式：十进制、八进制、十六进制。
  - Ø 十进制整数是由不以0开头，0～9数字组成数据：**12**；
  - Ø 八进制整数是由以0开头，0～7数字组成的数据：**012**；
  - Ø 十六进制整数是由以0x或0X开头，0～9数字及A～F的字母组成的数据：**0x12AB**。
- 整型数常量均为int类型，除非在其后有字母“L”来表示是长整型long的值。

Java常量

❖浮点型常量

- 浮点数类型有float单精度浮点数，double双精度浮点数。在数字后面带有字母F或f（float）、D或d（double）分别表示单/双精度的浮点数值。
- 在数值后面不带有任何大小写字母f或d时，表示为**double**数值。
- 例如：如下形式表示的单/双精度型数值

3.12E20	**一个带指数的大浮点数值**
1.567F	**一个单精度浮点数值**
42.314E+307D	**一个带指数的双精度浮点数值。**

Java常量

❖字符型常量

- 常量是由单引号''包括的单个Unicode字符。
  例：'A'、'9'、'@'
- 是一个16位无符号的Unicode字符。
- 在字符型常量中，也有用带"\"来表示的特殊字符，是其中的一些不可显示或有特殊意义的字符。
  例：'\n'、'\t'

Java常量

- 由"\"表示的转义字符。

字符序列	表示方法	功能
\n	Linefeed	换行符
\t	HT	水平制表符
\b	Backspace	退格符
\r	Carriage return	回车符
\f	Form feed	进纸
\\	\	反斜线
\'	'	单引号
\"	"	双引号
\ddd	0ddd	八进制位模式
\xdd	0xdd	十六进制位模式
\udddd	0xdddd	Unicode字符

Java变量

❖变量

- 变量是语言编程中用来标识存储地址的名称。
- 程序通过变量名访问所标识贮存空间的数据。
- 变量必须显式地声明变量的类型。遵循“先声明、后使用”原则。

Java变量

❖变量声明

- 变量声明包括两个部分：变量的数据类型和变量的名称。声明形式：

```
type varName1[=初值] [,varName2[=初值]];
```

- 例：

```
int score;
float x = 19.9F;
double pi = 3.14;
char alph = 'A';
boolean flag = true;
```

Java变量

- 变量的分类及作用域
  - 依变量创建所在处可分为：
    - Ø成员变量；
    - Ø方法的变量(包含参数)；
    - Ø语句块的变量；
    - Ø异常处理的变量。
  - 依变量作用域可分为：
    - Ø全局变量：成员变量；
    - Ø局部变量：方法的变量(包含参数)；
    - Ø局部变量：语句块的变量；
    - Ø局部变量：异常处理的变量。

Java变量

❖变量的初始化

- 变量作为成员变量，在声明时会有一个初始化的值。
- 变量作为局部变量，在声明时不会有初始化的值。
- 成员变量初始化的值如左表所示。

类型	值
byte	0
short	0
int	0
long	0L
float	0.0f
double	0.0d
char	'\u0000'
boolean	false
引用类型	null

运算符与表达式

- 运算符按数目可分为：
  - 单目(一元)运算符：
    有一个操作数；例：`i++`
  - 双目(二元)运算符：
    有两个操作数；例：`a+b`
  - 三目(三元)运算符：
    有三个操作数。
    例：`x>y?a:b`

- 运算符功能分类如左表：

类别		运算符
算术运算符		+ - * / % ++ --
关系运算符		> >= < <= = !=
布尔运算符		! && ||
位运算符		>> << >>> & | ^ ~
赋值运算符		= Op=
条件运算符		?:
其它运算符	下标	[ ]
	实例	instanceof
	内存分配	new
	强制类型转换	(数据类型)
	方法调用	()

运算符与表达式

❖表达式

- 表达式是变量、常量、运算符、方法等按照一定的运算规则组成的序列，并返回一个值。
  例：`( x + 12.3/y ) >= 10`
- 表达式是运算符运算的表述，它返回值不仅与表达式中的操作数有关，而且还是运算符操作顺序有关。
- 表达式有时也称为运算式。

运算符与表达式

❖算术运算符

运算符	使用方式	说明
+	opD1+opD2	opD1加上opD2
-	opD1-opD2	opD1减去opD2
*	opD1*opD2	opD1乘以opD2
/	opD1 / opD2	opD1除以opD2
%	opD1%opD2	opD1除以opD2的余数

运算符与表达式

❖关系运算符

运算符	使用方法	功能	说明
>	opD1>opD2	大于	若opD1>opD2成立，为true，否则为false
>=	opD1>=opD2	大于等于	若opD1>=opD2成立，为true，否则为false
<	opD1<opD2	小于	若opD1<opD2成立，为true，否则为false
<=	opD1<=opD2	小于等于	若opD1<=opD2成立，为true，否则为false
==	opD1==opD2	相等	若opD1==opD2成立，为true，否则为false
!=	opD1!=opD2	不相等	若opD1!=opD2成立，为true，否则为false

运算符与表达式

❖布尔运算符

运算符	使用方法	功能	说　　明
&&	opB1&&opB2	与	若opB1，opB2全true则为ture，有false则false
||	opB1||opB2	或	若opB1，opB2全false则为false，有true则ture
!	!opB1	非	若opB1是true则为false，是false则为true;

- Java的布尔运算符是一种优化的运算符。
- 运算符&&和II的第一操作数在某种值的情况下，就可以确定结果，就不用再去访问第二个操作数。

运算符与表达式

❖布尔运算符优化

- &&运算符：**opB1&&opB2**
  - 如果opB1值为false，则运算式的值就是false，无论opB2的值是什么。程序不会访问opB2；
  - 如果opB1值为true，则需要opB2的值才能确定运算式的值，程序需要访问opB2。
- ||运算符：**opB1||opB2**
  - 如果opB1值为true，则运算式的值就是true，无论opB2的值是什么。程序不会访问opB2；
  - 如果opB1值为false，则需要opB2的值才能确定运算式的值，程序需要访问opB2。

运算符与表达式

❖位运算符

- 位运算符是对数据的二进制位操作，位运算符的操作数只能是整型的数据。可分为移位操作和逻辑运算。

运算符	使用方法	说明
>>	opBt1 >> opBt2	opBt1右移opBt2位
>>>	opBt1 >>> opBt2	opBt1无符号右移opBt2位
<<	opBt1 << opBt2	opBt1左移opBt2位
&	opBt1 & opBt2	opBt1和opBt2按位与
|	opBt1 | opBt2	opBt1和opBt2按位或
^	opBt1 ^ opBt2	opBt1和opBt2按位异或
~	~ opBt1	opBt1按位取反

运算符与表达式

- 右移位运算符>>、>>>

运算式	结果
256	00000000000000000000000100000000
256>>3	00000000000000000000000000100000
256>>>3	00000000000000000000000000100000
-256	11111111111111111111111100000000
-256>>3	11111111111111111111111111100000
-256>>>3	00011111111111111111111111100000

运算符与表达式

❖左移位运算符<<

- 执行一个左移位。移位的结果是第一个操作数乘以2的幂，而这个幂的指数就是第二个操作数。
- 左移位时，高位被截去，低位填充0。

例：256 << 3 结果是 $256 \times 2^3 = 2048$。

运算式	结果
256	00000000000000000000000100000000
256<<3	00000000000000000000100000000000
-256	11111111111111111111111100000000
-256<<3	11111111111111111111100000000000

运算符与表达式

- 位逻辑运算符

```
~ 01001111  (79)
-----------
  10110000  (-80)

  00101101  (45)
^ 01001111  (79)
-----------
  01100010  (98)

  00101101  (45)
& 01001111  (79)
-----------
  00001101  (13)

  00101101  (45)
| 01001111  (79)
-----------
  01101111  (111)
```

运算符与表达式

❖数据的类型转换

- “拓宽类型”是指把值范围小类型的数据转换成值范围大类型的数据。
- “缩窄类型”是指把值范围大类型的数据转换成值范围小类型的数据。
- 类型转换
  - **自动转换：**

```
int a = 10; long b = a;
```

  - **强制转换：**

```
long b = 10; int a = (int)b;
```

运算符与表达式

自动转换

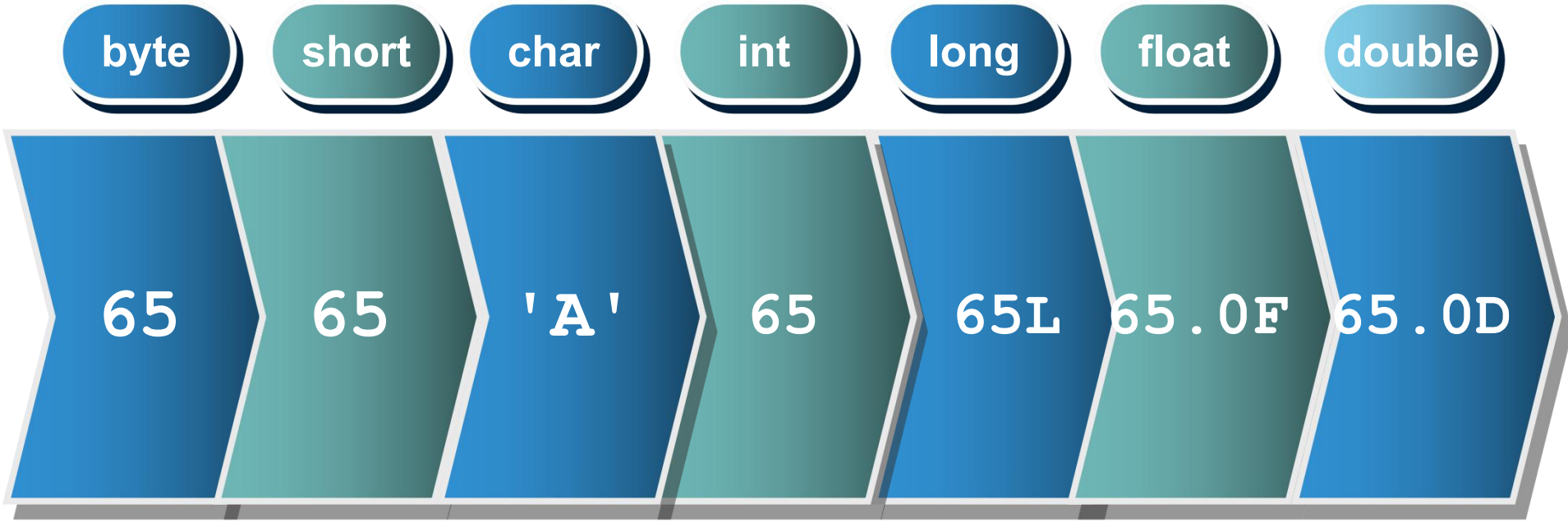
拓宽类型：
把值范围小类型的数据转换成值范围大类型的数据。

强制转换

缩窄类型：
把值范围大类型的数据转换成值范围小类型的数据。

运算符与表达式

数据类型自动转换示例



运算符与表达式

❖条件运算符

- 由?和:组成的三目运算符“?:”称为条件运算符。
  例： (a > b)? a : b
- 它的格式为：

  expreBool?expression1:expression2
  - **expreBool**表达式是boolean类型。
  - expression1和expression2表达式是相同类型。
  - 当**expreBool**为**true**时，取expression1的值；
  - 当**expreBool**为**false**时，取expression2的值。

思考问题

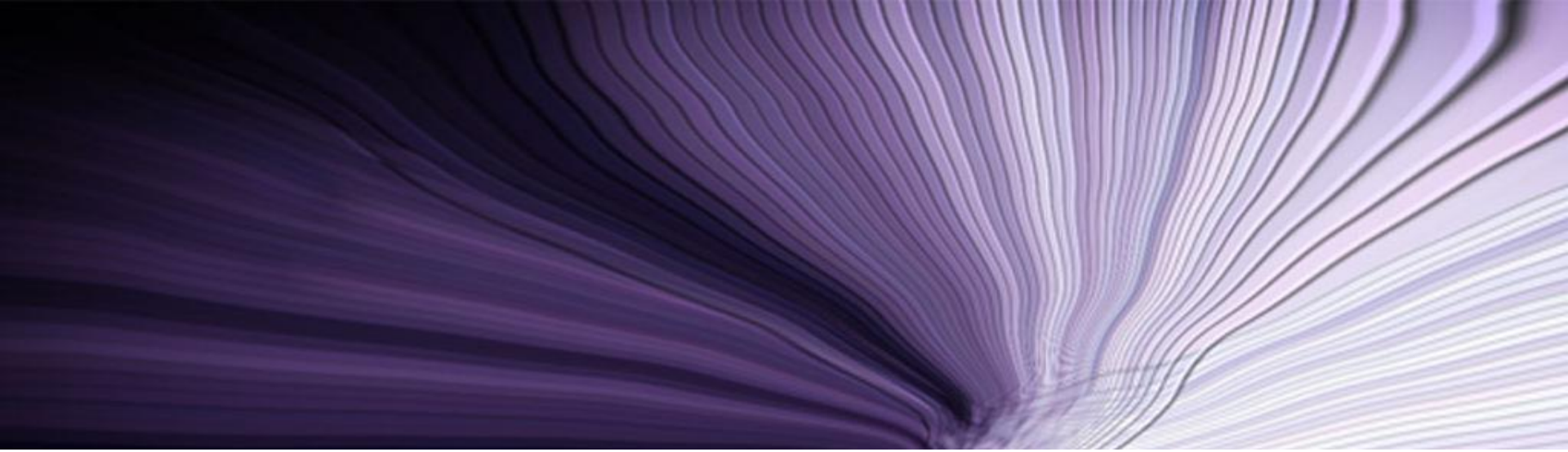
1. Java程序结构有哪几部分组成？
2. Java语言的字符集与C语言的有什么不同？
3. Java语言中的数据类型分为哪几种，它们可使用的操作符有哪些？

作业

1. 在Internet网络上收集使用Java技术方面的方案，并归纳整理出Java技术综述。

2. 理清本章Java言基础相关知识。
字符集
数据类型与变量
运算符与表达式
类型转换

3. 本章习题：
1-7题



Q&A

电子科技大学信息与软件工程学院