类型转换

类型转换

- 数据有不同的类型，不同类型数据之间进行混合运算时必然涉及到类型的转换问题。
- C++转换方式：
  - 自动类型转换（隐式转换）：遵循一定的规则，由编译系统自动完成。
  - 强制类型转换：把表达式的运算结果强制转换成所需的数据类型。
- C++自动执行很多类型转换：
  - 将一种算数类型的值赋值给另一种算数类型的变量时；
  - 表达式中包含不同类型时；
  - 将参数传递给函数时。

```
double d = 5;
```

```
double d = 2.0;
int i = 10;
double x = d + i;
```

初始化和赋值进行的转换

- 值赋值给取值范围更大的类型：

  short s = 12;

  int i = s;

- 值赋值给取值范围小的类型：

  douhle d = 3.1415926;

  float f = d;

潜在的数值转换问题

转换	潜在的问题
较大的浮点类型转换为较小的浮点类型	精度降低，值可能超出目标类型的取值范围，结果不确定。
浮点类型转换为整型	小数部分丢失，值可能超出目标类型的取值范围，结果不确定。
较大的整数类型转换为较小的整数类型	值可能超出目标类型的取值范围，通常只赋值右边的字节。

- 0赋值给bool变量时，将被转换为false，非零值转换为true

表达式中的转换

- 同一个表达式中包含多种不同的算数类型时，C++将执行两种自动转换：
  - 首先，一些类型在出现时会自动转换；
  - 其次，有些类型在与其他类型同时出现在表达式中时将被转换。
- 类型出现时的自动转换：
  - bool、char、unsigned char、signed char、short值转换为int，true转换为1，false转换为0，这些被称为整型提升。
  - 如果short比int短，则unsigned short转换为int，如果长度相同，则unsigned short转换为unsigned int，从而确保在对unsigned short提升时不会丢失数据。

表达式中的转换

- 与其他类型运算时的转换：
  - 当运算涉及两种类型时，较小的类型将被转换为较大的类型。（如int类型和float相加时，将int转换为float）
- 编译器通过校验表来确定在表达式中执行的转换，C++11校验表顺序：
  - 如果有一个操作数类型为long double，则将另一个操作数转换为long double。
  - 否则，如果有一个操作数类型为double，则将另一个操作数转换为double。
  - 否则，如果有一个操作数类型为float，则将另一个操作数转换为float。
  - 否则，说明操作数都是整型，因此执行整型提升。
  - .....

强制类型转换

- C++允许通过强制类型转换机制显式进行类型转换。
- 强制类型转换的格式：
  - (typeName) value
  - typeName (value)

将int类型num强制转换成long类型

```
long l = (long) num;
或者
long l = long (num);
```

- 强制类型转换不会修改转换的变量本身，而是创建一个新的、指定类型的值。
- 优先级的问题：
  - (int) a + b
  - (int) (a + b)

Thanks