哈爾濱工業大學
HARBIN INSTITUTE OF TECHNOLOGY

规格严格 功夫到家

第3章 简单的算术运算和表达式

哈尔滨工业大学

计算机科学与技术学院

苏小红

sxh@hit.edu.cn

本章学习内容

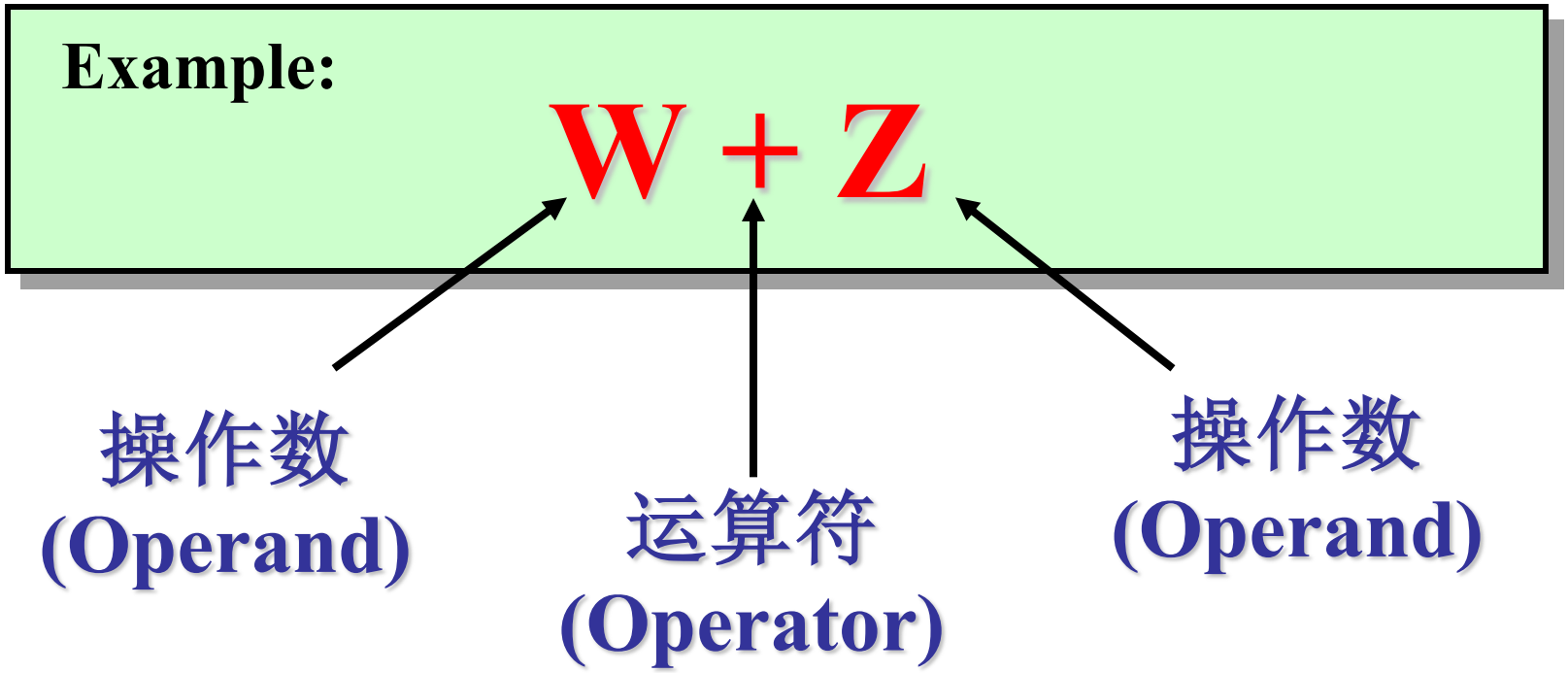
- 算术运算符
- 增1和减1运算符
- 宏常量与 const常量
- 表达式与赋值中的自动类型转换
- 强制类型转换运算符
- 常用的标准数学函数

运算符（Operator）

- 详见附录C
- 常见的运算符
  - 算术运算符
  - 赋值运算符
  - 类型强转
  - 关系运算符
  - 逻辑运算符
  - 增1和减1
  - 位运算符

3.1C运算符和表达式（Operator and Expression）

何谓运算符和操作数?



3.1.1算术运算符和表达式

Arithmetic Operators

Addition (+)

Subtraction (-)

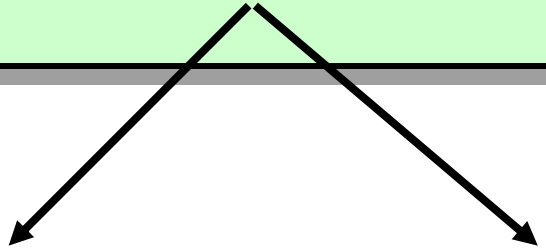
Multiplication (*)

Division (/)

Modulus (%)

除法（Division）

Example: **W / Z**



整数除法

(Integer Division)

- **W** and **Z** are integers

浮点数除法

(Floating Division)

- **W** or **Z** or both are floats

整数除法（Integer Division）

Example:

11 / 5 = 2

an integer

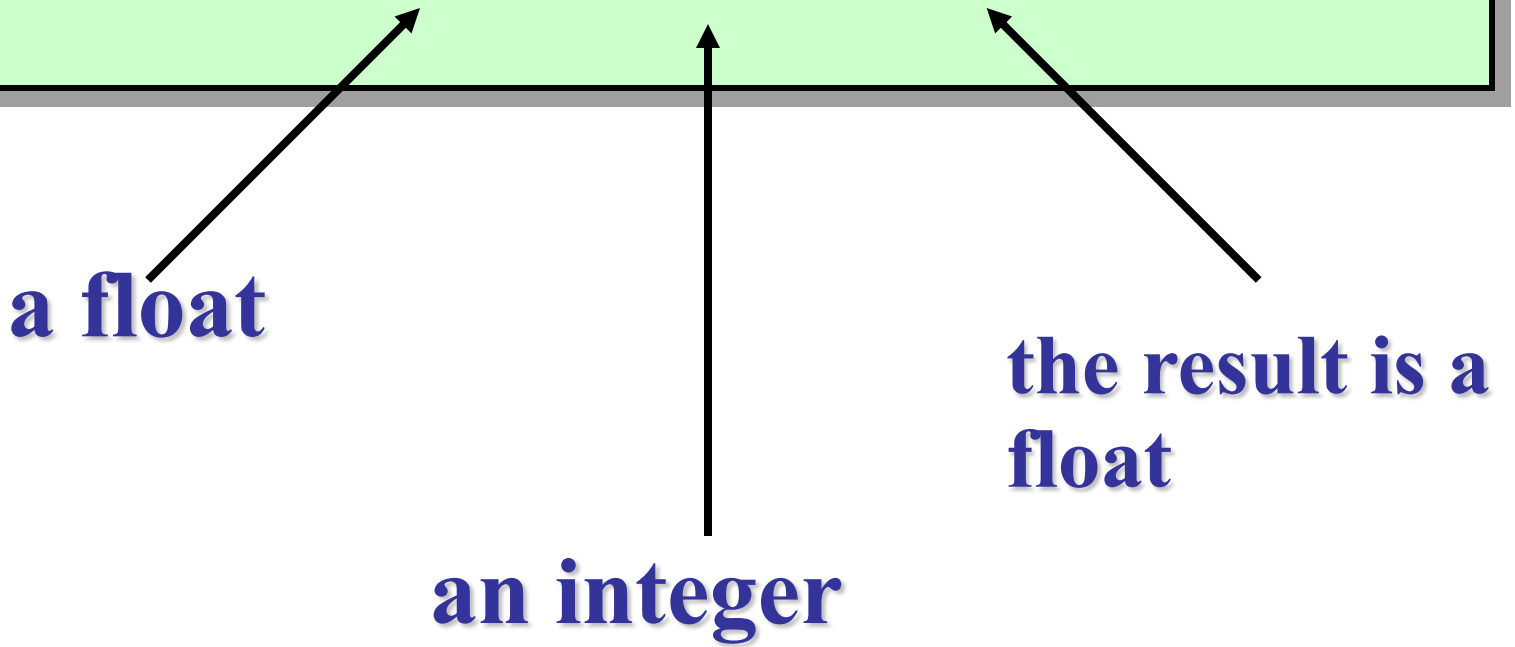
an integer

∴ the result is also an integer

实数除法（Floating Division）

Example:

11.0 / 5 = 2.2

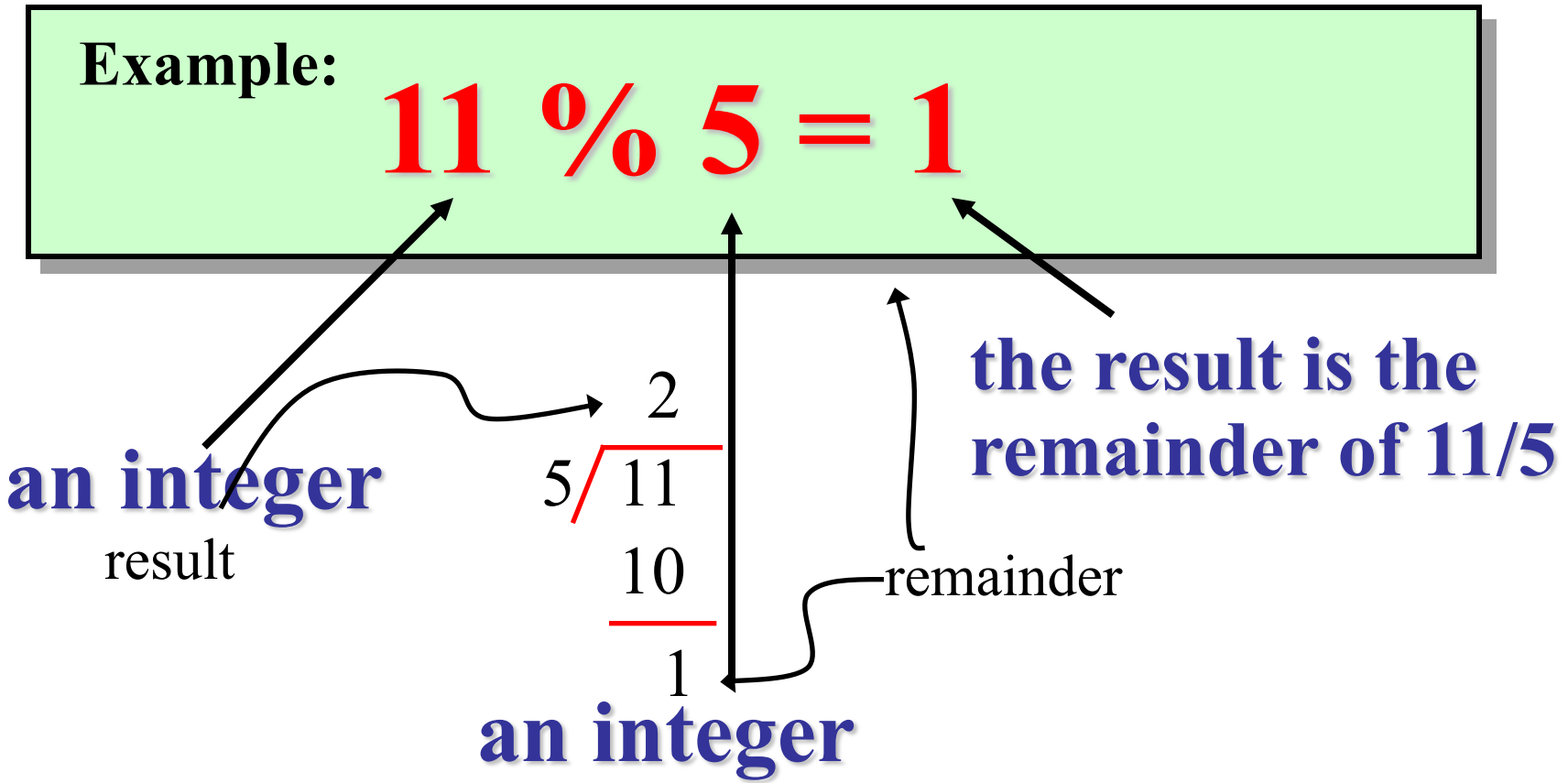


求余（Modulus）

- It returns the remainder that occurs after performing the division of 2 operands

- Rule:
  - 操作数必须是整数
  - Operands must be integers

求余（Modulus）


Example:
11 % 5 = 1
an integer
result
2
5/11
10
1
remainder
an integer
the result is the remainder of 11/5

求余（Modulus）

Example:

-11 % 5 = -1

an integer

result

-2
5 / -11
-10
-1

remainder

an integer

the result is the remainder of -11/5

求余（Modulus）

Example:

11 % -5 = 1

an integer

result

-2

-5 / 11

10

1

remainder

the result is the remainder of 11/-5

an integer

求余（Modulus）

Example:

11.0 % 5 = ?

a float

an integer

INVALID!

算术表达式（Arithmetic Expression）

- 当算术表达式包含两个或两个以上的算术运算符时
- 首先要确定运算顺序
- 所有的运算符都有一个优先级（**Order of Precedence**）

算术表达式（Arithmetic Expression）

优先级（Order of Precedence）

High:	* / %
Low:	+ -

不同优先级时的运算顺序：

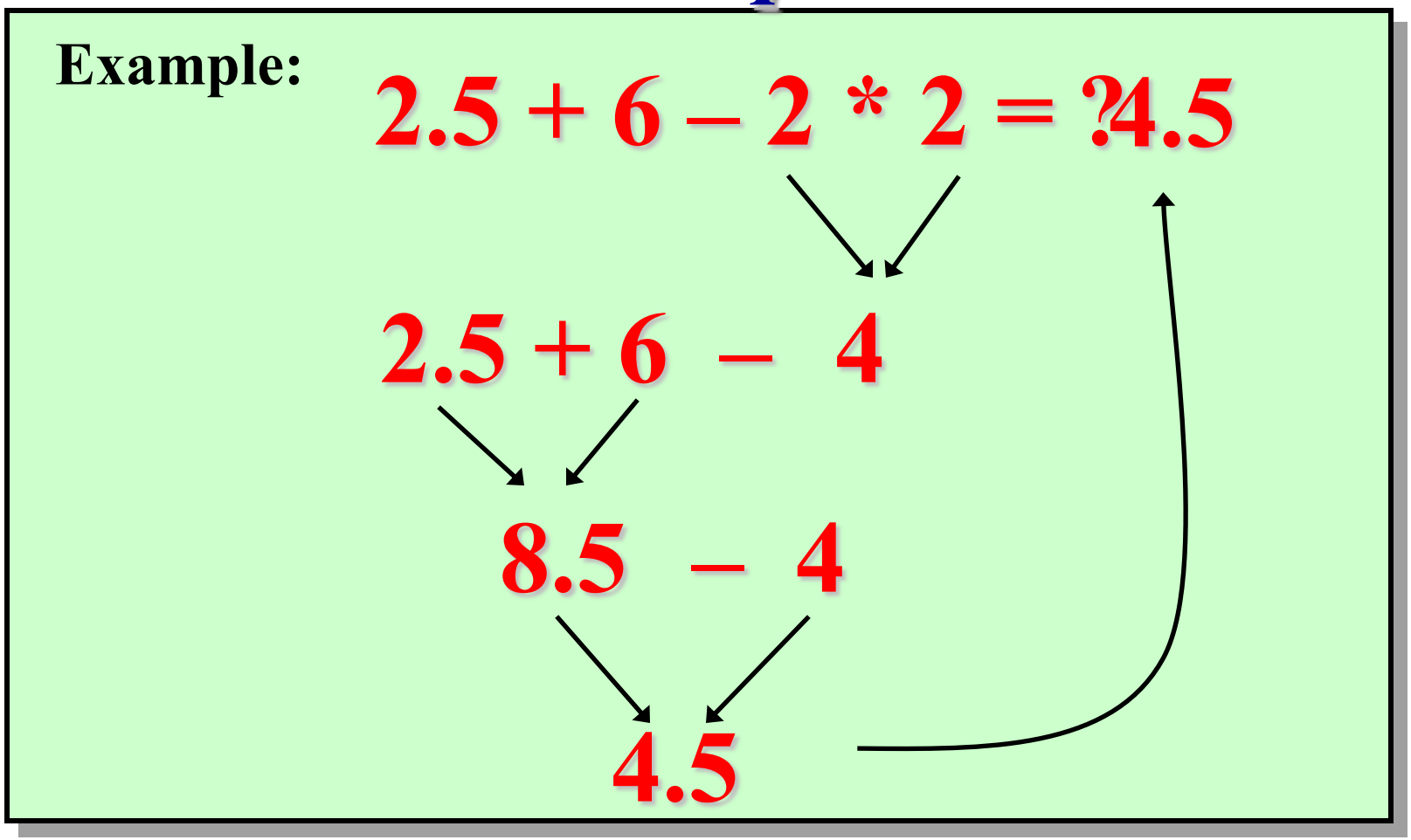
——从高到低

相同优先级时的运算顺序：

——算术运算符为左结合（从左到右）

算术表达式
（Arithmetic Expression）

Example:



算术表达式（Arithmetic Expression）

- 巧妙使用圆括号改变运算顺序
  - 从内往外运算

Example:

(9 – (3 + 2)) * 3 = ?

算术表达式
（Arithmetic Expression）

Example:

(4) * 3 = 12

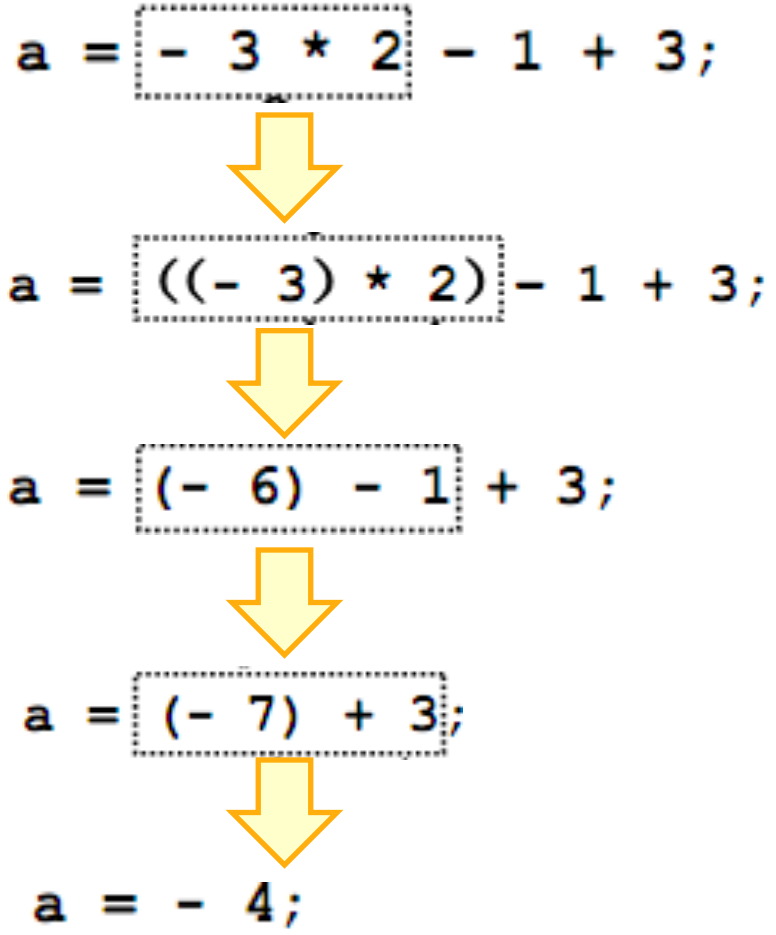
∴ (9 – (3 + 2)) * 3 = 12

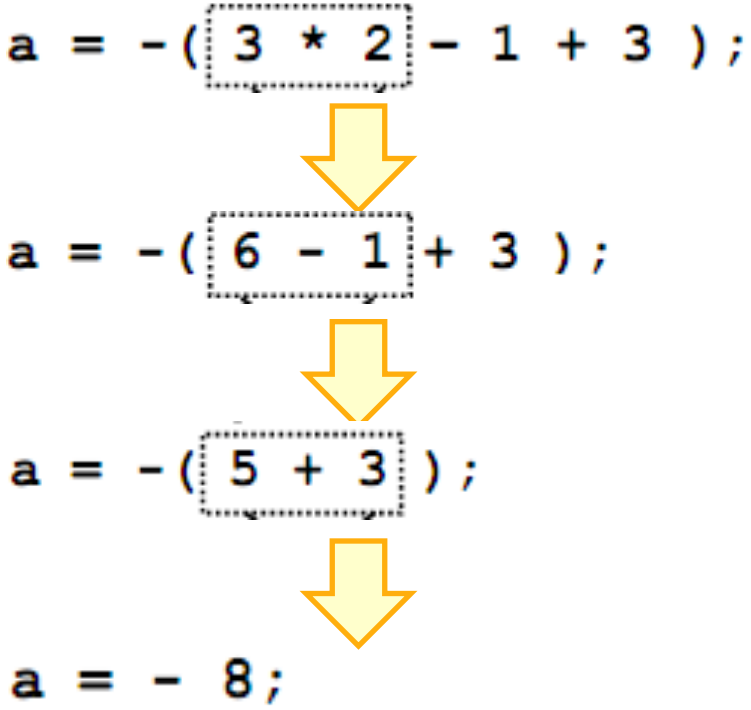
赋值语句
（Assignment Statement）

- 三种赋值形式:

- Simple——简单赋值
- Multiple——多重赋值
- Shorthand——简写的复合赋值

算术混合运算





【例3.1】计算并输出一个三位整数的个位、十位和百位数字之和

关键是如何分离个位、十位、百位数字?

153 % 10 = 3

153 / 100 = 1

153 - 1*100 = 53

53 / 10 = 5

【例3.1】计算并输出一个三位整数的个位、十位和百位数字之和

```
#include  <stdio.h>
main()
{
    int  x = 153, b0, b1, b2, sum;
    b2 = x / 100;
    b1 = (x - b2 * 100) / 10;
    b0 = x % 10;
    sum = b2 + b1 + b0;
    printf("b2=%d, b1=%d, b0=%d, sum=%d\n", b2, b1, b0, sum);
}
```

```
b2=1, b1=5, b0=3, sum=9
```

变量的赋值

简单赋值（Simple Assignment）：

变量　＝　表达式；

多重赋值（Multiple Assignment）：

变量1 ＝变量2 ＝ 表达式；

3.1.2复合的赋值运算符（Combined Assignment Operators）

Syntax:

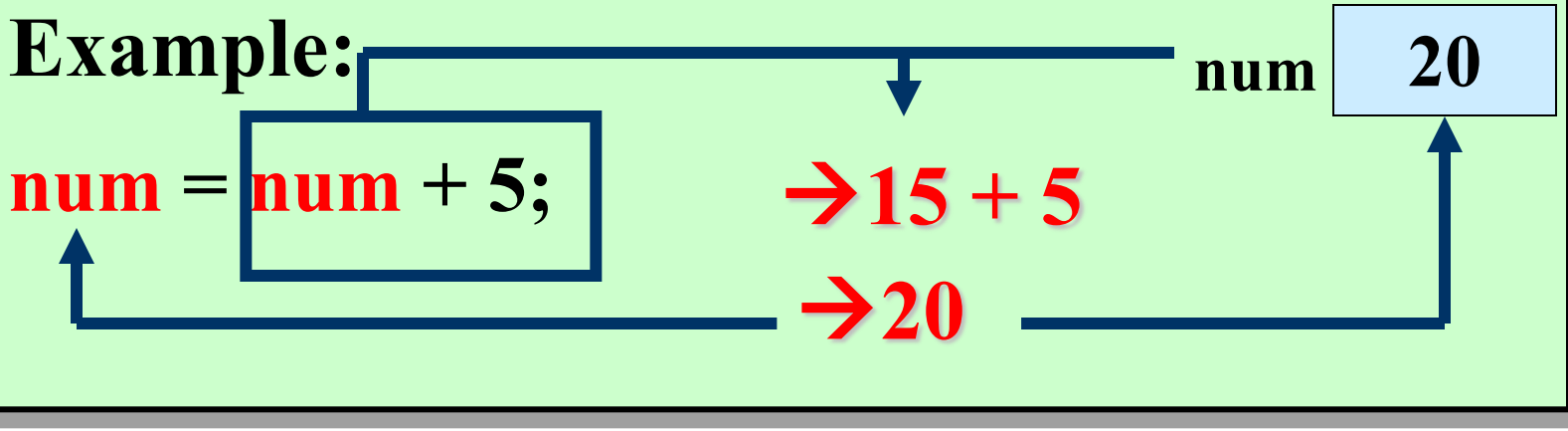
变量x = 变量x 运算符*op* 表达式；

变量x 运算符*op* = 表达式;

这种形式看起来更直观，且执行效率一般也更高一些

3.1.2复合的赋值运算符（Combined Assignment Operators）

Example:



Example:

num += 5; similar to num = num + 5

shorthand assignment operator

已知 int a = 3;

执行 a += a -= a * a 后，变量a的值？

```
a += a -= a * a
a += a -= 9
a += -6
a = -12
```

-12

执行 a += a -= a *= a 后，变量a的值？

```
a += a -= a *= a
a += a -= 9
a += 0
a = 0
```

0

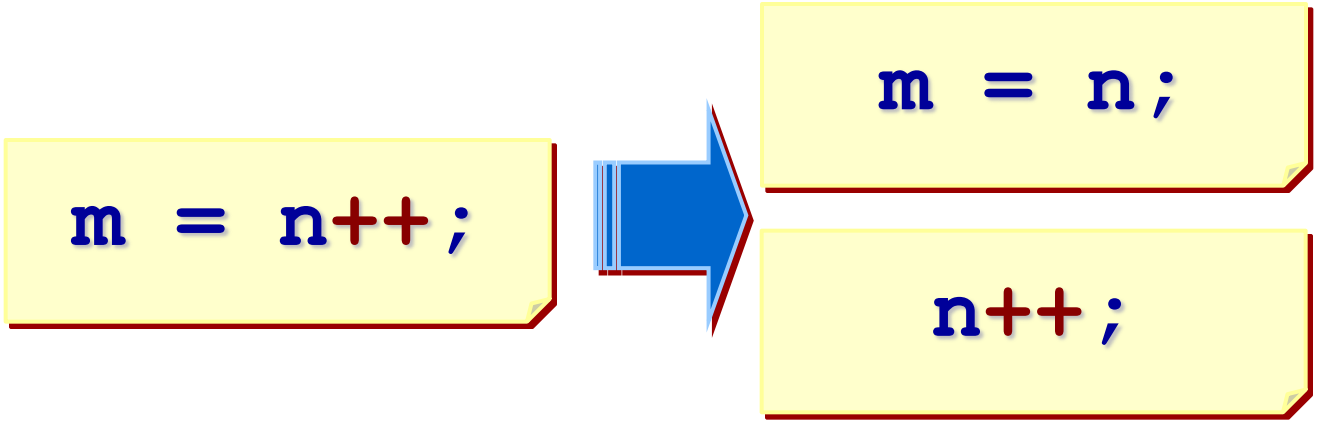
3.1.2复合的赋值运算符（Combined Assignment Operators）

Operation	Examples of expression	Description
+=	num += 5;	num = num + 5;
-=	num -= 5;	num = num - 5;
*=	num *= 5;	num = num * 5;
/=	num /= 5;	num = num / 5;
%=	num %= 5;	num = num % 5;

简写的复合赋值（Shorthand Assignment）

3.1.3增1和减1运算符 (Increment and Decrement)

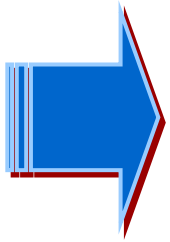
- **n++，n--，++n，--n**
  - ++让参与运算的变量加1，--让参与运算的变量减1
  - 作为后缀(postfix)运算符时，先取n的值，然后加/减1



3.1.3增1和减1运算符 (Increment and Decrement)

- n++，n--，++n，--n
  - ++让参与运算的变量加1，--让参与运算的变量减1
  - 作为后缀(postfix)运算符时，先取n的值，然后加/减1
  - 作为前缀(prefix)运算符时，先加/减1，然后取n的值

```
m = ++n;
```



```
n++;
m = n;
```

前缀（Prefix）增1和减1运算符

Example:

```
j = ++i - 2
```

similar to

```
i = i + 1;
j = i - 2;
```

i	6
j	4

后缀（Postfix）增1和减1运算符

Example:

```
j = i++ - 2
```

similar to

```
j = i - 2;
i = i + 1;
```

i	6
j	3

后缀（Postfix）增1和减1运算符

```
int a=3;
printf("%d", -a++);
```

similar to

```
printf("%d", -a);
a = a + 1;
```

a | 4

3.1.3 增1和减1运算符 (Increment and Decrement)

良好的程序设计风格提倡：

- 在一行语句中，一个变量只能出现一次增1或者减1运算
  - 过多的增1和减1运算混合，不仅可读性差，而且因为编译器实现的方法不同，导致不同编译器产生不同的运行结果

【例3.2】计算圆的周长和面积

```
#include  <stdio.h>
main()
{
    double r = 5.3;
    printf("circumference = %f\n", 2*3.14159*r);
    printf("area = %f\n", 3.14159*r*r);
}
```

```
circumference = 33.300854
area = 88.247263
```

【例3.3】计算圆的周长和面积

```
#include  <stdio.h>
main()
{
    double r;
    printf("Input r:");
    scanf("%lf", &r);
    printf("circumference = %f\n", 2*3.14159*r);
    printf("area = %f\n", 3.14159*r*r);
}
```

在程序中直接使用的常数称为幻数(Magic Number)

```
Input r: 5.3↙
circumference = 33.300854
area = 88.247263
```

使用幻数存在的问题？

- 假如直接使用常数，会有什么影响?
  - 程序的可读性变差
  - 容易发生书写错误
  - 当常数需要改变时，要修改所有引用它的代码，工作量大，还可能有遗漏
- 解决方案：
  - 避免使用幻数
  - 把幻数定义为常量（宏常量、const常量……）

3.2宏常量与宏替换

#define 标识符 字符串

- 宏常量（Macro constant）
  - 也称符号常量
  - 一般采用全大写字母表示
- 宏定义不是语句，而是一种编译预处理命令

【例3.4】计算圆的周长和面积

```
#include  <stdio.h>
main()
{
    double r;
    printf("Input r:");
    scanf("%lf", &r);
    printf("circumference = %f\n", 2*PI*r);
    printf("area = %f\n", PI*r*r);
}
```

计算圆的周长和面积

```
#include  <stdio.h>
#define  PI  3.14159;
#define  R   5.3;
main()
{
      printf("area = %f\n", PI * R * R);
      printf("circumference = %f\n", 2 * PI * R);
}
```

语法错误

相当于执行

```
#include  <stdio.h>
main()
{
      printf("area = %f\n", 3.14159;*5.3;*5.3;);
      printf("circumference = %f\n", 2*3.14159;*5.3;);
}
```

3.3 const常量

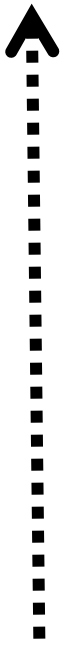
【例3.5】

```
#include  <stdio.h>
main()
{
    const double PI = 3.14159;
    double r;
    printf("Input r:");
    scanf("%lf", &r);
    printf("circumference = %f\n", 2*PI*r);
    printf("area = %f\n", PI*r*r);
}
```

- const常量与宏常量相比的优点是什么?
  - const常量有数据类型
  - 某些集成化调试工具可以对const常量进行调试

表达式与赋值中的自动类型转换

- 相同类型数据的运算结果，还是该类型
- 不同类型数据的运算结果，是两种类型中取值范围大的那种
  - 先做 **short, char → int**
  - 然后按下述规则转换
  - **long double >**
  - **double >**
  - **float >**
  - **unsigned long >**
  - **long >**
  - **unsigned int >**
  - **int**



表达式与赋值中的自动类型转换

- 取值范围小的类型赋值给取值范围大的类型是安全的
- 反之是不安全的
  - 若大类型的值在小类型能容纳的范围之内，则平安无事
    但是浮点数转为整数，会丢失小数部分，而非四舍五入
  - 反之转换后的结果必然是错误的，具体结果与机器和实现方式有关
  - 避免如此使用，好的编译器会发出警告

表达式与赋值中的自动类型转换

- 【例 3.6】下面程序演示了赋值中的类型转换

```
1   #include  <stdio.h>
2   main()
3   {
4       int       n = 256;
5       float     f = 3.6;
6       double    d = 2.5;
7       n = f;
8       f = n;
9       d = f;
10      printf("n = %d\n", n);
11      printf("f = %f\n", f);
12      printf("d = %f\n", d);
13  }
```

```
n = 3
f = 3.000000
d = 3.000000
```

3.4自动类型转换与强制类型转换运算符

- 强转（Casting）可以消除从大到小的警告
- 通过下面方式把表达式的值转为任意类型

(类型) 表达式

Example:

```
int x = 10;
float y;
y = (float)x;
```

→ (float)10

→ 10.000000

不改变x

x 10

y 10.000000

3.4 自动类型转换与强制类型转换运算符

两个整数运算的结果还是整数，不是浮点数

Example:

```
int total, number;
float average;
…
average = total / number;
```

total 15

number 2

→ 15 / 2

→ 7

average 7.000000

3.4 自动类型转换与强制类型转换运算符

Example:

```
int total, number;
float average;
…
average = (float)total / number;
```

total	15
number	2
average	7.500000

→ 15.000000 / 2

→ 7.500000

【例3.7】演示类型强转运算符

```
1   #include  <stdio.h>
2   main()
3   {
4       int   m = 5;
5       printf("m/2=%d\n", m/2);
6       printf("(float)(m/2) = %f\n", (float)(m/2));
7       printf("(float)m/2 = %f\n", (float)m/2);
8       printf("m = %d\n", m);
9   }
```

```
m/2 = 2
(float)(m/2) = 2.000000
(float)m/2 = 2.500000
m = 5
```

常用的标准数学函数

函　数　名	功　　能
sqrt(x)	计算 x 的平方根，x 应大于等于 0
fabs(x)	计算 x 的绝对值
log(x)	计算 ln x 的值，x 应大于 0
log10(x)	计算 lg x 的值，x 应大于 0

函　数　名	功　　能
exp(x)	计算e^x的值
pow(x,y)	计算x^y的值
sin(x)	计算 sin x 的值，x 为弧度值，而非角度值
cos(x)	计算 cos x 的值，x 为弧度值，而非角度值

【例3.8】计算三角形面积

$$\text{area} = \sqrt{s(s-a)(s-b)(s-c)}$$

$$s = \frac{1}{2}(a + b + c)$$

```
area = sqrt(s * (s - a) * (s - b) * (s - c))
area = sqrt(s(s-a)(s-b)(s-c))
s = 0.5 * (a + b + c)
s = 1.0/2 * (a + b + c)
s = (a + b + c) / 2.0
s = (float)(a + b + c) / 2
s = 1/2 * (a + b + c)
s = (float)((a + b + c) / 2)
```

【例3.8】计算三角形面积

```
 1  #include <stdio.h>
 2  #include  <math.h>
 3  main()
 4  {
 5      float  a, b, c, s, area;
 6      printf("Input a,b,c:");
 7      scanf("%f,%f,%f", &a, &b, &c);
 8      s = (float)(a + b + c) / 2;
 9      area = sqrt(s * (s - a) * (s - b) * (s - c));
10      printf("area = %f\n", area);
11  }
```

```
Input a,b,c:3,4,5↙
area = 6.000000
```

Questions and answers