面向对象技术与编程

C++ How to Program（9th）

西安财经学院 信息学院

本章内容

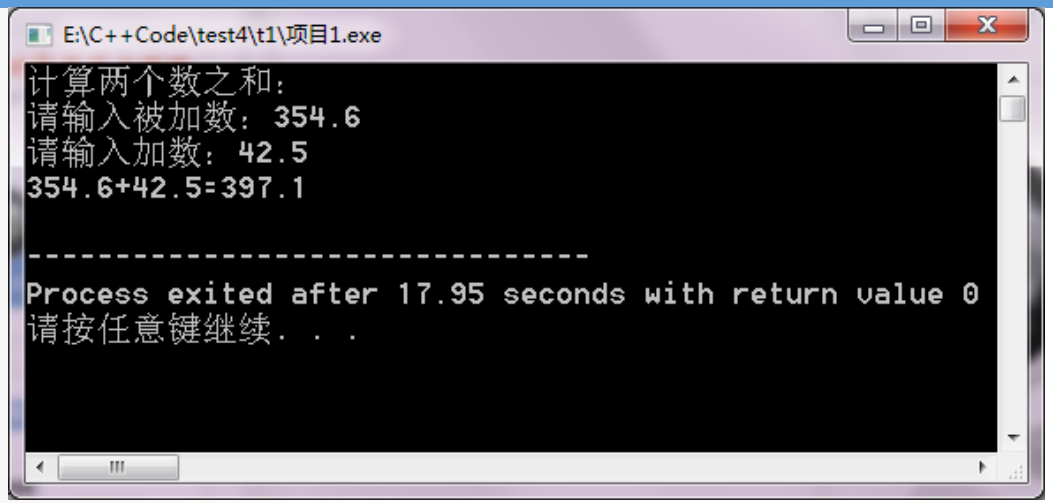
1. 内存
2. 常量与变量
3. 数据类型
4. 运算符
5. 数据类型强制转换

6

❖ 前节回顾

程序示例演示：

```
#include <iostream>
using namespace std;
int main() {
        double x=0,y=1,s;
        cout<<"计算两个数之和："<<endl;
        cout<<"请输入被加数：";
        cin>>x;
        cout<<"请输入加数：";
        cin>>y;
        s=x+y;
        cout<<x<<"+"<<y<<"="<<s<<endl;
        return 0;
}
```



- 程序编写完成后为什么不直接让计算机执行，为什么还要编译？

- 编译器除了将高级语言翻译成绩机器语言，还有什么作用？编译通过了，编程是否就顺利完成了？

1.内存的概念

- 冯·诺依曼体系结构

计算机由运算器、控制器、存储器、输入设备和输出设备五大部分组成。需要把程序加载到存储器中、采用程序控制来运行。

存储器即内存，是用来存储计算机需要执行的程序和数据的地方。

- 数据在内存中是如何存储、如何访问的?
- 计算机如何快速找到数据的存放位置?
- 为什么要定义变量?

- 计算机运行的程序和所使用的数据存储在内存中，内存中的每一个存储单元都有一个编号即内存地址，通过内存地址可以快速找到某个单元中存放的数据。

- 内存单元地址表述复杂，使用不方便，变量时某个内存单元临时的替代符号（别名），通过变量名来访问和存取内存单元中的内容更方便、简捷。

- 既然已经有硬盘存储数据了，为什么还要内存？
- 直接在硬盘上存取数据速度太慢，为了加快数据的存取速度，预先把要执行的程序指令和数据装载到存取速度快得多的内存中去，可以有效地缓解CPU的执行指令的效率与数据存取之间速度匹配的问题，进一步提高计算机的工作效率。

- 内存中各个存储空间是如何定位的？

2.变量

- C++中的数据类型分为基本类型、逻辑型和构造类型。构造类型是编程人员更据需要自定义的复杂数据类型，其中往往由若干个不同的基本数据类型的成员组成。

- 如何确定构造类型变量在内存中占用存储空间的大小?

- 构造数据类型有多个不同类型的基本成员变量组成，其变量所占内存空间大小是所有成员变量所占空间大小的总和。但共用体除外，共用体变量所占空间大小有所有成员变量中占用空间最大的成员决定。

1. 常量与变量

```
#include <iostream>
using namespace std;
void main()
{
    const int PRICE=30;
    int num,total;
    float v ,r ,h;
    num=10;
    total=num*PRICE;
    cout<<total <<endl;
    r=2.5;
    h=3.2;
    v=3.14159*r*r*h;
    cout<< v <<endl;
}
```

符号常量

变量先声明后使用

变量

常量

Constants（常量）

- As the name suggests, a constants does NOT change its value in a program.
- Integer(整型): 100, -3, 0
- Floating-point(浮点型、实型)： 0.34,-12.3
- Character(字符型): ‘x’ ,‘X’, ‘*’, ‘9’
- String(字符串型): “abc” ,“A100”, “9”

（1）整型常量

- **十进制**整型常量：由数字0~9和正负号表示。

  如 123,-456,0，而024，25o错误。

- **八进制**整型常量：以数字0为前缀,后跟数字0~7表示。如 0123,011。

- **十六进制**整型常量：由0x或0X开头,后跟0~9,a~f,A~F表示。如 0x123,0Xff。

问题：

135= (135)$_{10}$

0135 = (93)$_{10}$

0x135 = (309)$_{10}$

续

注意：

- 在程序中是根据前缀来区分各种进制数的，一定要前缀正确。
- 八进制与十六进制一般只表示正数。
- 长整型常数的表示方法是加后缀 “L”或 “l”。
- 无符号数可用后缀 “U”或 “u”来表示。
- 常量的前后缀可同时使用以表示各种类型的数， 如：
  98L,78u,017lu,0xaau,0X1fL。

(2) 实型常量

C++中实型常量只能用十进制形式表示，有以下两种表示方式：

- 一般形式：由0~9的数字和小数点组成如：
  2.1，2.， .1， -1.456
- 指数形式：由十进制数加上阶码标志 “e”或 “E”以及阶码组成，可表示为a E n，其中a为十进制数，n为十进制整数且可以带符号，其所表示的值为$a*10^n$

注意：

- 小数点不能单独出现，如：． 是错误的。
- 指数形式表示中 “e”或 “E”两边必须有数据且后面必须为整数

（3）字符常量

- 字符常量是用单撇号括起来的单个字符或转义字符，

  如：　‘a’，　‘9’，　‘&’，　‘\n’，　‘\’，　‘\101’。

- 字符常量的值为该字符的ASCII码值，如　‘a’值为97，　‘\n’　值为10，　‘0’值为48。

注意：

- 字符常量只能用单撇号括起来，不能用其他符号。
- 字符常量只能是单个字符，不能是字符串。
- 字符可以是字符集中任意字符，其中的数字字符与数字整型常量是不同的，例如：　‘6’和6的不同。
- 转义字符：以反斜杠　‘\’开头的一个或几个字符，具有特定的含义，不同于字符原有的含义。

  如：　‘\n’中的n不代表字符n，而作为换行符。

续

字符形式	转义后功能描述
\n	换行
\t	横向跳格
\b	退格
\r	回车
\\	反斜杠字符
\’	单撇号字符
\ddd	1-3位八进制数所代表的字符
\xhh	1-2位十六进制数所代表的字符

（4）字符串常量

- **字符串常量**是由一对双撇号括起来的字符序列，

  如“hello”，“a”，“19”，“”，“ ”，“0&$”。

- **字符串常量的长度**：字符串中的字符个数，如“”长度为0。
- 每个字符串串尾自动加一个‘\0’作为字符串结束标志。

如“hello”在内存中为

h	e	l	l	o	\0

而空串“”在内存中为

\0

注意：

- 一个字符常量占1个字节的内存空间，而字符串常量所占内存字节数等于其长度加1。

- 可把一个字符常量赋值给一个字符变量，但不能把一个字符串常量赋值给一个字符变量。

例如： char m;

m= “A” ; ✗

（5）符号常量

- **符号常量**指用标识符定义一个直接常量，它的值在程序中始终不变，也叫**常变量**。
- 符号常量在使用之前一定要先声明。符号常量声明语句的形式：

  const 数据类型说明符　常量名=常量值;

  或　数据类型说明符 const 常量名=常量值;

例如，我们可以声明一个代表圆周率的符号常量：

```
const  float pi=3.1415;
```

符号常量在声明时一定要赋初值，而在程序中间不能改变其值。

```
Const  float pi;
Pi=3.1415;
```

符号常量的定义

1) 使用#define命令定义符号常量

用一个标识符来代表一个常量，通过宏定义预处理指令来实现。

格式: # define 标识符 常量

如：#define PI 3.14159

由用户命名的标识符是符号常量名。一般大写。一旦定义，在程序中凡是出现常量的地方均可用符号常量名来代替。对使用了符号常量的程序在编译前会以实际常量替代符号常量。

例如在主函数中出现：float x=PI*10*10，则编译时变为

float x=3.14159*10*10.

★ 注意宏替换后的优先顺序

如 #define A1 2+5

int x=A1*A1;

int x=2+5*2+5;

2) 符号常量定义语句

定义格式：const 数据类型 符号常量=表达式;

如： const float PI=3.1415926;

注意

1、符号常量习惯上用大写字母表示。

2、符号常量必须在定义时初始化，且以后在程序中不能进行修改。

3、符号常量定义中的表达式，只能是常量表达式（运算结果为常量，表达式中允许变量）。

4、使用const语句定义符号常量带有数据类型，以便系统进行类型检查。且可计算初值表达式。故比define命令优越。

判断下列语句是否正确

(1) const float PI; PI=3.1415926;（错） X

(2) const float PI=Π;（错） X

(3) const int number=max(15,23);（对）

例：

```
int x=6;

#define NU1 x+5        // 末尾缺少;
const int NU=x+5;
void main()
{
    cout<<NU*7<<´,´<<NU1*7<<endl;
}
```

结果：77,41

Variables（变量）

- Unlike a constant, a variable can vary its values in a program, and a variable must be defined before it can be used.
  (与常量不同，变量是指其值在程序内可以变化的量。变量在使用之前需要先被定义)
- A variable is defined by giving it a data type and a name.
  (变量定义由变量名字与变量数据类型共同组成)

Program Example

```
1   main()
2   {
3       int v1;
4       float v2;
5       char v3;
6       v1=65;
7       v2=-18.23;
8       v3='A';
9   }
```

C++ program start with the line

Line 3,4 and 5 define three variables

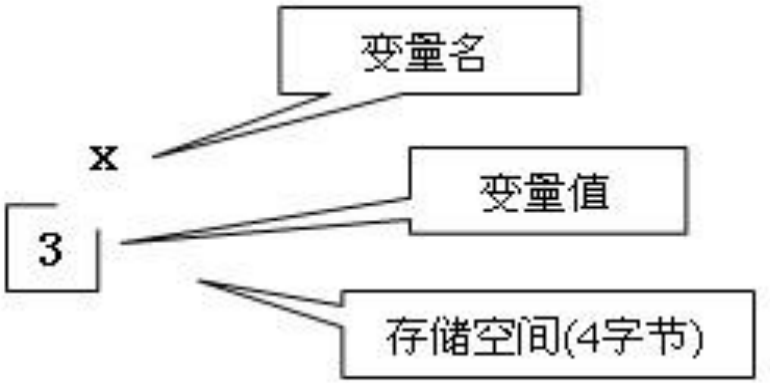
Line 6,7 and 8 assign a value to each variables

Variables and Assignments

- Variables are like small `blackboards`
  - We can write a number on them
  - We can change the number
  - We can erase the number
- Variables are names for `memory locations`
  - We can write a value in them
  - We can change the value stored there
  - We cannot erase the memory location

变量

- 变量名、变量类型和存储单元是变量的三要素。进行变量声明后，计算机系统会为声明的变量分配存储空间,用以存放数据。
- 变量名实际上是一个符号地址。在程序中对变量的赋值和取值操作实际上是通过变量名找到相应的内存地址,然后从对应的存储空间中读取数据。
- 如：int x=3;



1. 变量的声明

格式：数据类型 变量名列表;

如： int a; float x, y; char ch;

int m=4 , n=21 ; // 声明时初始化

char ch=´A´ ;

2. 变量的赋值

格式: 变量 = 表达式;

如: x = 6 ;

y = x+2 ;

a = sqrt(b) ;

y = x = x+2 ;

Identifiers(标识符)

- Variables names are called identifiers（标识符）
- Choosing variable names
  - Use meaningful names that represent data to be stored (起一个有意义的名字来代表存储的数据)
  - First character must be
    - a letter
    - the underscore character（下划线）
  - Remaining characters must be
    - letters
    - numbers
    - underscore character（下划线）
  - Case sensitive（大小写敏感）
  - Cannot be a C++ Keyword（关键字）

C++字符集

- 大小写的英文字母：A~Z，a~z
- 数字字符：0~9
- 特殊字符：

空格　!　#　%　^　&　*
_(下划线)　+　=　-　~　<　>
/　\　'　"　;　.　,　()
[]　{}

词法记号

- 关键字　C++预定义的单词
- 标识符　程序员声明的单词，它命名程序正文中的一些实体
- 操作符　用于实现各种运算的符号
- 分隔符 ()　{}　,　:　;
  用于分隔各个词法记号或程序正文
- 空白符　空格、制表符（TAB键产生的字符）、换行符、（Enter键所产生的字符）和注释的总称

标识符的构成规则

- 以大写字母、小写字母或下划线(_)开始。
- 可以由以大写字母、小写字母、下划线(_)或数字0~9组成。
- 大写字母和小写字母代表不同的标识符。

- 例如：
  - 合法标识符：_22A，lea_1，avg3，day，BCde43xyw8
  - 不合法标识符：M.J.YORK，$_238，#xy，a*b，8Tea

注意：

- ✓ 在C++语言中，大小写字母不等效。因此，a和A，l和i，Sum和sum，分别是两个不同的标识符。(大小写敏感)
- ✓ 标识符长度不要超过32个字符
- ✓ 标识符最好能见名知意，不宜混淆，如l与i、o与0

Initializing Variables(初始化)

- Declaring a variable does not give it a value
  - Giving a variable its first value is initializing the variable
- Variables are initialized in assignment statements

```
double mpg;        // declare the variable
mpg = 26.3;        // initialize the variable
```

- Declaration and initialization can be combined using two methods(变量的声明与初始化可以同步进行)
  - Method 1

```
double mpg = 26.3, area = 0.0 , volume;
```

  - Method 2

```
double mpg(26.3),  area(0.0), volume;
```

2. Simple output to the screen

- How are the values of variables displayed on the screen?（如何将变量的值显示在屏幕上）
- This can be done with cout.

Example:

```
#include <iostream>
using namespace std;
main()
   {  int a=3;
         cout<<“Program Example!”<<endl;
         cout<<“a has the value ”<<a<<endl;
   }
```

Input and Output

- A data stream is a sequence of data
  (数据流是数据组成的一个序列)
  - Typically in the form of characters or numbers
- An input stream is data for the program to use
  - Typically originates
    - at the keyboard
    - at a file
- An output stream is the program's output
  - Destination is typically
    - the monitor
    - a file

Output using cout

- cout is an output stream sending data to the monitor(cout 是将输出流的内容发送到显示器上)
- The insertion operator "<<" inserts data into cout(<< 符号是将相关内容发送到 cout 流中)
- Example:
  cout << number_of_bars << " candy bars\n ";
  - This line sends two items to the monitor
    - The value of *number_of_bars*
    - The quoted string of characters " candy bars\n"
      - Notice the space before the 'c' in candy
      - The '\n' causes a new line to be started following the 's' in bars
    - A insertion operator is used for each item of output

Examples Using cout

- This produces the same result as the previous sample

```
cout << number_of_bars ;
cout << " candy bars\n";
```

- Here arithmetic is performed in the cout statement

```
cout << "Total cost is $" << (price + tax);
```

- Quoted strings are enclosed in double quotes ("Walter")
  - Don't use two single quotes, such as (' ')
- A blank space can also be inserted with

```
cout << " " ;
```

Include Directives(包含指令)

- Include Directives(指令) add library files to our programs
  - To make the definitions of the cin and cout available to the program:

```
#include <iostream>
```

- Using Directives include a collection of defined names
  - To make the names cin and cout available to our program:

```
using namespace std;
```

Escape Sequences(转义序列)

- Escape sequences tell the compiler to treat characters in a special way
- '\' is the escape character(转义符)
  - To create a newline in output use

    \n – cout << "\n";

    or the newer alternative

    cout << endl;
  - Other escape sequences:

    \t -- a tab

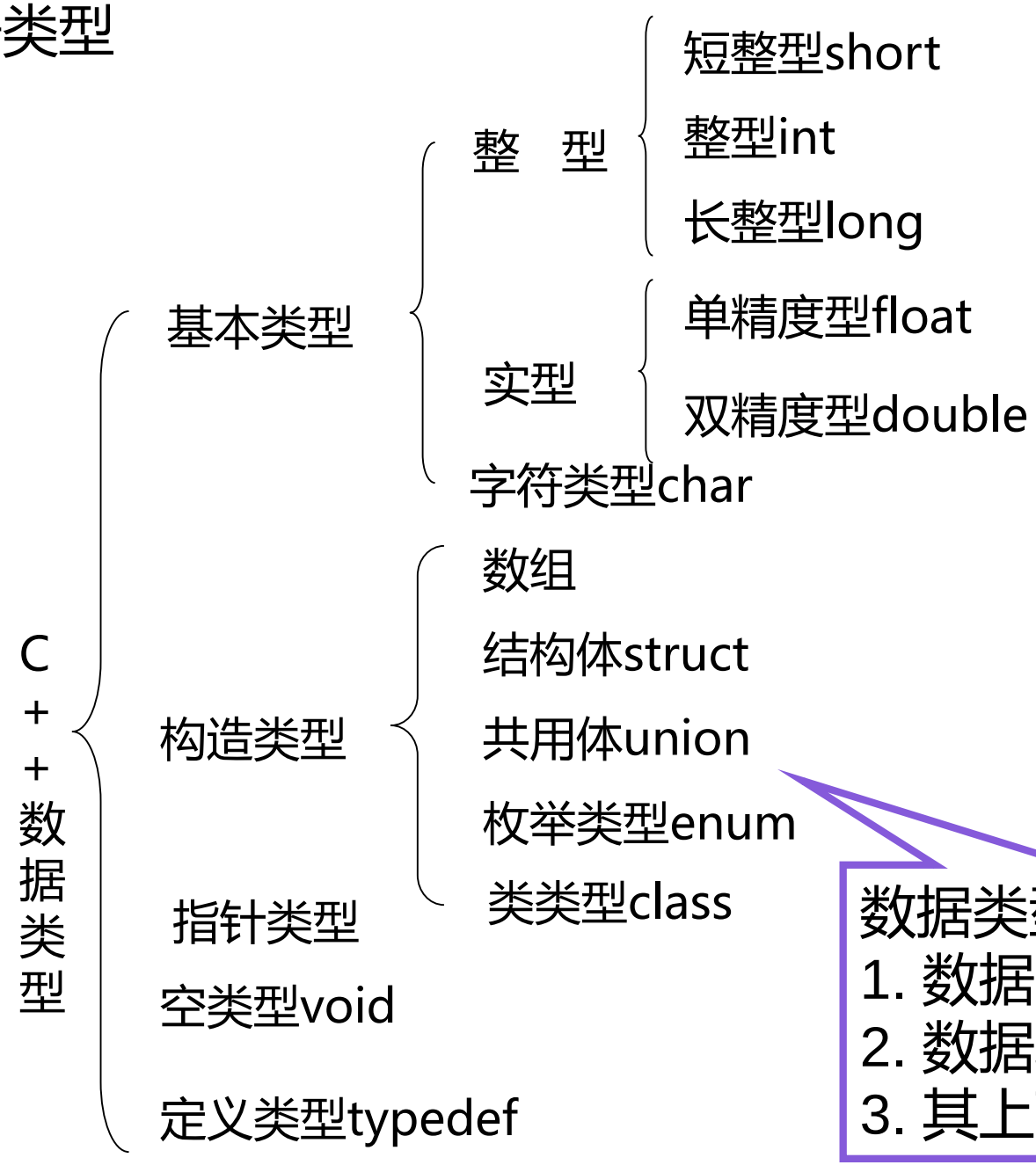
    \\ -- a backslash character

    \" -- a quote character

3. Comments(注释)

- Comments（注释）
  - Document programs
  - Improve program readability（可读性）
  - Ignored by compiler
  - Single-line comment
    - Begin with //
    - Cannot span more than one line （只能占一行）
  - C风格注释　　/*　*/
    - Can span more than one line(可以占多行)

4. 数据类型



数据类型决定：
1. 数据占内存字节数
2. 数据取值范围
3. 其上可进行的操作

数据类型修饰符：

short　　long　　unsigned　　signed

例：

- 无符号整型　unsigned int
- 无符号长整型 unsigned long
- 无符号短整型 unsigned short

说明

(1) 数据类型的描述确定了数据在内存所占空间大小，也确定了数据的表示范围及对该类型数据所进行的操作，如下表是16位机上各种基本数据类型的类型名、表示范围。

(2) 不同计算机，同一数据类型占用空间不同。如int(16位机2字节，32位机4字节)。

(3) 不同类型数据占用空间大小决定其存储范围。

(4) 计算机对内存存放同样信息的解释会因其所表示的数据类型不同而不同。

如：(01000001)B，int型 65， char型为 ´A´

至于它们之间的差异，请看下表(p24)：(16位计算机)

类型	所占位数	数的范围
int	16	-32768 ~ 32767
short [int]	16	-32768 ~ 32767
long [int]	32	-2147483648 ~ 2147483647
unsigned [int]	16	0 ~ 65535
unsigned short	16	0 ~ 65535
unsigned long	32	0 ~ 4294967295

注意

(1) 整型和实型的几种不同方式

(2) 各种数据类型的取值范围

(3) C无布尔类型（用0代表“假”，非0表“真”）。C++增添了布尔类型bool，但并非每个C++编译器都支持。

声明： bool old_enough;

(4) 用sizeof(数据类型)可以确定数据类型的字节长度。如：

cout<<“sizeof int is ”<<sizeof(int)<<endl;

(5) C++强制类型语言----要求在使用数据之前对数据的类型进行声明。

5. Operators（运算符）

- The assignment operator(赋值号)　　(=)
- Arithmetic operators　　(+ - * / %)
- Increment and decrement operators (++ --)
- Combined assignment operators
  (+= -= *= /= %=)

一般来说，C++语言的基本运算符与表达式包括：

- 算术运算符及其表达式
- 赋值运算符及其表达式
- 关系运算符及其表达式
- 逻辑运算符及其表达式
- 条件运算符及其表达式
- 逗号运算符及其表达式
- sizeof运算符

表达式就是变量、常量、函数等运算量按照一定规则和运算符连接而成的式子。

C++语言运算符

- 算术运算符（+ - * / % ++ --）
- 关系运算符（< <= == > >= !=）
- 逻辑运算符（! && ||）
- 位运算符 （<< >> ~ | ^ &）
- 赋值运算符（= 及其扩展赋值运算符）
- 条件运算符（?:）
- 逗号运算符（,）
- 指针运算符（* &）
- 求字节数运算符 （sizeof）
- 强制类型转换（类型）
- 分量运算符（. ->）
- 下标运算符（[]）
- 其它 （( ) -）

- 学习运算符应注意的几个问题：
  - 运算符的功能
  - 与运算量的关系
    - 运算量的个数（是几目或几元运算符）
    - 运算量的类型
  - 运算符的优先级别（先高后低）
  - 结合方向（在运算量两侧运算符优先级相同时）
  - 表达式值的类型（尤其不同类型数据进行运算时）

The assignment operator

- The ‘= ‘ operator in C++ is NOT an equal sign
  - 这是赋值运算符，并不是判断相等与否的符号（==）

- 赋值运算符用于赋值运算，分为简单赋值（=）、复合算术赋值（+=、-=、*=、/=、%=）和复合位运算赋值（&=、|=、^=、>>=、<<=）3类共11种。
- 相应的，由赋值运算符将一个变量和一个表达式连接起来的式子称为赋值表达式。

Combined assignment operators

- All arithmetic operators can be used this way

  `+=` `count = count + 2;` becomes
  `count += 2;`

  `*=` `bonus = bonus * 2;` becomes
  `bonus *= 2;`

  `/=` `time = time / rush_factor;` becomes
  `time /= rush_factor;`

  `%=` `remainder = remainder % (cnt1+ cnt2);` becomes
  `remainder %= (cnt1 + cnt2);`

如果赋值运算符两边的数据类型不相同，系统将自动进行数据类型的转换，把赋值符右边的类型转换成左边的类型.常见的几种转换规定：

①实型数据赋给整型变量：舍去小数部分

```
如：int i;  float j=2.72; i=j;
    则i 的值为2
```

②整型数据赋给实型变量：数值不变，以浮点形式存放

```
如：int i=2;  float j; j=i;
    则j 的值为2.0
```

③ 字符型数据赋给整型变量：将字符ASCII码放到整型变量的低8位中，高8位全为0

如：int a; char c1='A'; a=c1;

则a 的值为65

④ 整型数据赋给字符型变量：只把低8位赋给字符型变量

如：int a=322; char c2; c2=a;

因a的低8位为01000010，即十进制66，按ASCII码对应于字符B，所以c2为字符B

注意：

① 赋值运算符的优先级低于算术运算符、关系运算符和逻辑运算符（赋值运算符的优先级最低）；

如：x = 8 <1

应先求表达式8 <1的值为0，再将该值赋值给x，则x的值为0

② 赋值表达式具有右结合性；

如：a = b = 20/5

运算时先计算20/5，结果为4，将4赋值给变量b，再将4赋值给变量a，自右至左运算

- 赋值表达式中的表达式又可以是一个赋值表达式；

```
x = ( y = 20)
x = y = z = 8
x = 10 + ( y = 5 )
a = ( b = 20 ) / (c = 10)
```

- 赋值表达式左侧必须是变量，而不能为常量或表达式；

如：3 = x - 2*y ，　a + b =3　都是错误的

Results of Operators

- 运算符的运算结果类型取决于参加运算的操作数类型
  - If both operands are int, the result is int
  - If one or both operands are double, the result is double

Division of Integers

- Be careful with the division operator!
  - int / int produces an integer result
    (true for variables or numeric constants)

```
int dividend, divisor, quotient;
dividend = 5;
divisor = 3;
quotient = dividend / divisor;
```

  - The value of quotient is 1, not 1.666…
  - Integer division does not round the result, the fractional part is discarded!
    两个整数相除得一个整数。

Integer Remainders

- % operator gives the remainder from integer division

```
int dividend, divisor, remainder;
  dividend = 5;
  divisor = 3;
  remainder = dividend % divisor;
```

The value of remainder is 2

Arithmetic Expressions

- Use spacing to make expressions readable
  - 适当增加空格可以增加程序的可读性

    Which is easier to read? x+y*z or x + y * z
- Use parentheses(括号) to alter the order of operations 通过括号可以改变运算顺序
- x + y * z (y is multiplied by z first)

  (x + y) * z (x and y are added first)

逗号运算和逗号表达式

- 格式
  - 表达式1，表达式2
- 求解顺序及结果
  - 先求解1，再求解2，最终结果为表达式2的值
- 例:
  - a=3*5 , a*4　最终结果为 60

6. Type conversions and casts

- 如果一个计算表达式中有多种数据类型，其类型的精度顺序为:
- char<short<int<long<float<double
- C++ 自动将低精度的转换为高精度的

Type Compatibilities

- In general store values in variables of the same type
  - This is a type mismatch:

```
int int_variable;
int_variable = 2.99;
```

  - If your compiler allows this, int_variable will most likely contain the value 2, not 2.99

int ←→ double (part 1)

- Variables of type double should not be assigned to variables of type int

```
int int_variable;
double double_variable;
double_variable = 2.00;
int_variable = double_variable;
```

  - If allowed, int_variable contains 2, not 2.00

int ←→ double (part 2)

- Integer values can normally be stored in variables of type double

```
double double_variable;
double_variable = 2;
```

  - double_variable will contain 2.0

char ← → int

- The following actions are possible but generally not recommended!
- It is possible to store char values in integer variables

  int value = 'A';

  value will contain an integer representing 'A'
- It is possible to store int values in char variables

  char letter = 65;

bool ← → int

- The following actions are possible but generally not recommended!
- Values of type bool can be assigned to int variables
  - True is stored as 1
  - False is stored as 0
- Values of type int can be assigned to bool variables
  - Any non-zero integer is stored as true
  - Zero is stored as false

Manual conversion（强制转换）

- In addition to automatic conversion, C++ allows you to perform manual conversion with a static cast. The general format of a static cast is:

```
static_cast<type>(expression)
```

Example:

```
double num1=1.9, num2=2.9;
cout<<static_cast<int>(num1)+static_cast<int>(num2);
//将double型的变量num1和num2强制转换为int型后二者之和输出
```

在使用显式转换时应注意几个问题：

① 类型说明符和表达式都必须加括号（单个变量可以不加括号）

如: (int)(x+y) 和 (int)x + y 强制类型转换的对象是不同的：(int)(x+y)是对 (x+y)进行强制类型转换；而(int)x + y只对x进行强制类型转换

② 无论是显式转换还是隐式转换，转换得到的是所需类型的中间变量，原变量类型不变

③ 较高类型向较低类型转换时可能发生精度损失问题

- 无论是显式转换还是隐式转换，转换得到的是所需类型的中间变量，原变量类型不变

```
例：main( )
{
    float f = 6.25;
    cout<<“f1=”<< (int) f<<“f2=“<<f <<endl;
}
```

结果为：f1 =6，f2 = 6.25

- 较高类型向较低类型转换时可能发生精度损失问题