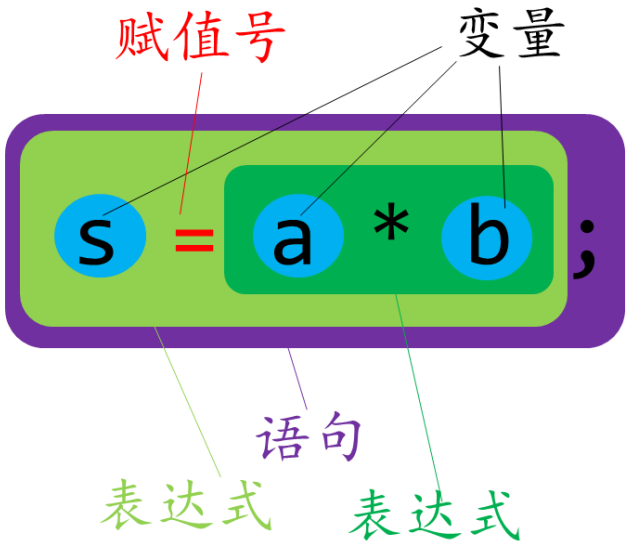
变量-存储数据的容器



变量

- 变量代表了一个存储单元，其中的值是可以改变的，因此称为变量。如游戏中玩家命的条数最初为3，当你死了一次，命减少一条，这里命的条数就是一个变量（或者说命的条数存储在一个存储单元中）。
- 一个程序中可能要使用到若干个变量，为了区别不同的变量，必须给每个变量（存储单元）取一个名（称为变量名或变量标识符），该变量存储的值称为变量的值，变量中能够存储值的类型为变量的类型。例如游戏中用于存储“命”的变量，在游戏程序中的存储命的变量名可取为life，它的类型为整型，游戏初始时这个变量的值为3。

标识符

由程序员定义的命名符

语法：以字母或下划线开始，由字母、

break main int 等

不能用于命名程序中的自定义

注意：

(1) 不能使用关键字作用户标识

Aa 和 *aa*

是两个不同的标识符

(2) C++中，字母大小写敏感；

(3) C++没有规定标识符的长度，不同编译系统有不同的识别长度；

(4) 标识符尽可能做到见文知义。

标识符

由程序员定义的命名符

语法：以字母或下划线开始，由字母、数字和下划线组成的符号串

例 判断以下标识符的正确性：

合法标识符有： **a x1 no_1 _a2c sum Name name**

不合法标识符有：**2a x+y α π a,b a&b const**

标识符

由程序员定义的命名符

语法：以字母或下划线开始，由字母、数字和下划线组成的符号串

它们是不同的标识符

例 判断以下标识符的正确性：

合法标识符有： **a x1 no_1 _a2c sum Name *name***

不合法标识符有：**2a x+y α π a,b a&b const**

标识符

由程序员定义的命名符

语法：以字母或下划线开始，由字母、数字和下划线组成的符号串

例 判断以下标识符的正确性：

合法标识符有： **a x1 no_1 _a2c sum Name name**

不合法标识符有：***2a*** **x+y α π a,b a&b const**

以数字开头

标识符

由程序员定义的命名符

语法：以字母或下划线开始，由字母、数字和下划线组成的符号串

例 判断以下标识符的正确性：

合法标识符有： **a x1 no_1 _a2c sum Name name**

不合法标识符有：**2a *x+y α π a,b a&b* const**

非法符号

标识符

由程序员定义的命名符

语法：以字母或下划线开始，由字母、数字和下划线组成的符号串

例 判断以下标识符的正确性：

合法标识符有： **a x1 no_1 _a2c sum Name name**

不合法标识符有：**2a x+y α π a,b a&b *const***

关键字

变量的定义

- 变量是存储单元
- 变量定义：申请指定类型的存储空间，并以指定标识符命名

变量定义形式： 类型 标识符, 标识符, … , 标识符 ;

变量的定义

- 变量是存储单元
- 变量定义：申请指定类型的存储空间，并以指定标识符命名

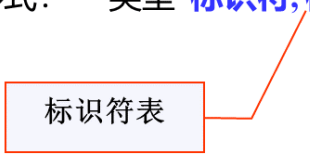
变量定义形式： **类型** 标识符, 标识符, … , 标识符 ;

已定义类型

变量的定义

- 变量是存储单元
- 变量定义：申请指定类型的存储空间，并以指定标识符命名

变量定义形式：　　类型 **标识符, 标识符, … , 标识符**；



变量的定义

- 变量是存储单元
- 变量定义：申请指定类型的存储空间，并以指定标识符命名

变量定义形式：　　类型 标识符, 标识符, … , 标识符；

例如：

```
int   x ;
int   wordCut , Radius , Height ;
double   FlightTime , Mileage , Speed ;
```

赋值语句

在C++语言中，“=”作为赋值运算符，而不表示“等于”判断。赋值语句是由赋值表达式再加上分号构成的表达式语句，它是程序中使用最多的语句之一。

变量=表达式;

在赋值语句的使用中，需要注意以下几点：

1）由于赋值运算符“=”右边的表达式也可以是赋值表达式，因此，下述形式：

变量=（变量=表达式）；

是成立的，从而形成嵌套的情形。其展开之后的一般形式为：

变量=变量=…=表达式;

例如，“a=b=c=d=e=5;”，它实际上等价于：e=5;d=e;c=d;b=c;a=b;

2）在进行赋值运算时，如果赋值运算符两边的数据类型不同，系统将会自动进行类型转换，即将赋值运算符右边的数据类型转换成左边的变量类型。当左边是整型而右边是实型时，将去掉小数部分并截取该整型对应的有效位数。

赋值表达式

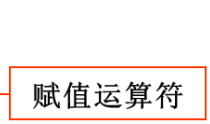
赋值表达式的作用是把数据值写入变量，修改对象的值

一般形式为：　　*变量 = 表达式*

赋值表达式

赋值表达式的作用是把数据值写入变量，修改对象的值

一般形式为：　　*变量 = 表达式*



例如

```
int  Score1 = 90 ;
int  Score2 = 75 ;
int Temp = Score2;
Score2 = Score1;
Score1 = Temp;
```

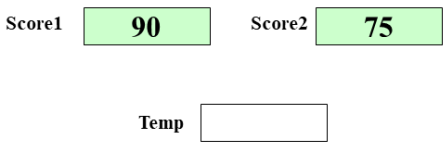
赋值表达式

赋值表达式的作用是把数据值写入变量，修改对象的值

一般形式为：　　*变量 = 表达式*

例如

```
int  Score1 = 90 ;
int  Score2 = 75 ;
int Temp = Score2;
Score2 = Score1;
Score1 = Temp;
```



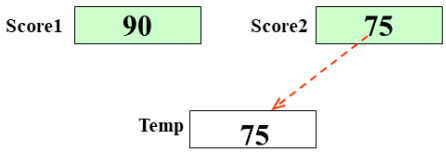
赋值表达式

赋值表达式的作用是把数据值写入变量，修改对象的值

一般形式为：　　*变量 = 表达式*

例如

```
int  Score1 = 90 ;
int  Score2 = 75 ;
int Temp = Score2;
Score2 = Score1;
Score1 = Temp;
```



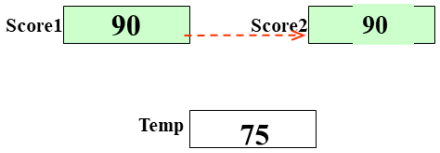
赋值表达式

赋值表达式的作用是把数据值写入变量，修改对象的值

一般形式为：　　*变量 = 表达式*

例如

```
int  Score1 = 90 ;
int  Score2 = 75 ;
int Temp = Score2;
Score2 = Score1;
Score1 = Temp;
```



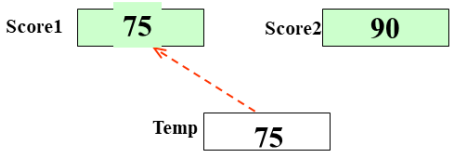
赋值表达式

赋值表达式的作用是把数据值写入变量，修改对象的值

一般形式为：　　*变量 = 表达式*

例如

```
int  Score1 = 90 ;
int  Score2 = 75 ;
int Temp = Score2;
Score2 = Score1;
Score1 = Temp;
```



赋值表达式

赋值运算的强制类型转换

例如

```
int  x = 0 ;
x = 2.3 ;
cout << "x = " << x ;
```

x = 2

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合（由右向左）
- 赋值表达式称为左值表达式（左边是一个变量或对象），允许关联赋值

例如

```
x = y = z + 2 ;
x = y = ( z + 2 ) ;
x = ( y = z + 2 ) ;
( x = y ) = z + 2 ;
z + 2 = x = y ;
```

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

（1）求值

例如

```
x = y = z + 2 ;
x = y = ( z + 2 ) ;
x = ( y = z + 2 ) ;
( x = y ) = z + 2 ;
z + 2 = x = y ;
```

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

例如

```
x = y = z + 2;
x = y = ( z + 2 );
x = ( y = z + 2 );
( x = y ) = z + 2;
z + 2 = x = y;
```

（2）把 z+2 的值写入 y

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

例如

```
x = y = z + 2;
x = y = ( z + 2 );
x = ( y = z + 2 );
( x = y ) = z + 2;
z + 2 = x = y;
```

（3）把 y 的值写入 x

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

例如

```
x = y = z + 2;
x = y = ( z + 2 );
x = ( y = z + 2 );
( x = y ) = z + 2;
z + 2 = x = y;
```

与第一行等价

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

例如

```
x = y = z + 2 ;
x = y = ( z + 2 ) ;
x = ( y = z + 2 ) ;
( x = y ) = z + 2 ;
z + 2 = x = y ;
```

与第一行等价

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

例如

```
x = y = z + 2 ;
x = y = ( z + 2 ) ;
x = ( y = z + 2 ) ;
( x = y ) = z + 2 ;
z + 2 = x = y ;
```

（1）把 y 的值写入 x

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

例如

```
x = y = z + 2 ;
x = y = ( z + 2 ) ;
x = ( y = z + 2 ) ;
( x = y ) = z + 2 ;
z + 2 = x = y ;
```

（2）求值

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

例如

```
x = y = z + 2 ;
x = y = ( z + 2 ) ;
x = ( y = z + 2 ) ;
( x = y ) = z + 2 ;
z + 2 = x = y ;
```

（3）把 z+2 的值写入 x

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

例如

```
x = y = z + 2 ;
x = y = ( z + 2 ) ;
x = ( y = z + 2 ) ;
( x = y ) = z + 2 ;
z + 2 = x = y ;
```

注意
对变量x 作了两次写操作

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

例如

```
x = y = z + 2 ;
x = y = ( z + 2 ) ;
x = ( y = z + 2 ) ;
( x = y ) = z + 2 ;
z + 2 = x = y ;
```

注意
第一次赋值操作没有意义

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

例如

```
x = y = z + 2 ;
x = y = ( z + 2 ) ;
x = ( y = z + 2 ) ;
( x = y ) = z + 2 ;
z + 2 = x = y ;
```

错误
向哪一个对象赋值？

赋值表达式

赋值运算的优先级和关联性

- 赋值运算的优先级很低
- 赋值运算的右结合
- 赋值表达式称为左值表达式，允许关联赋值

例如

```
x = y = z + 2 ;
x = y = ( z + 2 ) ;
x = ( y = z + 2 ) ;
( x = y ) = z + 2 ;
z + 2 = x = y ;
```

它是一个右值表达式
只能放在赋值号右边

赋值表达式

复合赋值运算

双目算符 *op* 的表达式：A = A *op* B

可以缩写成： A *op* = B

例：

c = c - k ⇨ c - = k　　x = x + 3 ⇨ x + = 3

x = x % 3 ⇨ x % = 3　　x = x * (y + 8) ⇨ x * = y + 8

赋值表达式

复合赋值运算

双目算符 *op* 的表达式：A = A *op* B

可以缩写成： A *op* = B

C++ 提供的10个复合赋值运算符：

+=　　-=　　*=　　/=　　%=

<<=　　>>=　　&=　　^=　　|=

用于位运算

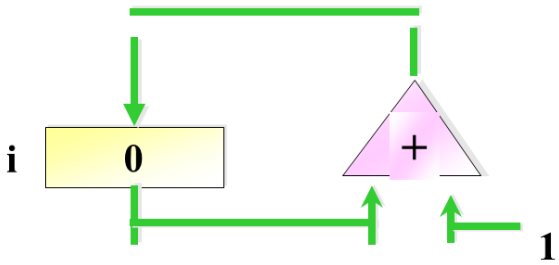
尼克每天背一首古诗后，模仿古人在一根木棒上刻一条痕，若连续刻了5天，试编一程序，算算一共有多少道痕？

自增和自减

程序设计中，常对变量进行如下操作：

i = i + 1　　i = i - 1

此时，变量 i 称为计数器

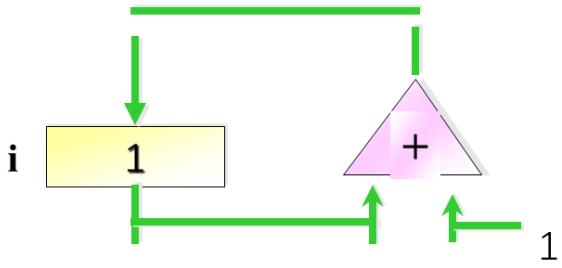


自增和自减

程序设计中，常对变量进行如下操作：

i = i + 1　　i = i - 1

此时，变量 i 称为计数器

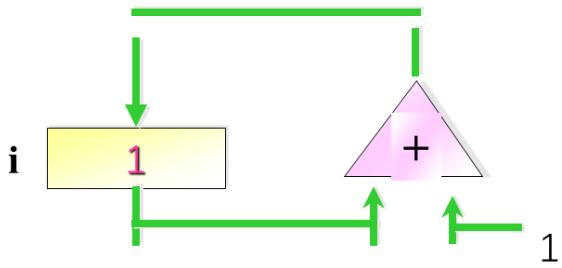


自增和自减

程序设计中，常对变量进行如下操作：

i = i + 1　　i = i - 1

此时，变量 i 称为计数器

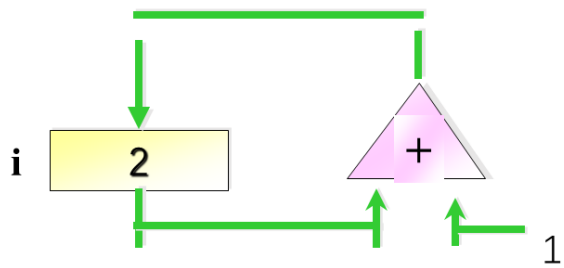


自增和自减

程序设计中，常对变量进行如下操作：

i = i + 1　　i = i - 1

此时，变量 i 称为计数器



自增和自减

程序设计中，常对变量进行如下操作：

i = i + 1　　i = i - 1

此时，变量 i 称为计数器

C++ 为其提供自增和自减算符

	算 符	前 缀 式	后 缀 式	等 价 语 句
自 增	+ +	+ + i	i + +	i = i + 1
自 减	- -	- - i	i - -	i = i - 1

自增和自减

注：

- 自增、自减算符的运算对象只能是整型变量，不能为常量或表达式；

例：　5++　++(a++)　(x + y) --　*错误*

自增和自减

注：

- 自增、自减算符的运算对象只能是整型变量，不能为常量或表达式；

例：　*5*++　++(*a*++)　(*x* + *y*) --　*错误*

自增和自减

注：

- 自增、自减算符的运算对象只能是整型变量，不能为常量或表达式；

  例： 5++ ++(a++) (x + y) -- *错误*

- 自增式和自减式作为独立的表达式，前缀式和后缀式没有区别；

  但作为表达式右值时：

  （1）前缀式 先增值后引用

  例： x = ++ i 相当于 i = i + 1； x = i；

  （2）后缀式 先引用后增值

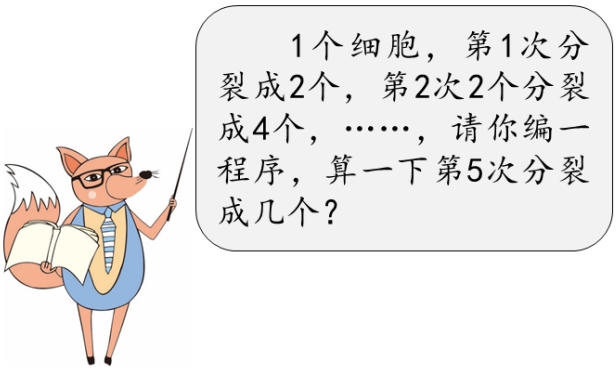
  例： x = i ++ 相当于 x = i； i = i + 1；

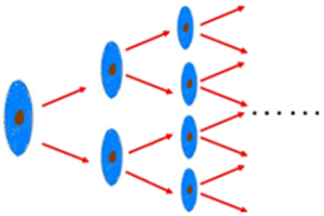
自增和自减

例：

```
int  a = 3 ;
int  b = ++ a ;          // 相当于a = a +1 ; b = a ;
int  c = a ++ ;          // 相当于c = a ; a = a + 1;
b = a -- ;               // 相当于b = a ; a = a - 1 ;
c = -- a ;               // 相当于a = a - 1; c = a ;
++ ( a ++ ) ;            // 错 (a++)不是变量名
c = a + + b;             // 错 a++ 无法对b 操作
c = a ++ + b;            // 相当于c = a + b; a = a + 1;
c = a + + + + b;         // (a++) ++b 错
```

试编一程序，算一算数字1、2、3、4的和是多少？





快乐刷题

- P39 交换墨水

```cpp
#include <iostream>
#include <cstdio>
#include <cmath>
using namespace std;

int main() {

    int a, b;
    cin >> a >> b;

    // cout << b << " " << a << endl;

    /*
        int t = a;
        a = b;
        b = t;
    */

    swap(a, b);

    cout << a << " " << b << endl;

    return 0;
}
```