西北工业大学
NORTHWESTERN POLYTECHNICAL UNIVERSITY

# C++程序设计
# Programming in C++

**1011018**

主讲：魏英，计算机学院

# 函数的设计

1、默认参数

2、函数重载

- C++允许在函数定义或函数声明时，为形参指定默认值，这样的参数称为默认参数（default argument），一般形式为：

```
返回类型 函数名(...,类型 默认参数名=默认值)
{       函数体              }
```

```
int add(int x=5,int y=6)
{
    return x+y;
}
```

```
int main()
{   add(10,20); //10+20
    add(10); //10+6
    add(); //5+6
    return 0;
}
```

- （1）如果在函数定义时设置了默认参数，那么就不能在函数声明中再次设置，反之亦然。

```
int volume(int L, int W , int H=1);
int volume(int L, int W , int H=1)
//定义再次设置，错误
{
    return length * width * height;
}
```

```
int volume(int L, int W , int H=1);
int volume(int L, int W , int H) // 正确
{
    return length * width * height;
}
```

- （2）可以设置多个默认参数，设置的顺序为自右向左，换言之，要为某个参数设置默认值，则它右边的所有参数必须都是默认参数。

```
int volume(int L, int W , int H=1);
//H为默认参数，正确
int volume(int L, int W=1 , int H=1);
//W,H为默认参数，正确
int volume(int L=1, int W=1 , int H=1);
//L,W,H为默认参数，正确
int volume(int L=1, int W , int H); // 错误
int volume(int L, int W=1 , int H); // 错误
```

- （3）默认值可以是常量、全局变量，甚至是一个函数调用（调用实参必须是常量或全局变量的表达式），不可以是局部变量。

```
int p1=2 , p2=10 ;
int max(int a,int b)
{
    return a>b ? a : b;
}
int volume(int L, int W , int H=1); //允许常量
int volume(int L, int W , int H=p1+p2);
//允许全局变量及表达式
int volume(int L, int W , int H=max(5,6));
//允许函数调用
int volume(int L, int W , int H=max(p1,p2))
//允许全局变量函数调用
```

- 默认参数函数的调用

- 默认参数本质上是编译器根据函数声明或函数定义时的默认参数设置，对函数调用中没有给出来的实参自动用默认值表达式“补齐”再进行编译。

【例10.1】带默认参数的函数举例。

```
#include <iostream>
using namespace std;
int p1=2 , p2=10 ;
int max(int a,int b)
{      return a>b ? a : b;  }
int volume(int L=1, int W=p1+p2 , int H=max(p1,p2))
{
    return L * W * H ;
}
int main()
{
    cout << "v0=" << volume() <<endl;
    cout << "v1=" << volume(5) <<endl;
    cout << "v2=" << volume(5,10) <<endl;
    cout<<"v3="<<volume(5,10,15) <<endl;
    return 0;
}
```

- 函数重载（function overloading）是在同一个域中用同一个函数名来定义多个函数，但函数参数列表应彼此有不同，或者是参数个数不同，或者是参数类型不同，或者两者均有不同。

```
int add(int a, int b);
double add(double a, double b);
```

形参类型不同

```
int add(int a, int b);
int add(int a, int b, int c);
```

形参个数不同

【例10.2】函数重载举例。

```
1  #include <iostream>
2  using namespace std;
3  int max(int a, int b) //整型版本
4  {
5      return (a>b ? a : b);
6  }
7  double max(double a, double b) //双精度版本
8  {
9      return (a>b ? a : b);
10 }
11 long max(long a, long b) //长整型版本
12 {
13     return (a>b ? a : b);
14 }
```

```
15  int main()
16  {
17      int i=12 , j=-12 , k;
18      k = max(i,j) ; //调用整型版本max
19      cout << "int max=" << k << endl;
20      double x=123.4 , y=65.43 , z;
21      z = max(x,y) ; //调用双精度版本max
22      cout << "double max=" << z << endl;
23      long a=7654321 , b=1234567, c;
24      c = max(a,b) ; //调用长整型版本max
25      cout << "long max=" << c << endl;
26      return 0;
27  }
```

【例10.3】函数重载举例。

```
 1  #include <iostream>
 2  using namespace std;
 3  int max(int a, int b) //两个参数版本
 4  {    return (a>b ? a : b);  }
 7  int max(int a, int b, int c) //三个参数版本
 8  {    a = a>b ? a : b ;   a = a>c ? a : c ;
11       return (a);
12  }
13  int main()
14  {
15       int a , b, i=10, j=8 , k=12;
16       a = max(i,j) ; //调用两个参数版本max
18       b = max(i,j,k) ; //调用三个参数版本max
20       return 0;
21  }
```

- 函数重载的使用说明：
- （1）重载函数的形参必须不同（个数不同或类型不同）。
- （2）编译程序将根据实参和形参的类型及个数的最佳匹配来选择调用哪一个函数。
- （3）不要将不同功能的函数声明为重载函数，以免出现调用结果的误解、混淆。

# C程序设计