# C++与C的主要差异 – 函数重载

C++允许不同的函数具有相同的函数名，这就是函数重载。

当调用一个函数时，除了要写出函数名，还要根据函数的形参列表传递实参值。

对于函数名相同的多个函数，要在调用时能够区分开到底要调用哪个函数，只能根据传递实参在数量或类型上的不同来进行判断。

也就是说，函数名相同的函数形参列表不能完全一样，否则会因无法区分而报错。

【例1-7】绝对值函数的重载。

提示：三个myabs函数形参的数据类型不同，因此，在调用myabs函数时，

系统会根据传入的实参类型决定调用哪个myabs函数。

```cpp
#include <iostream>
using namespace std;
int myabs(int x);
float myabs(float x);
double myabs(double x);
int main()
{
    int a = -5;
    float b = -3.2f;
    double c = -4.75;
    cout<<myabs(a)<<endl;
    cout<<myabs(b)<<endl;
    cout<<myabs(c)<<endl;
    return 0;
}
```

```cpp
int myabs(int x)
{
    out<<"int abs(int x)被调用！"<<endl;
    return (x<0)?-x:x;
}
float myabs(float x)
{
    cout<<"float abs(float x)被调用！"<<endl;
    return (x<0)?-x:x;
}
double myabs(double x)
{
    cout<<"double abs(double x)被调用！"<<endl;
    return (x<0)?-x:x;
}
```

【例1-8】最大值函数的重载。

提示：两个max函数形参的数据类型虽然相同，但数量不同，因此，在调用max函数时，系统会根据传入的实参数量决定调用哪个max函数。

```cpp
// max.cpp
#include <iostream>
using namespace std;
int max(int x, int y);
int max(int x, int y, int z);
int main()
{
    int a = 5, b = 10, c = 15;
    cout<<max(a, b)<<endl;
    cout<<max(a, b, c)<<endl;
    return 0;
}
```

```cpp
int max(int x, int y)
{
    cout<<"int max(int x, int y)被调用！"<<endl;
    return (x>y)?x:y;
}
int max(int x, int y, int z)
{
    int c;
    cout<<"int max(int x, int y, int z)被调用！"<<endl;
    c = (x>y)?x:y;
    return (c>z)?c:z;
}
```

提示：

功能相近的函数才有必要重载，互不相关的函数进行重载会降低程序的可读性。

重载的函数必须在形参列表上有所区别。如果仅仅是返回类型不同，不能作为重载函数。比如：

```
int myabs(int a);
float myabs(int b);   // 错误：与“int myabs(int a);”相比只有返回类型不同，不构成重载
```

避免默认形参所引起的函数二义性。比如：

```
int max(int a, int b);
int max(int a, int b, int c=0);
```

从形式上来看，两个max函数的形参数量不同，符合函数重载的条件。但实际上，两个max函数都可以通过“max(a, b)”的形式进行调用，此时就产生了二义性。