§4. 函数(补充)

1. 函数的重载

重载：同一作用域中多个函数使用相同的名称

引入：对同一类功能的实现，仅参数的个数或类型不同，希望采用相同的函数名

C不允许
C++允许

```
      int    imax(int x,    int y);
      float  fmax(float x,  float y);
      long   lmax(long x,   long y);
====> 希望 imax/fmax/lmax 都叫 max ?
      int    max(int x,    int y);
      float  max(float x,  float y);
      long   max(long x,   long y);
```

```
      int max2(int x, int y);
      int max3(int x, int y, int z);
      int max4(int x, int y, int z, int w);
====> 希望 max2/max3/max4 都叫 max ?
      int max(int x, int y);
      int max(int x, int y, int z);
      int max(int x, int y, int z, int w);
```

例：分别求两个int和double型数的最大值

```
int max(int x, int y)
{   cout << sizeof(x) << endl;
    return (x > y ? x : y);
}
double max(double x, double y)
{   cout << sizeof(x) << endl;
    return (x > y ? x : y);
}
int main()
{   cout << max(10,   15)   << endl;
    cout << max(10.2, 15.3) << endl;
}
```

?

例：分别求两个/三个int数的最大值

```
int max(int x, int y)
{   cout << 2 << ' ';
    return (x > y ? x : y);
}
int max(int x, int y, int z)
{   cout << 3 << ' ';
    int t = (x > y ? x : y);
    return (t > z ? t : z);
}
int main()
{   cout << max(10, 17)    << endl;
    cout << max(23, 15, 8) << endl;
}
```

?

§4. 函数(补充)

1. 函数的重载

重载：同一作用域中多个函数使用相同的名称

引入：对同一类功能的实现，仅参数的个数或类型不同，希望采用相同的函数名

重载函数调用时的匹配查找顺序：

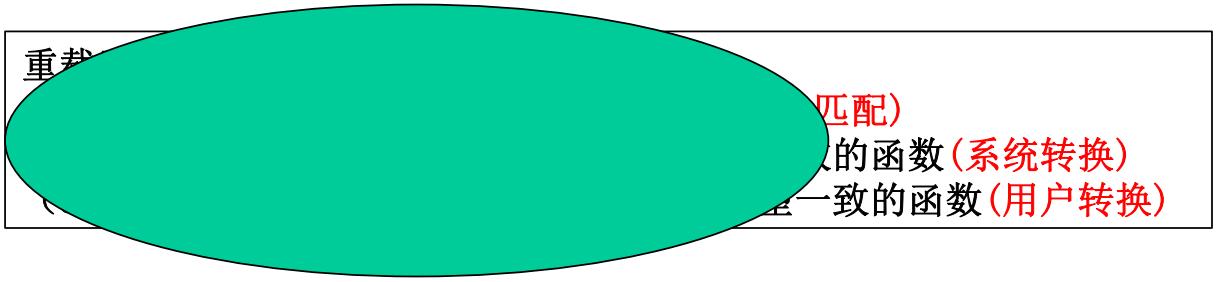
（1）寻找参数个数、类型完全一致的定义(严格匹配)

（2）通过系统定义的转换寻找匹配函数

（3）通过用户定义的转换寻找匹配函数

★ 若某一步匹配成功，则不再进行下一顺序的匹配

★ 若某一步中发现两个以上的匹配则出错



例：分别求两个int和double型数的最大值

```
#include <iostream>
using namespace std;
int max(int x, int y)
{   cout << sizeof(x) << ' ';
    return (x > y ? x : y);
}
double max(double x, double y)
{   cout << sizeof(x) << ' ';
    return (x > y ? x : y);
}
int main()
{   cout << max(10,   15)     << endl;    //int, int
    cout << max(10.2, 15.3)   << endl;    //double, double
    cout << max(10,   int(15.3)) << endl; //int,double
    cout << max(5+4i, 15.3)   << endl;    //复数，double
    return 0;
}
```

1

2

哪句语句编译会错?
其它正确语句的输出是什么？

复数形式目前编译会错，如何定义复数以及定义复数向double的转换，具体见后续课程相关内容

严格匹配1
严格匹配2
系统转换1
需自定义转换

自行将max的参数换成U/L/F等不同组合，看是否报错，按什么类型做系统转换

§4. 函数(补充)

1. 函数的重载

使用：

★ 要求同名函数的参数个数、参数类型不能完全相同

`void fun(int x, int y);` `void fun(int x, int y, int z);`	正确 参数个数不同,类型同
`void fun(int x, int y);` `void fun(long x, long y);`	正确 参数个数同,类型不同
`void fun(int x, int y);` `void fun(long x, long y, long z);`	正确 个数类型均不同
`void fun(int x, int y);` `void fun(int x, int y);`	错误 个数类型均相同

★ 返回类型及参数名不做检查(仅这两个不同认为错)

`int  max(int x, int y);` `long max(int x, int y);`	错误,仅返回类型不同 参数类型个数完全相同
`int  max(int x, int y);` `int  max(int p, int q);`	错误,仅参数名不同 参数类型个数完全相同

★ 尽量使同名函数完成相同或相似的功能，否则可能导致概念混淆(建议)

§4. 函数(补充)

2. 有默认参数的函数

含义：对函数的某一形参，可以指定默认值，从而简化函数的调用(默认值建议为常量)

形式：

返回类型 函数名(无默认参数形参,有默认参数形参)

```
{
    函数体
}
```

```
void circle(int x, int y, int r=10)
{
    ...
}
```

调用：circle(0,0); ⇔ circle(0,0,10);

```
circle(5,8,12);
```

§4. 函数(补充)

2. 有默认参数的函数

使用：

★ 便于函数功能的扩充，减少代码维护，修改的数量

例：画圆函数circle，之前实现时考虑了圆心、半径、颜色，仅空心圆
现在实际应用需求中（甲方）要求考虑是否填充(空心/实心)

原函数：`void circle(int x,int y,int r=10,int color=7)`

原使用方法：

```
circle(15, 20);          //圆心(15,20)，半径10(缺省)，颜色白(缺省)，空心圆
circle(15, 20, 8);       //圆心(15,20)，半径12(指定)，颜色白(缺省)，空心圆
circle(25, 18, 12, 1); //圆心(25,18)，半径12(指定)，颜色蓝(指定)，空心圆
```

扩充为新函数：`void circle(int x,int y,int r=10,int color=7,bool filled=false)`

则：在原函数基础上补充填充部分的代码即可，原有的涉及圆心、半径、颜色等处理的代码可以保持不变，调用方法维持不变

新使用方法：

```
circle(15, 20);          //圆心(15,20)，半径10(缺省)，颜色白(缺省)，空心圆(缺省)
circle(15, 20, 8);       //圆心(15,20)，半径12(指定)，颜色白(缺省)，空心圆(缺省)
circle(25, 18, 12, 1);   //圆心(25,18)，半径12(指定)，颜色蓝(指定)，空心圆(缺省)
circle(25, 18, 15, 6,  true); //圆心(25,18)，半径15(指定)，颜色黄(指定)，实心圆(指定)
```

原使用方法不变

结论：

1、有默认参数的函数，能有效地减少了修改次数，减少了工作量

2、最好的方法，是在初始设计函数时，就考虑到更多可能的因素(包括客户暂时未想到的问题)

§4. 函数(补充)

2. 有默认参数的函数

使用：

★ 便于函数功能的扩充，减少代码维护，修改的数量

★ 允许有多个默认参数，但必须是连续的最后几个

```
void circle(int y, int x=0, int r=5)(对)
void circle(int x=0, int y, int r=5)(错)
```

★ 若有多个默认参数，调用时，前面使用缺省值，后面不使用缺省值，则前面也要加上

```
void circle(int x,int y,int r=5,int c=WHITE)
circle(10, 15);
circle(10, 15, 10);
circle(10, 15, 12, BLUE);
circle(10, 15, 5, BLUE);
```

虽然是缺省，也要加

§4. 函数(补充)

2. 有默认参数的函数

使用：

★ 若函数定义在调用函数之后，则声明时必须给出默认值，定义时不再给出

```
void circle(int x, int y, int r=10);
int main()
{   ...
}
void circle(int x, int y, int r)
{   ...
}
```

正确

```
void circle(int x, int y, int r=10);
int main()
{   ...
}
void circle(int x, int y, int r=10)
{   ...
}
```

错误，即使相同

```
void circle(int x, int y, int r);
int main()
{   ...
}
void circle(int x, int y, int r=10)
{   ...
}
```

错误

```
void circle(int x, int y, int r=10);
int main()
{   ...
}
void circle(int x, int y, int r=5)
{   ...
}
```

错误

★ 重载与带默认参数的函数一起使用时，可能会产生二义性

```
int fun(int a, int b=10);
int fun(int a);
```

若调用为：fun(10，20)　正确

fun(50)　二义性