3-6:

重载的函数的函数名是相同的，但它们的参数的个数和数据类型不同，编译器根据实参和形参的类型及个数的最佳匹配，自动确定调用哪一个函数。

3-9:

#include <iostream>

#include <cmath>

**using** **namespace** std;

**int** prime(**int** i);

**int** main()

{

**int** i;

    cout<<"请输入一个整数：";

    cin>>i;

**if** (prime (i))

        cout<<i<<"是质数"<<endl;

**else**

        cout<<i<<"不是质数"<<endl;

**return** 0;

}

**int** prime(**int** i)

{

**int** j,k,flag;

    flag=1;

    k=sqrt(i);

**for** (j=2;j<k;j++)

    {

**if**(i%j==0)

        {

            flag=0;

**break**;

        }

    }

**return** flag;

}

运行结果：

**请输入一个整数：Program ended with exit code: 0**7

**7是质数**

3-10:

#include <iostream>

#include <cmath>

**using** **namespace** std;

**int** fnl(**int** i,**int** j);

**int** main()

{

**int** i,j,x,y;

    cout<<"请输入一个正整数";

    cin>>i;

    cout<<"请输入另一个正整数";

    cin>>j;

    x=fnl(i,j);

    y=i\*j/x;

    cout<<i<<"和"<<j<<"的最大公约数是："<<x<<endl;

    cout<<i<<"和"<<j<<"的最小公倍数是："<<y<<endl;

**return** 0;

}

**int** fnl(**int** i,**int** j)

{

**int** temp;

**if**(i<j)

    {

        temp=i;

        i=j;

        j=temp;

    }

**while**(j!=0)

    {

        temp=i%j;

        i=j;

        j=temp;

    }

**return** i;

}

运行结果：

**请输入一个正整数Program ended with exit code: 0**9

**请输入另一个正整数**24

**9和24的最大公约数是：3**

**9和24的最小公倍数是：72**

3-13:

#include <iostream>

**using** **namespace** std;

**int** fib(**int** n);

**int** main()

{

**int** n, answer;

    cout<<"Enter number:";

    cin>>n;

    cout<<"\n\n";

    answer=fib(n);

    cout<<answer<<"is the"<<n<<"th Fibonacci number\n";

**return** 0;

}

**int** fib (**int** n)

{

    cout<<"processing fib("<<n<<")...";

**if** (n<3)

    {

        cout<<"Return 1!\n";

**return**(1);

    }

**else**

    {

        cout<<"call fib("<<n-2<<") and fib("<<n-1<<").\n";

**return** (fib (n-2)+fib(n-1));

    }

}

运行结果：

**Enter number:Program ended with exit code: 0**10

**processing fib(10)...call fib(8) and fib(9).**

**processing fib(8)...call fib(6) and fib(7).**

**processing fib(6)...call fib(4) and fib(5).**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(5)...call fib(3) and fib(4).**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(7)...call fib(5) and fib(6).**

**processing fib(5)...call fib(3) and fib(4).**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(6)...call fib(4) and fib(5).**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(5)...call fib(3) and fib(4).**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(9)...call fib(7) and fib(8).**

**processing fib(7)...call fib(5) and fib(6).**

**processing fib(5)...call fib(3) and fib(4).**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(6)...call fib(4) and fib(5).**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(5)...call fib(3) and fib(4).**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(8)...call fib(6) and fib(7).**

**processing fib(6)...call fib(4) and fib(5).**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(5)...call fib(3) and fib(4).**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(7)...call fib(5) and fib(6).**

**processing fib(5)...call fib(3) and fib(4).**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(6)...call fib(4) and fib(5).**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(5)...call fib(3) and fib(4).**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**processing fib(4)...call fib(2) and fib(3).**

**processing fib(2)...Return 1!**

**processing fib(3)...call fib(1) and fib(2).**

**processing fib(1)...Return 1!**

**processing fib(2)...Return 1!**

**55is the10th Fibonacci number**

3-15:

#include <iostream>

**using** **namespace** std;

**int** getPower(**int** x,**int** y);

**double** getPower(**double** x, **int** y);

**int** main()

{

**int** iNumber,power;

**double** dNumber;

**int** iAnswer;

**double** dAnswer;

    cout<<"Enter an int base number:";

    cin>>iNumber;

    cout<<"Enter an double base number:";

    cin>>dNumber;

    cout<<"To what power?";

    cin>>power;

    iAnswer= getPower(iNumber, power);

    dAnswer= getPower(dNumber, power);

    cout<<iNumber<<"to the"<<power<<"th power is"<<iAnswer<<endl;

    cout<<dNumber<<"to the"<<power<<"th power is"<<dAnswer<<endl;

**return** 0;

}

**int** getPower(**int** x,**int** y)

{

**if**(y==1)

**return** x;

**else** **if**(y==0)

**return** 1;

**else** **if**(y<0)

**return** 0;

**else**

**return** (x\*getPower(x,y-1));

}

**double** getPower(**double** x,**int** y)

{

**if**(y==1)

**return** x;

**else** **if**(y==0)

**return** 1;

**else** **if**(y<0)

**return** 1/getPower(x,-y);

**else**

**return** (x\*getPower(x,y-1));

}

运行结果：

**Enter an int base number:Program ended with exit code: 0**9

**Enter an double base number:**6.7

**To what power?**4

**9to the4th power is6561**

**6.7to the4th power is2015.11**

3-16:

对同一个函数的多次不同调用中，编译器会为函数的形参和局部变量分配不同的空间，它们互不影响。

4-3:  
构造函数的作用就是在对象被创建时利用特定的值构造对象，将对象初始化为一个特定的状态，使此对象具有区别于彼对象的特征，完成的就是一个从一般到具体的过程，构造函数在对象创建时由系统自动调用。

析构函数与构造函数的作用几乎正好相反，它是用来完成对象被删除前的一些清理工作，也就是专门做扫尾工作的。一般情况下，析构函数是在对象的生存期即将结束时由系统自动调用的，它的调用完成后，对象消失，相应的内存空间也被释放。

4-5:

可以，类的每一个对象都有自己的数据成员。

4-10:

#include <iostream>

#include <string>

**using** **namespace** std;

**class** Date

{

**private**:

**int** year;

**int** month;

**int** day;

**public**:

    Date(){}

    Date(**int** y,**int** m,**int** d)

    {

        year=y;

        month=m;

        day=d;

    }

**void** set()

    {

        cin>>year>>month>>day;

    }

**void** display()

    {

        cout<<year<<"年"<<month<<"月"<<day<<"日";

    }

};

**class** Person

{

**private**:

**int** num;

**char** sex;

    Date birthday;

**char** ID[18];

**public**:

    Person(){}

    Person(**int** n,**int** y,**int** m,**int** d,**char** id[18],**char** s='m'):birthday(y,m,d)

    {

        num=n;

        sex=s;

        strcpy(ID,id);

    }

    Person(Person& p)

    {

        num=p.num;

        sex=p.sex;

        birthday=p.birthday;

        strcpy(ID,p.ID);

    }

**void** input()

    {

        cout<<"录入数据："<<endl;

        cout<<"编号：";

        cin>>num;

        cout<<"性别(m/f)：";

        cin>>sex;

        cout<<"生日：";

        birthday.set();

        cout<<"身份证号：";

        cin>>ID;

        ID[18]='\0';

        cout<<endl;

    }

**void** output()

    {

        cout<<"编号："<<num<<endl;

        cout<<"性别："<<sex<<endl;

        cout<<"生日：";

        birthday.display();

        cout<<endl;

        cout<<"身份证号："<<ID<<endl;

    }

    ~Person() *//析构函数*

    {

        cout<<" "<<num<<"号人员已经录入"<<endl;

    }

};

**int** main()

{

    Person p1;

    p1.input();

    p1.output();

**return** 0;

}

4-12:

#include <iostream>

**using** **namespace** std;

**class** DataType{

**enum**{

        character,

        integer,

        floating\_point

    }vartype;

**union**

    {

**char** c;

**int** i;

**float** f;

    };

**public**:

    DataType(**char** ch){

        vartype=character;

        c=ch;

    }

    DataType(**int** ii){

        vartype=integer;

        i=ii;

    }

    DataType(**float** ff){

        vartype=floating\_point;

        f=ff;

    }

**void** print();

};

**void** DataType::print(){

**switch** (vartype){

**case** character:

            cout<<"字符型"<<c<<endl;

**break**;

**case** integer:

            cout<<"整型"<<i<<endl;

**break**;

**case** floating\_point:

            cout<<"浮点型"<<f<<endl;

**break**;

    }

}

**int** main(){

    DataType a('g'),b(24),c(2.53F);

    a.print();

    b.print();

    c.print();

**return** 0;

}

运行结果：

**字符型g**

**整型24**

**浮点型2.53**

**Program ended with exit code: 0**

4-18:

Teacher

Department

\*

1..n