C/C++程序设计案例实战
——邀你来做客

华中农业大学信息学院　翟瑞芳

邀你来做客

邀你来做客：友元函数

```
#include<iostream>
using namespace std;
class Room
{public:
    Room(string myname)
    {
      name = myname;
    }
    string GetName() const
    {return name;}
private:
    string name;
};
int main()
{
 Room myroom(“huiyuan424”);
 cout<<myroom.GetName()<<endl;
 return 0;
}
```

邀你来做客：友元函数

```
#include<iostream>
using namespace std;
class Room
{public:
    Room(string myname){ name = myname;   }
    friend void visit(const Room& room);
private:
    string name;
};


int main()
{
 Room myroom(“huiyuan110”);
 cout<<myroom.GetName()<<endl;
 return 0;
}
```

邀你来做客：友元函数

```
#include<iostream>
using namespace std;
class Room
{public:
    Room(string myname){ name = myname;   }
    friend void visit(const Room& room);
private:
    string name;
};
void visit(const Room& room)
{
     cout<<room.name<<endl;
}
int main()
{
 Room myroom("huiyuan110");
 visit(myroom);
 return 0;
}
```

邀你来做客：友元函数

friend 返回类型 函数名称（形式参数列表）

友元函数的声明和定义

1）类体内声明和定义；
2）类体内声明，类体外定义。

```
friend void visit(const Room &);
```

```
friend void Person::visit(const
Room &);
```

邀你来做客：友元类

```
#include<iostream>
using namespace std;
class Person;
class Room
{public:
 Room(string myname)
     {
        name = myname;
     }

private:
     string name;
};
```

邀你来做客：友元类

```
#include<iostream>
using namespace std;
class Person;
class Room
{public:
 Room(string myname)
     {
        name = myname;
     }
 friend class Person;
private:
     string name;
};
```

```
class Person
{
public:
    Person(string name)
    {person_name = name;}




private:
    string person_name;
};
```

邀你来做客：友元类

```
#include<iostream>
using namespace std;
class Person;
class Room
{public:
 Room(string myname)
      {
         name = myname;
      }
 friend class Person;
private:
      string name;
};
```

```
class Person
{
public:
       Person(string name)
       {person_name = name;}
       void visit (const Room&
room){
         cout<<"My name is ” <<
person_name <<endl;
         cout<<“I‘m going to
visit Room 〃 << room.name <<
endl; }
private:
       string person_name;
};
```

邀你来做客：友元类

```
#include<iostream>
using namespace std;
class Person;
class Room
{public:
 Room(string myname)
      {
         name = myname;
      }
 friend class Person;
private:
      string name;
};
```

```
class Person
{
public:
    Person(string name)
    {person_name = name;}
    void visit (const Room&
room){
     cout<<"My name is ” <<
person_name <<endl;
     cout<<“I‘m going to
visit Room 〃 << room.name <<
endl; }
private:
    string person_name;
};
```

```
int main()
{    Room my_room("H1_424");
     Person myfriend("Zhang San");
     myfriend.visit(my_room);
     return 0; }
```

邀你来做客：友元类

友元关系是单向的，不具备交换性。

友元关系不具备传递性。

友元

优势：有助于数据共享，能提高程序的效率。

缺点：破坏了类的封装性。

小结

友元函数

友元类

延申

在邀你来做客案例的基础上，设计University类 。

1）在Person类中，添加成员函数visit_univ，使之成为University类的友元函数；请编程实现。

2)将Room类设计为University类的友元类，增加display函数，显示University对象的相关信息；请编程实现。