

Севастопольский государственный университет
Кафедра информационных систем

Курс лекций по дисциплине
**«ОБЪЕКТНО-ОРИЕНТИРОВАННОЕ
ПРОГРАММИРОВАНИЕ»**
(ООП)

Лектора: Пелипас Всеволод Олегович
Сметанина Татьяна Ивановна

Лекция 6

**ДРУЖЕСТВЕННЫЕ ФУНКЦИИ.
ДРУЖЕСТВЕННЫЕ КЛАССЫ.**

Дружественность

C++ предоставляет возможность обойти (или нарушить) один из основополагающих механизмов ООП – механизм инкапсуляции – с помощью друзей.

Существует два вида друзей класса:

- **дружественная функция;**
- **дружественный класс.**

Дружественные функции **не являются** элементами класса и применяются для доступа к скрытым полям одного класса (или нескольким классам).

Чтобы объявить дружественную функцию некоторому классу, в определении этого класса включают ее прототип, перед которым ставятся служебное слово **friend**.

Пример использования дружественной функции

```
class MyClass{
    int i;
    public: MyClass (int _i) {i=_i;} // конструктор
    friend bool zero(MyClass ob); // объявление друж. функции
};
// описание дружественной функции
bool zero(MyClass ob){
    if (!ob.i) return true;
    else return false;
}

main(){
    MyClass ob(10); // объявление объекта
    if (zero (ob)) cout<< "+"; // вызов друж. функции
    else cout<< "-";
}
```

Пример использования дружественной функции

Дружественная функция не наследуется и может быть дружественной к нескольким классам.

```
class B; // неполное описание класса B
class A { int a; // скрытое поле
        public: A(int _a){a=_a;} // конструктор
        friend int f(A ob_a, B ob_b); // объявление друж. функции
};
class B{ int b; // скрытое поле
        public: B(int _b){b=_b;} // конструктор
        friend int f(A ob_a, B ob_b); // объявление друж. функции
};
int f(A ob_a, B ob_b) // описание друж. функции
{ return (ob_a.a + ob_b.b); }
main() { A ob_a(2); B ob_b(3); // создание объектов
        cout<<f(ob_a, ob_b); // вызов друж. функции
}
```

Неполное объявление класса дает возможность использовать его имя в объявлении функции еще до его определения.

Правила описания и особенности дружественных функций

- Дружественная функция объявляется внутри класса, к элементам которого ей нужен доступ, с ключевым словом **friend**. В качестве параметра ей должен передаваться объект или ссылка на объект класса, поскольку указатель `this` ей не передается.
- Дружественная функция может быть обычной функцией или методом другого ранее определенного класса. На нее не распространяется действие спецификаторов доступа, место размещения ее объявления в классе безразлично.
- Одна функция может быть дружественной сразу несколькими классами.

Использования дружественных функций нужно по возможности избегать, поскольку они нарушают принцип инкапсуляции и, таким образом, затрудняют отладку и модификацию программы.

Дружественный класс

Если все методы какого-либо класса должны иметь доступ к скрытым полям другого, весь класс объявляется дружественным с помощью ключевого слова **friend**.

В приведенном ниже примере класс **B** объявляется дружественным классу **A**:

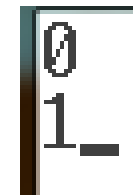
```
class A{  
    friend class B; // класс B объявлен другом класса A  
    int x; // скрытые поля т.к. по умолчанию private  
    void inc_x(){x++;}  
public: // открытые поля  
    A(){x=0;} // конструктор по умолчанию  
    A(int _x){x=_x;} // конструктор с параметрами  
};
```

Дружественный класс

```
class B{  
    A obA;  
    public: void show(); //объявление метода show  
};
```

```
void B::show() { //описание метода show  
    cout<<obA.x<<endl;  
    obA.inc_x();  
    cout<<obA.x;  
}
```

```
main()  
{ B obB; //создание объекта класса B  
  obB.show(); // печать скрытых полей  
}
```



Пример описания дружественных классов

Два класса могут объявить друг друга друзьями.

class B; // неполное объявление класса

```
class A{  
    friend class B;  
    //...  
};
```

```
class B{  
    friend class A;  
    //...  
};
```