

5. Классы, использующие свободную память

5.1. Копирование объектов

```
class S {  
private:  
    int n;  
    Type arr[N];  
public:  
    S():n(0) {}  
    S(int, ...);  
    . . .  
};
```

5.1. Копирование объектов (продолжение)

```
S ob1(5);
```

```
...
```

```
{
```

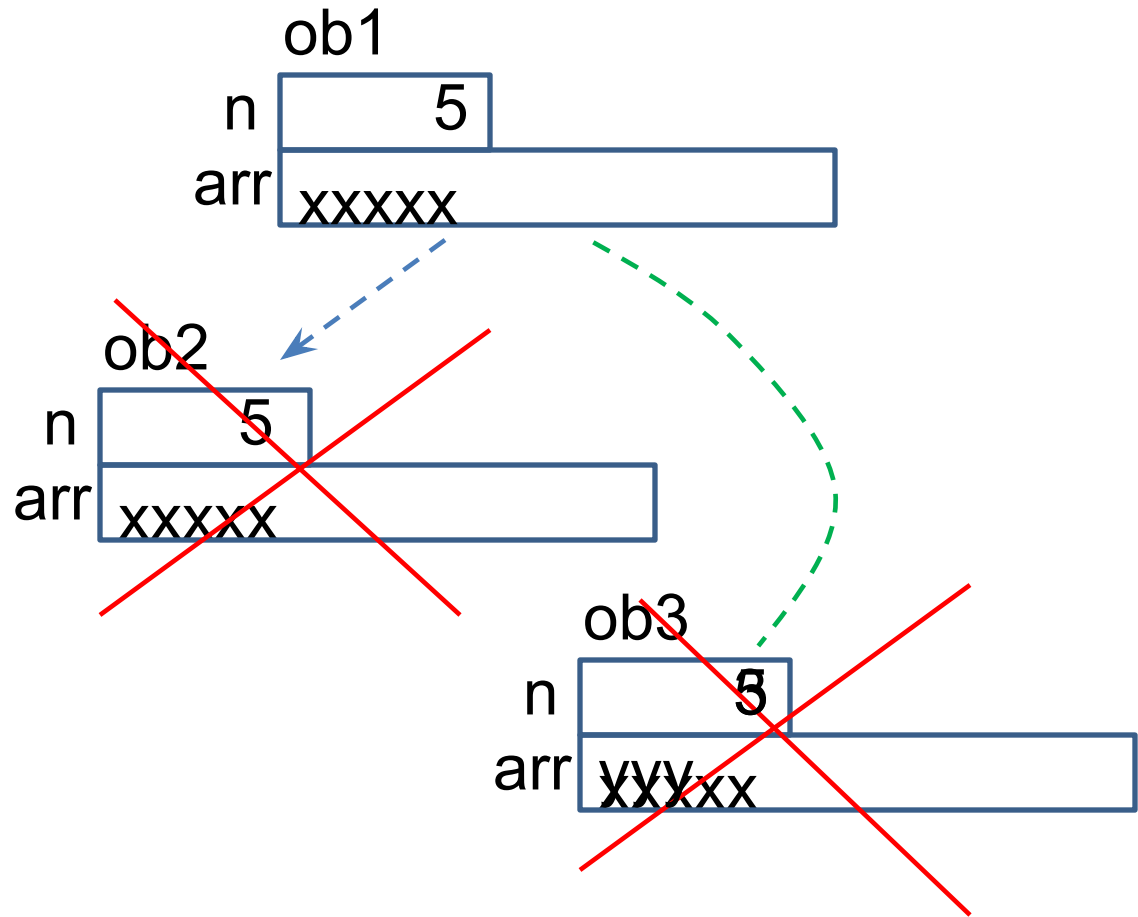
```
  S ob2 = ob1;
```

```
  S ob3(3);
```

```
  ob3 = ob1;
```

```
}
```

```
...
```



5.2. Динамические объекты

```
class S {  
    private:  
        int n;  
        Type *ptr;  
    public:  
        S():n(0), ptr(NULL) { }  
        S(int, ...);  
        ~S(){delete [ ] ptr;}  
};
```

5.2. Динамические объекты (продолжение)

```
S::S(int k, ...):n(k), ptr(new Type[k])  
{  
    // необходимая инициализация  
    // выделенной области памяти  
    . . .  
}
```

5.3. Проблемы

```
S ob1(5, ...);
```

```
...
```

```
{
```

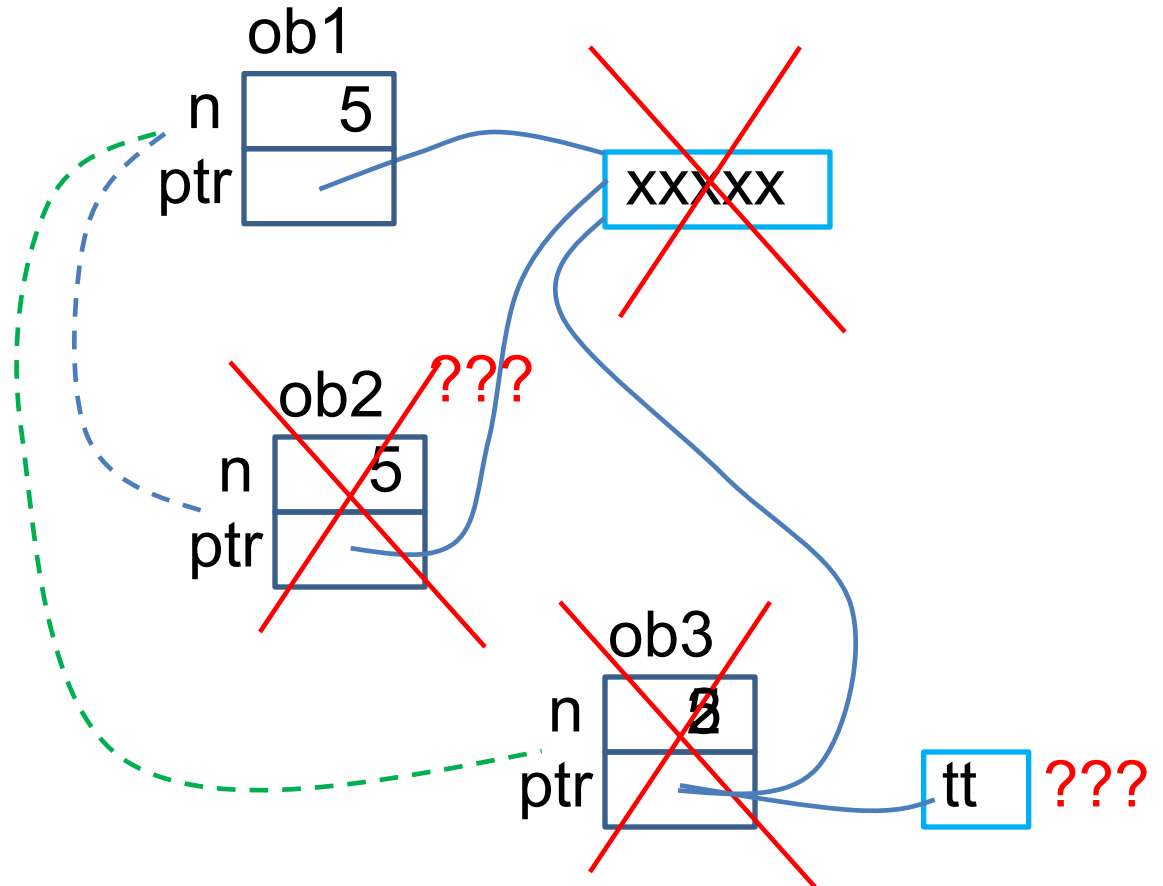
```
  S ob2 = ob1;
```

```
  S ob3(2, ...);
```

```
  ob3 = ob1;
```

```
}
```

ob1 ???



5.4. Копирующий конструктор

- выделение памяти
- копирование данных

```
S::S(const S &ob):n(ob.n), ptr(NULL)
{
    if(n){
        ptr = new Type[n];
        // копирование данных
    }
}
```

5.5. Оператор присваивания

- освобождение памяти
- выделение новой памяти
- копирование данных
- проверка ситуации $a = a$

5.5. Оператор присваивания (продолжение)

```
S & S::operator = (const S& ob)
{
    if(this != &ob) { // проверка a = a
        delete [ ] ptr;
        ptr = NULL;
        if((n = ob.n) != 0) {
            ptr = new Type[n];
            // копирование данных
        }
    }
    return *this;
}
```

5.6. Использование

S a1(3, ...), a2;

a2 = a3; // присваивание

S a3(a1); // копирующий конструктор

S a4 = a1; // копирующий конструктор

5.6. Использование (продолжение)

```
S func (S p) {           // S func(const S &p)
    S a(p);
    . . .
    return a;
}
. . .
a2 = func(a3); // передача аргумента
               // возврат результата
```