MICROOH 麦可网

Android-从程序员到架构师之路

出品人：Sundy
讲师：高焕堂（台湾）

http://www.microoh.com

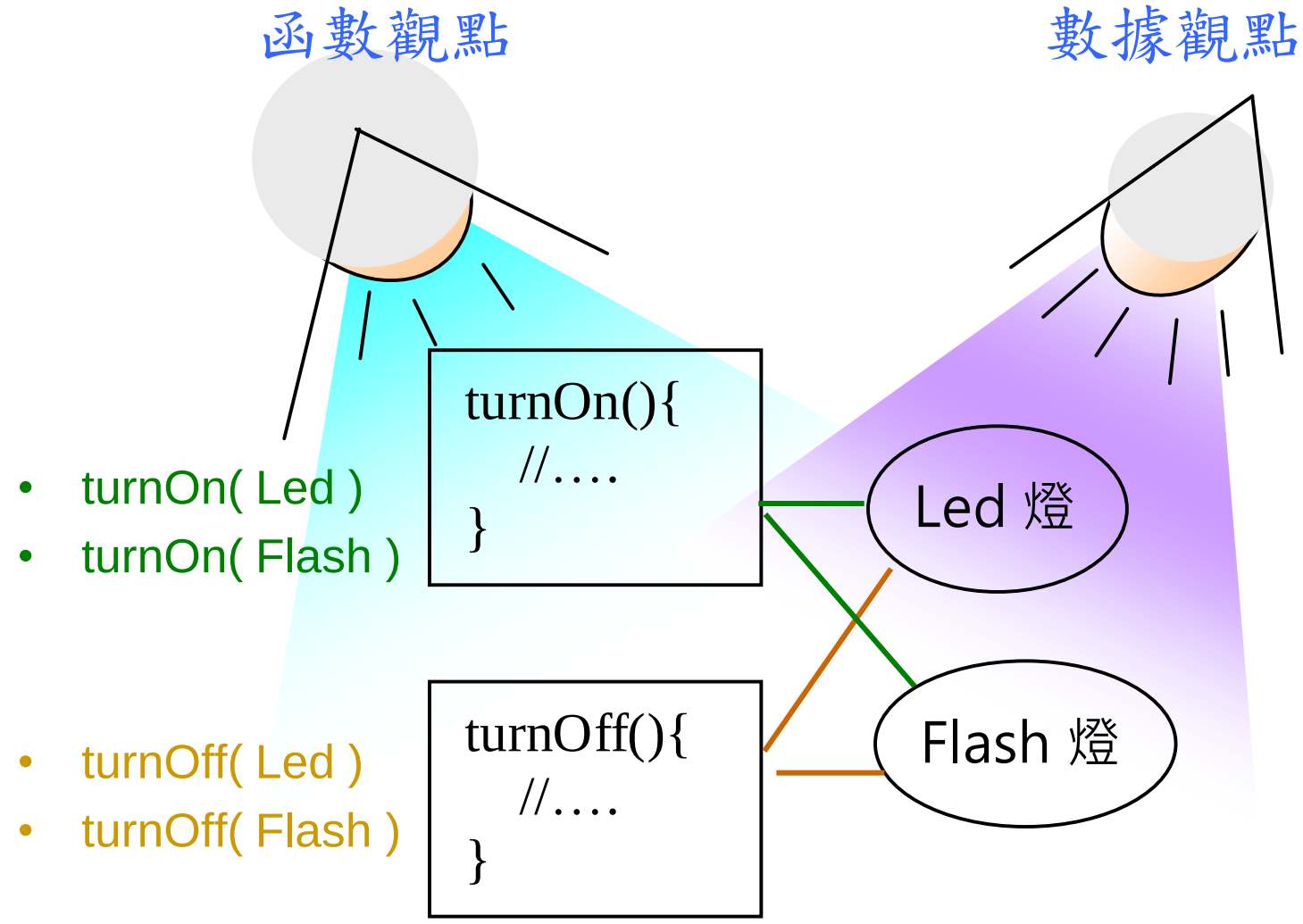
C01_c

JNI架构原理：Java与C的对接(c)

By 高焕堂

复习：C与Java两个观点

幕后一致的本质


函數觀點
數據觀點
turnOn(){
//….
}
turnOff(){
//….
}
Led 燈
Flash 燈
• turnOn(Led)
• turnOn(Flash)
• turnOff(Led)
• turnOff(Flash)

定義結構(class)

```
// Java代碼
```

C函數

```
// a.so 檔案(File)
void turnOn( Led *px )
   {   px->state = 1;   }
void turnOn(Flash *px )
   {   px->state = 1;   }
void turnOff(Led* px)
   {   px->state = 0;   }
void turnOff(Flash* px)
   {   px->state = 0;   }
```

誕生對象&調用函數

```
// Java代碼
```

JNI

C函數

```
// a.so 檔案(File)
void turnOn( Led *px )
   {   px->state = 1;   }
void turnOn( Flash *px )
   {   px->state = 1;   }
void turnOff(Led* px)
   {   px->state = 0;   }
void turnOff(Flash* px)
   {   px->state = 0;   }
```

定義結構(class)

```
// Java代碼
```

誕生對象&調用函數

```
// Java代碼
```

Java对象与C函数有何关系呢?

2.4 以C结构表达类(class)，并创建对象(object)

- 目的：要了解Java对象如何与C函数对接?
- 途径：先了解C对象如何与C函数对接呢?

前言

认识C函数指针

- struct里不能定义函数本身，但能定义函数指针(function pointer)属性。

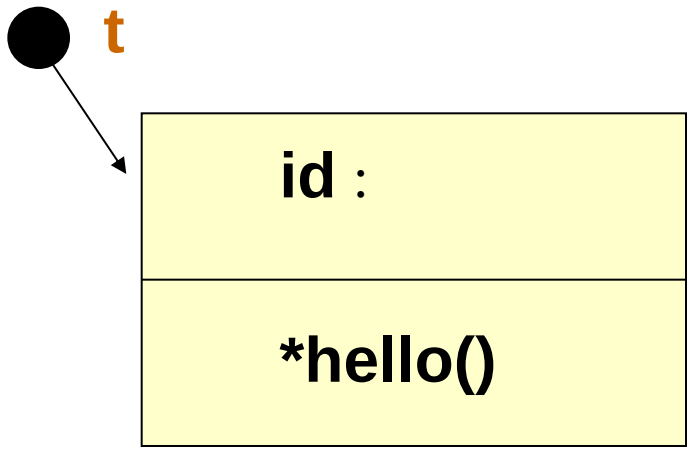
```
typedef struct cc {
    int id;
    void (*hello)();
} CC;
```

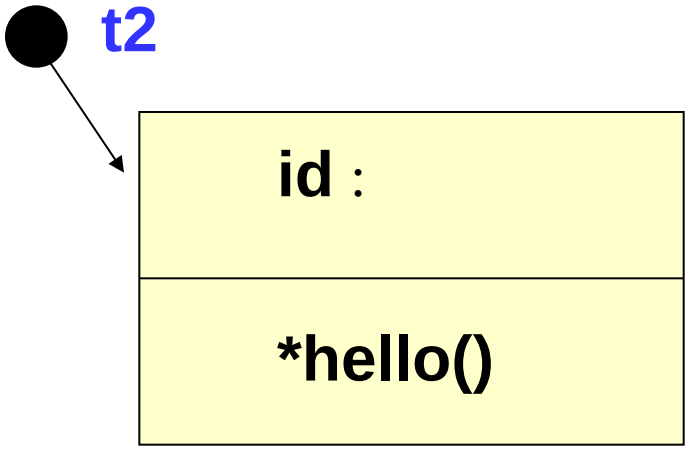
这个hello就是一个函数指针属性了。

```
static void my_hello() {
        printf("Hello");
 }
```

```
typedef struct cc {
            int id;
            void (*hello)();
          } CC;
```

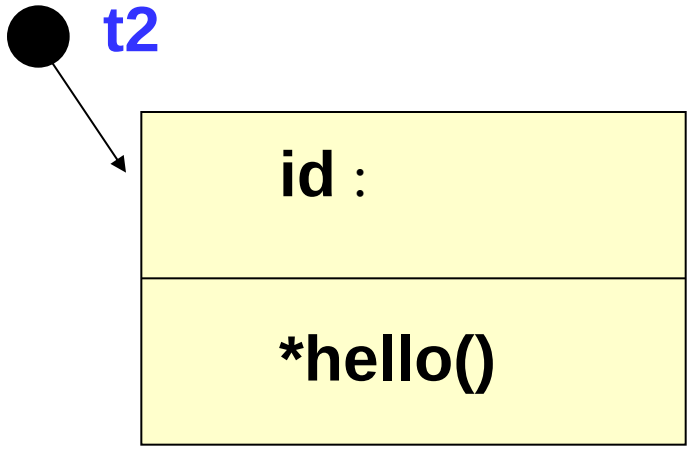
```
static void my_hello() {
        printf("Hello");
 }
```

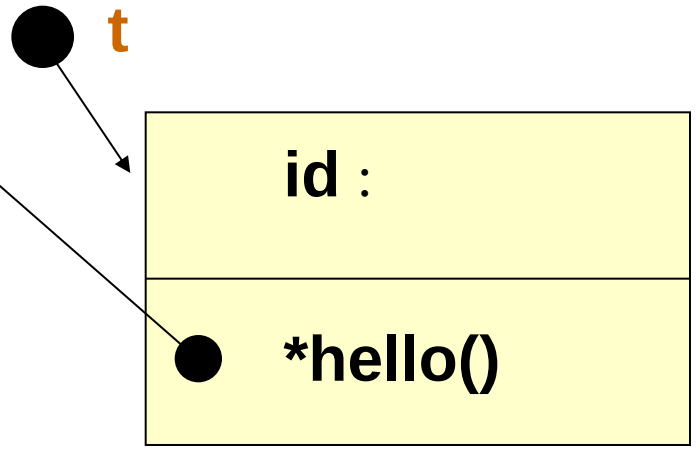




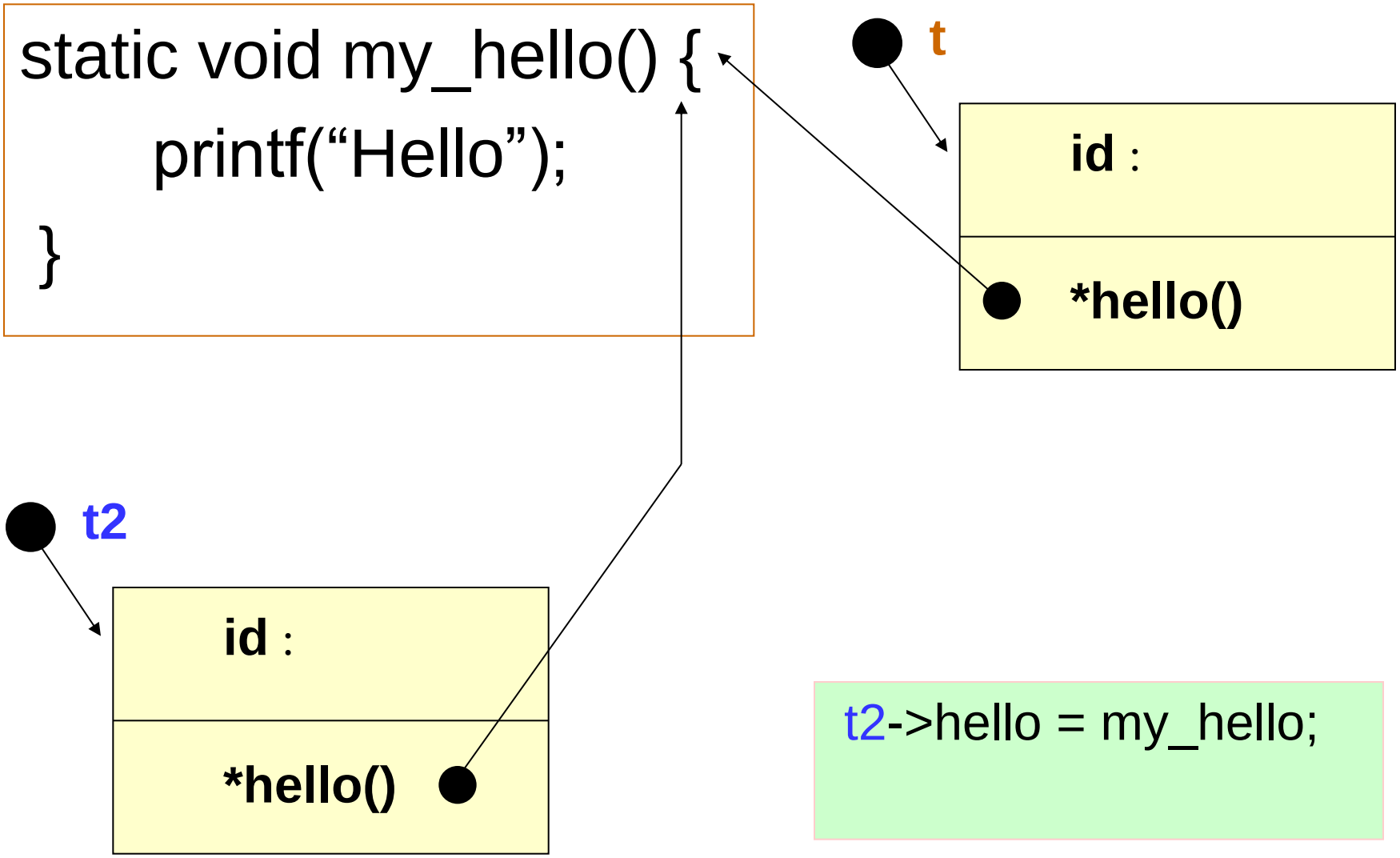

```
CC *t, *t2;
t = (CC *)malloc(sizeof(CC));
t2 = (CC *)malloc(sizeof(CC));
```

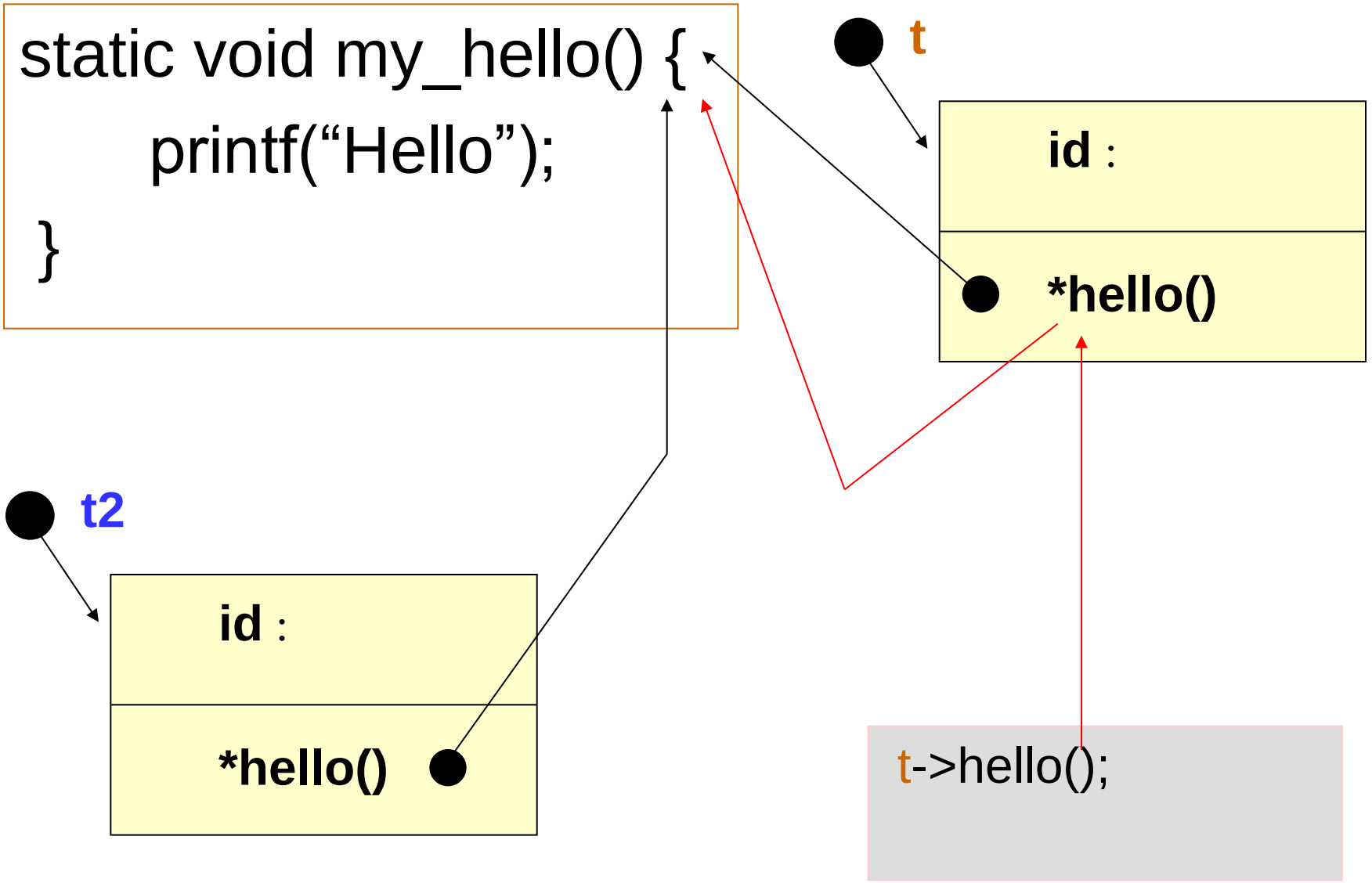
```
static void my_hello() {
        printf("Hello");
 }
```

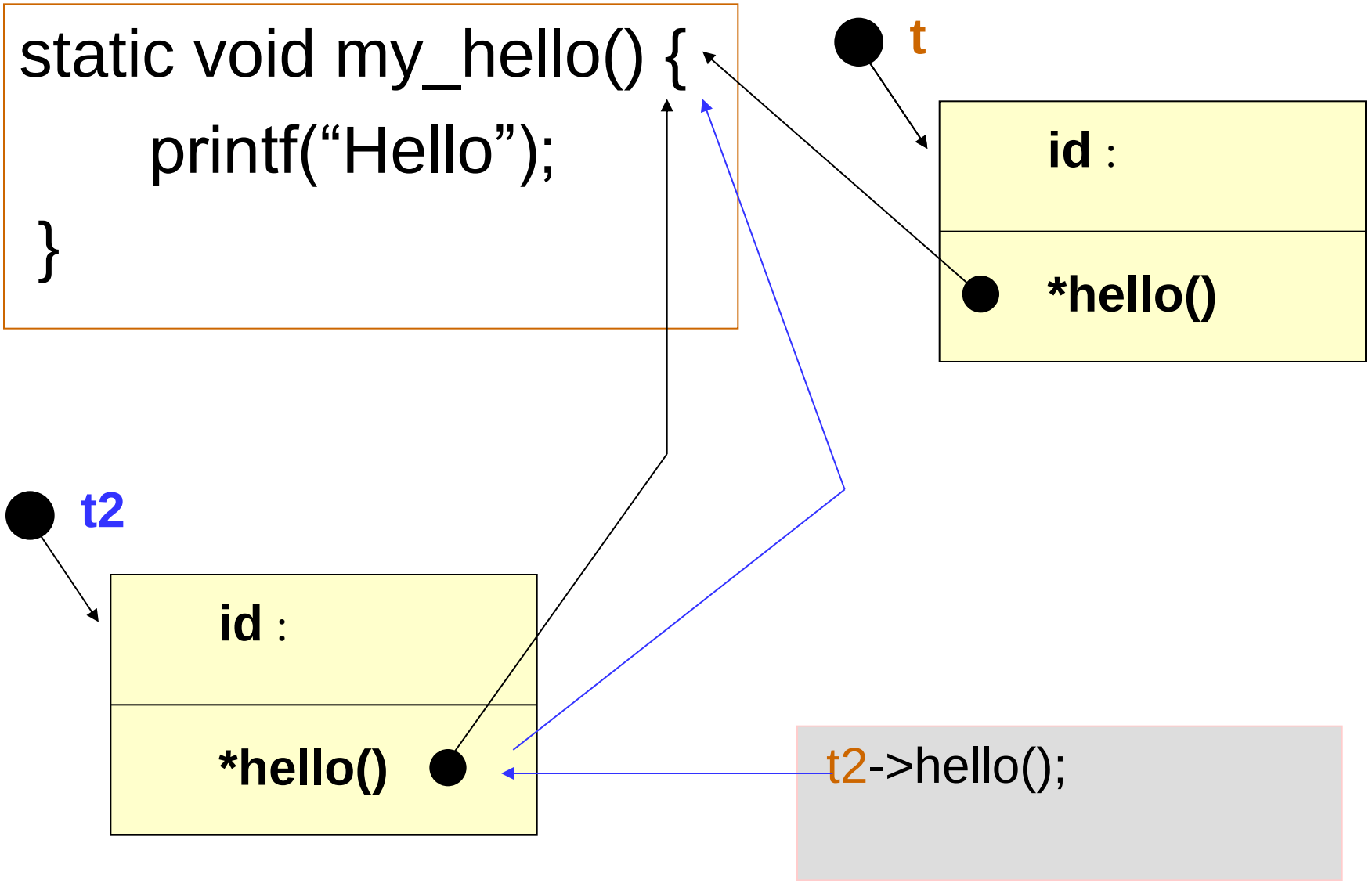





```
t->hello = my_hello;
```

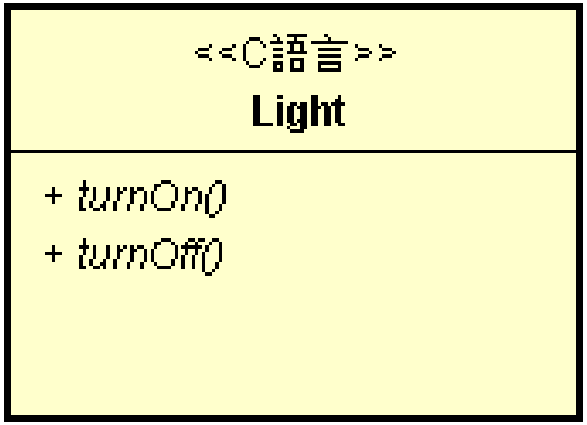

static void my_hello() {
printf("Hello");
}
t
id :
*hello()
t2
id :
*hello()
t2->hello = my_hello;


static void my_hello() {
printf("Hello");
}
t
id :
*hello()
t2
id :
*hello()
t->hello();


static void my_hello() {
printf("Hello");
}
t
id :
*hello()
t2
id :
*hello()
t2->hello();

範例

- 兹以C来定义一个Light类(class)，创建其对象(object)，并调用其函数。



定义Light类

```
struct Light {
    void (*turnOn)();
    void (*turnOff)();
};
typedef struct Light Light;
```

撰写函数：

```
static void turnOn(){
        printf( “ON” );
 }
static void turnOff() {
        printf( “OFF” ); }

struct Light * LightNew(){   // 構造式
       struct Light *t;
       t = (Light *)malloc(sizeof(Light));
       t->turnOn = turnOn;
       t->turnOff = turnOff;
       return (void*) t;
 }
```

创建对象，调用函数：

```
void main() {
        Light *led = LightNew();
        led->turnOn();
        led->turnOff();
    }
```

定義結構

```
typedef struct Light Light;
struct Light {
    void (*turnOn)();
    void (*turnOff)();
};
```

撰寫構造式

```
struct Light * LightNew(){
      struct Light *t;
      t = (struct type *)
            malloc(sizeof(struct type));
      t->turnOn = turnOn;
      t->turnOff = turnOff;
      return (void*) t;
 }
```

撰寫函數

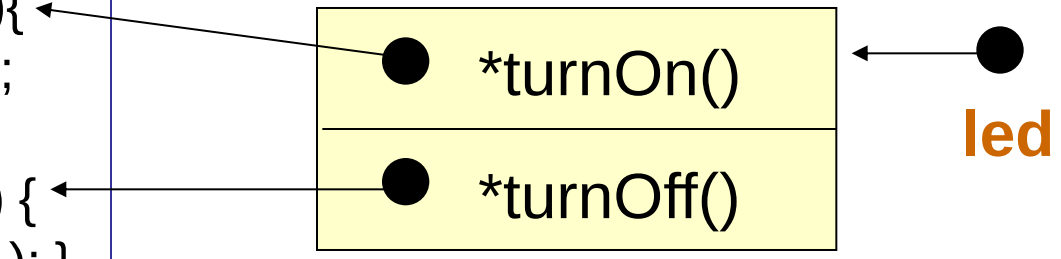
```
static void turnOn(){
      printf( “ON” );
 }
static void turnOff() {
      printf( “OFF” ); }
```

```
typedef struct Light Light;
struct Light {
    void (*turnOn)();
    void (*turnOff)();
};
```

```
struct Light * LightNew(){
      struct Light *t;
      t = (struct type *)
            malloc(sizeof(struct type));
      t->turnOn = turnOn;
      t->turnOff = turnOff;
      return (void*) t;
 }
```

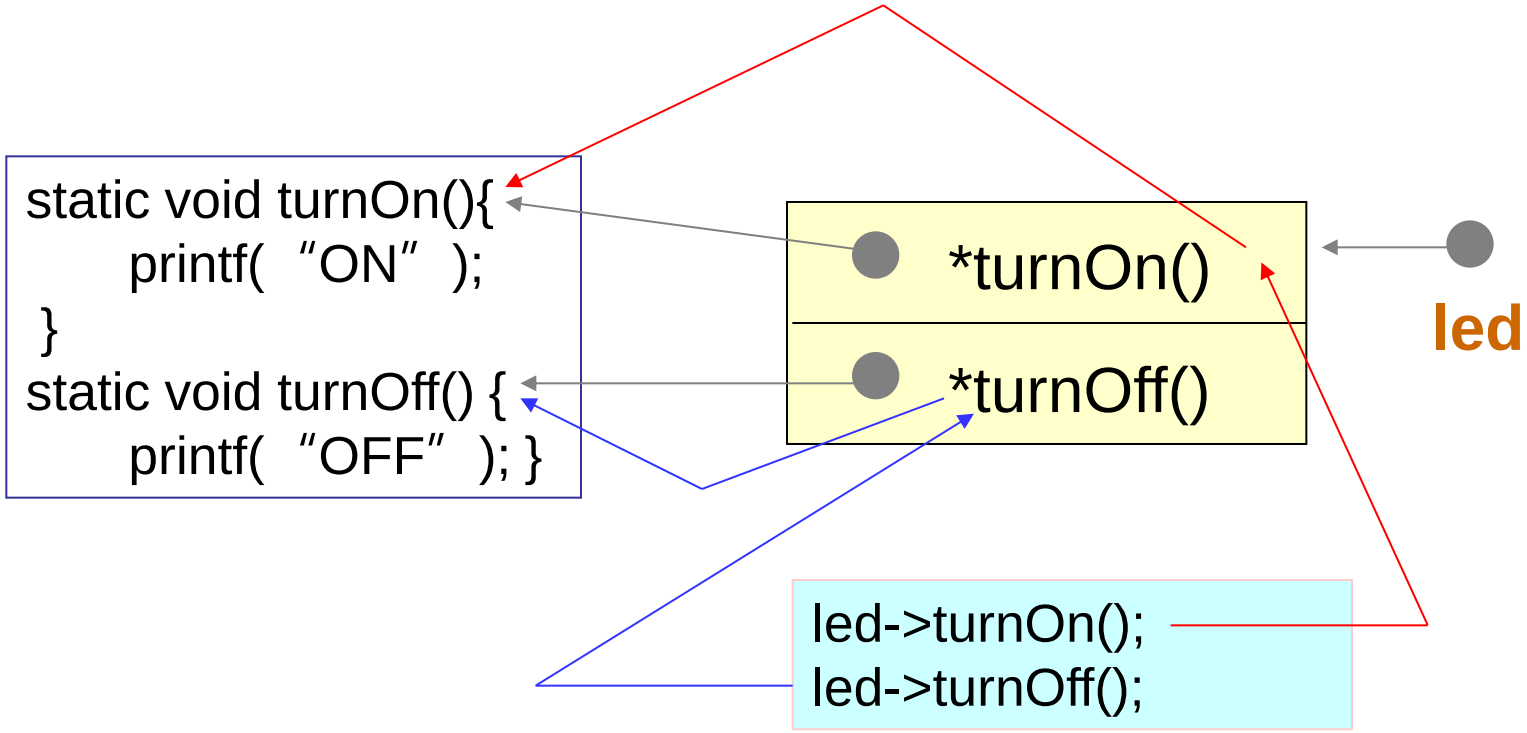
<<new>>

```
static void turnOn(){
      printf( "ON" );
 }
static void turnOff() {
      printf( "OFF" ); }
```




```
Light *led = LightNew();
```

調用函數


static void turnOn(){
printf(“ON”);
}
static void turnOff() {
printf(“OFF”); }
*turnOn()
*turnOff()
led
led->turnOn();
led->turnOff();

2.5 在C函数里存取对象的属性(attribute)值

- 刚才调用C函数时，其函数并没有存取(access)对象里的属性或数据。

請看一個新範例

定义Light类

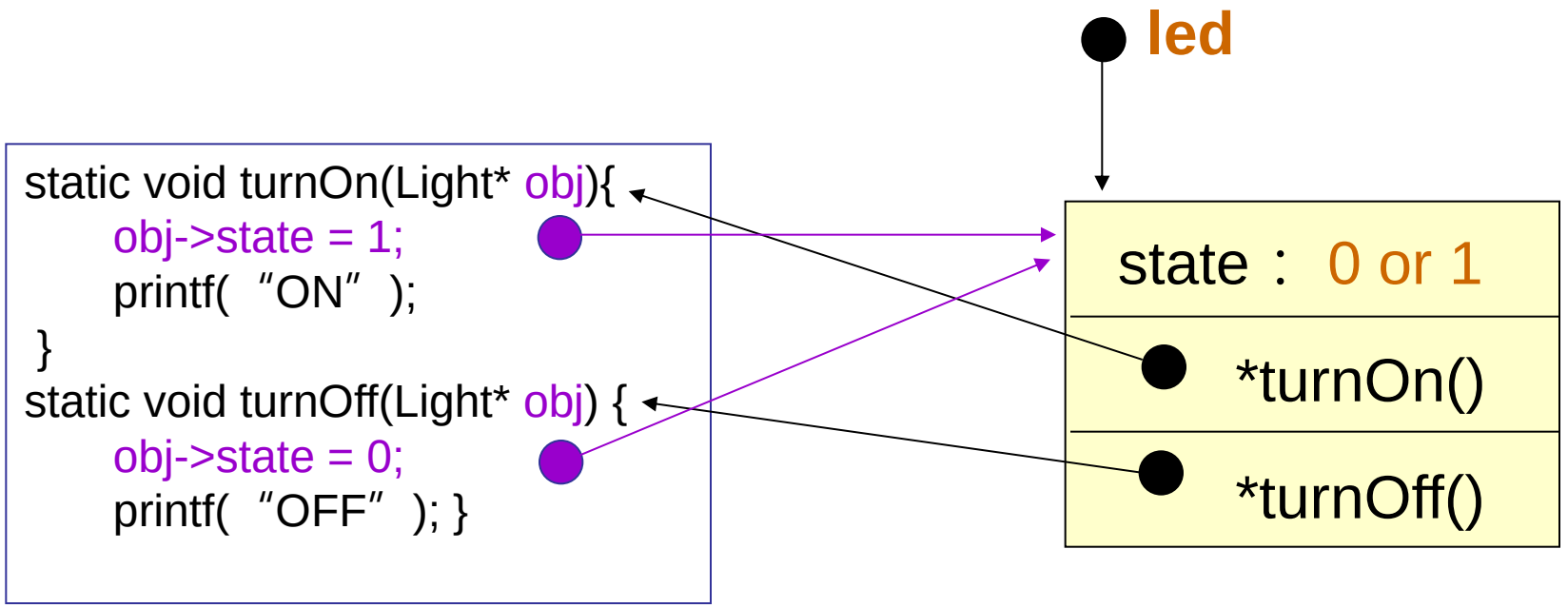
```
typedef struct Light Light;
struct Light {
    int state;
    void (*turnOn)(Light*);
    void (*turnOff)(Light*);
};
```

撰写函数：

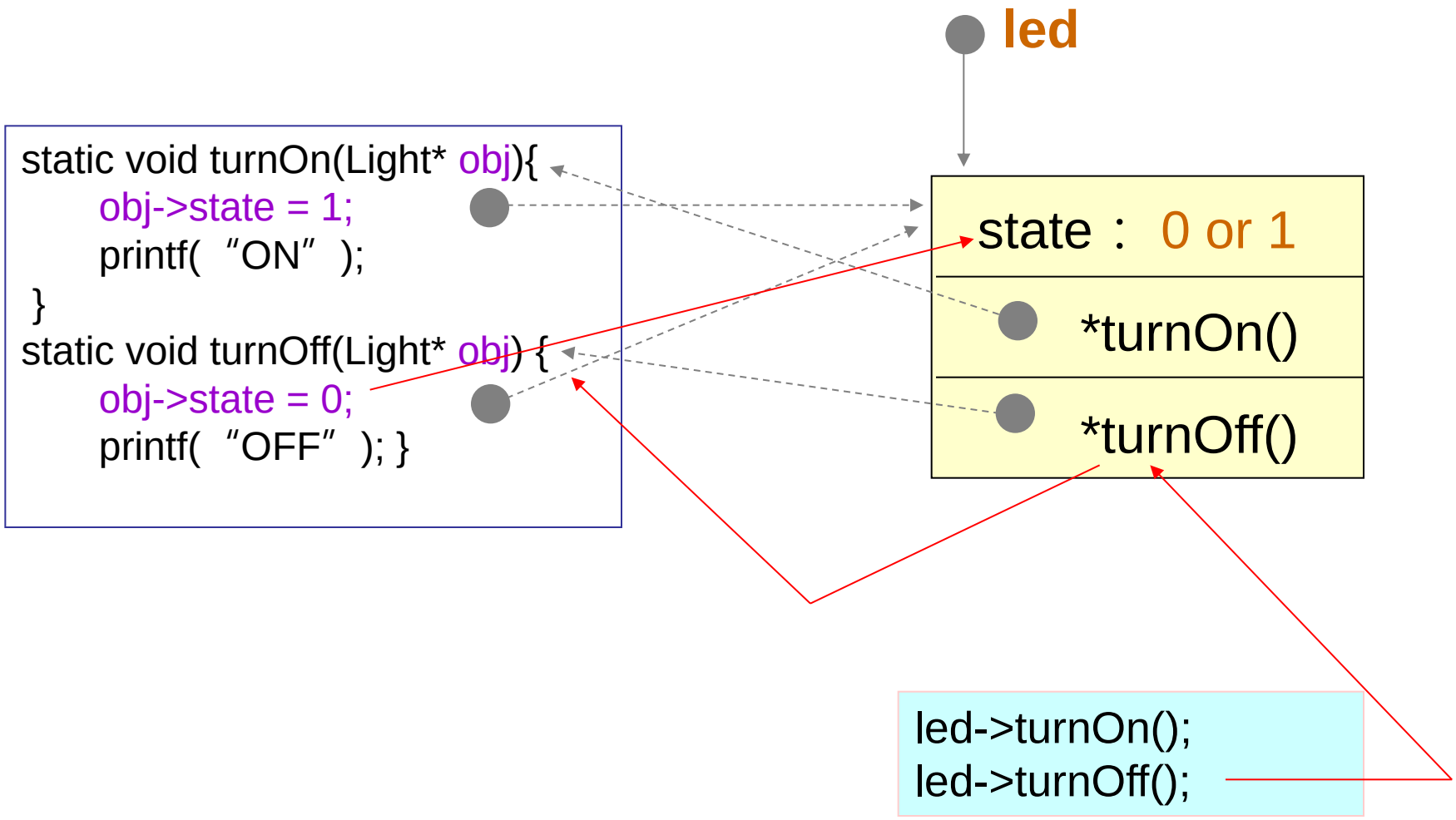
```
static void turnOn( Light *cobj ){
        cobj->state = 1;
        printf( “ON” );
 }
static void turnOff( Light *cobj ) {
         cobj->state = 0;
         printf( “OFF” );
}

struct Light *LightNew(){   // 構造式
      struct Light *t;
      t = (Light *)malloc(sizeof(Light));
      t->turnOn = turnOn;
      t->turnOff = turnOff;
      return (void*) t;
 }
```

创建对象，调用函数：

```
void main() {
    Light *led = LightNew();
    led->turnOn( led );
    led->turnOff( led );
  }
```




```
Light *led = LightNew();
```


led
static void turnOn(Light* obj){
obj->state = 1;
printf("ON");
}
static void turnOff(Light* obj) {
obj->state = 0;
printf("OFF"); }
state : 0 or 1
*turnOn()
*turnOff()
led->turnOn();
led->turnOff();

~ Continued ~