【第十二章】

繼承與多型

講師: 李根逸 (Ken-Yi Lee), E-mail: feis.tw@gmail.com

★ ★ ★ ★

★ ★ ★ ★

《範例》函式、結構與類別

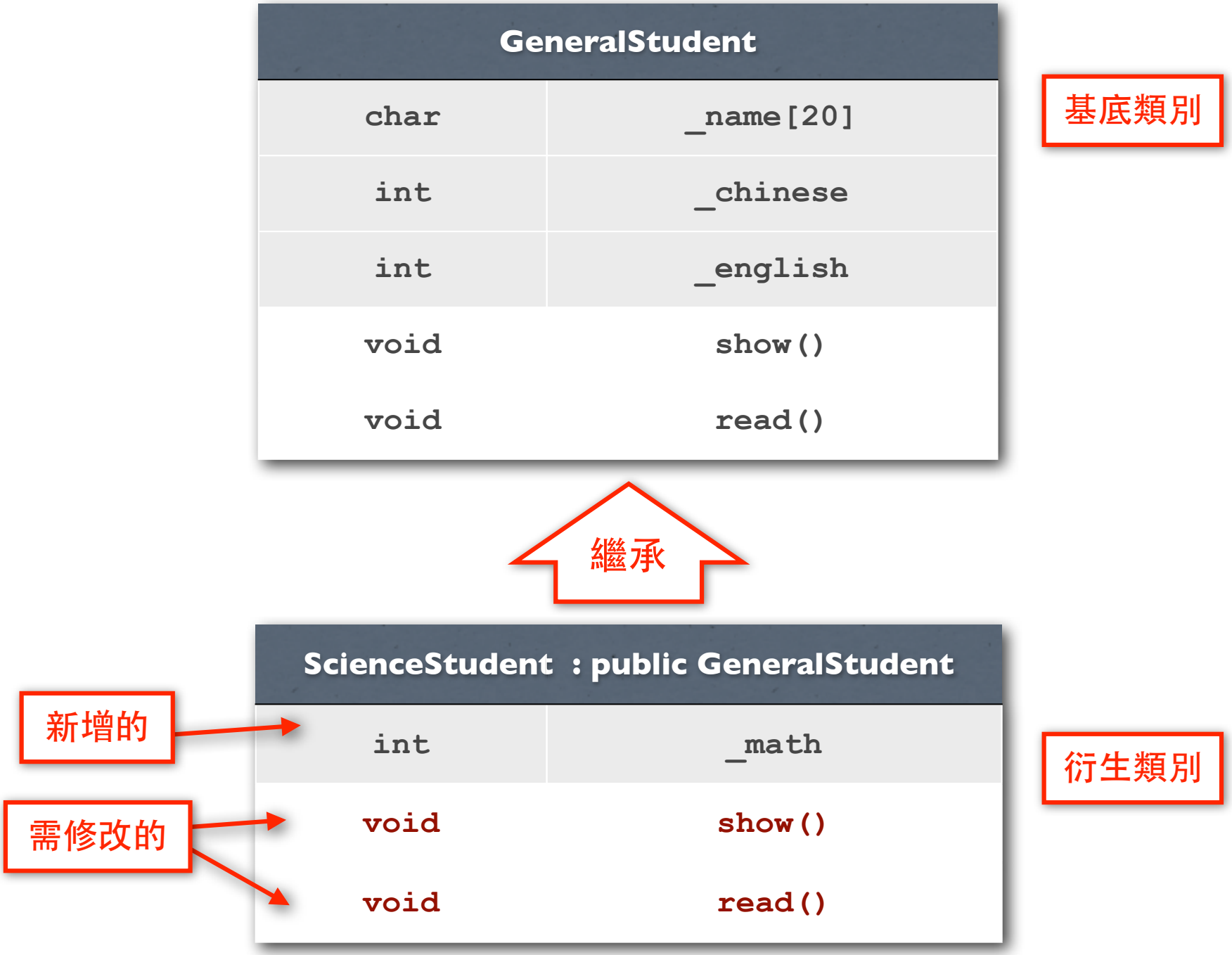
- 修改 **origin.cpp** 為 **origin-all.cpp** 將其選單增加一指令顯示全部資料
- 修改 **origin-all.cpp** 為 **origin-math.cpp** 將其增加數學 **(math)** 一科目，計算平均時以三科總和除以三計算
- 完成 **func.cpp** ，如 **origin** 依序制作另兩檔
- 完成 **struct.cpp** ，如 **origin** 依序制作另兩檔
- 完成 **class.cpp** ，如 **origin** 依序制作另兩檔

繼承的使用

- 情境：
  - 某學校科系具有中文 (chinese) 與英文 (english) 兩門必修科目，而理學生需要多修數學 (math)

GeneralStudent	
char	_name[20]
int	_chinese
int	_english
void	show()
void	read()

ScienceStudent	
char	_name[20]
int	_chinese
int	_english
int	_math
void	show()
void	read()


GeneralStudent
char _name[20]
int _chinese
int _english
void show()
void read()
基底類別
繼承
ScienceStudent : public GeneralStudent
新增的
需修改的
int _math
void show()
void read()
衍生類別

《範例》繼承範例

- 繼承語法：

```
class 衍生類別名稱 : public 基底類別名稱 {
  ...
};
```

  - 注意的是衍生類別的新增成員在繼承之後無法存取基底類別原有的 private 成員
    - 需要的話要將基底類別的 private 成員改為 protected 成員
    - 在 protected 的情況下，可以被衍生類別的成員存取，但不可被其他的函式存取

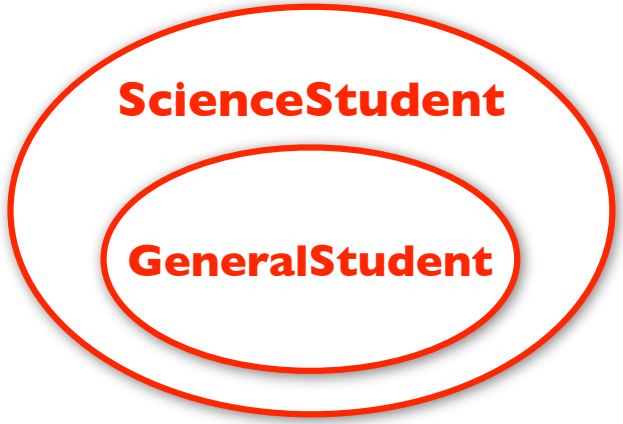
多型的使用

- 當 **ScienceStudent** 繼承 **GeneralStudent** 之後，可以將 **ScienceStudent** 當做 **GeneralStudent** 的延伸

```
ScienceStudent t;
GeneralStudent s = t; /* 合法但是 s 裡面沒有 math 的資料 */
s.read();    /* 這裡的 read() 是 GeneralStudent::read()*/
```

  - 不丟失資料的轉型是用指標實現：

```
ScienceStudent t;
GeneralStudent *s = &t;
s->read();
```



  - 而 C++ 裡為了效率，讓事情沒那麼單純
    - 因為現在有 GeneralStudent::read() 跟 ScienceStudent::read()

當用指向 **ScienceStudent** 的 **GeneralStudent** 指標呼叫 **GeneralStudent** 設定為 **virtual** 的函式時，才會去找看看 **ScienceStudent** 是不是有這個函式。

```
class GeneralStudent {
  virtual void read();
};
```