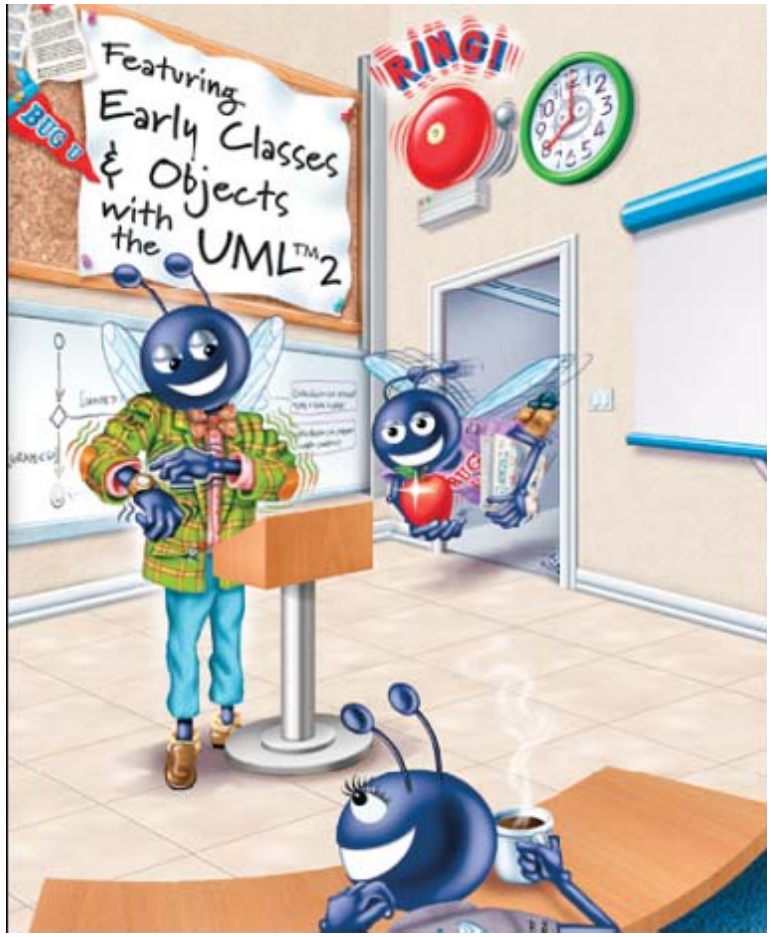
C++程序设计



复习

- Better C
- 类的使用
- 运算符重载
- 继承
- 多态
- 模板
- 异常
- 文件的处理

1 Better C

- 默认参数
- 函数重载
- 引用的概念
- const的使用
- 变量的作用域
- 指针的使用

1 Better C

- **默认参数**

```
int f( int,  int = 5,  double = 3.1 );

....

g = f(5, 2, 6.2) + f(3, 7) + f(4);

// same as g = f(5, 2, 6.2) + f(3, 7, 3.1) + f(4, 5, 3.1);
```

1 Better C

- **函数重载**

```
int f(int n) { ..... }

int f(double x) { ..... }       // overloaded function

int f(int m, int n) { ..... }   // overloades function

.....

g = f(-3) + f(5.0) + f(3, 5);
```

1 Better C

- ***函数签名*** ： 函数名 + 参数个数 + 参数类型

```
int g(int n) { ..... }
double g(int a) { ..... }                    // Error
int g(const int x) { ..... }                 // Error
int g(int &n) { ..... }                      // Error
int g(int a, int b=0) { ..... }              // Error
```

Which one 'g(3)' call?

1 Better C

- 引用

```
int a = 5;

int &b = a;  // ‘b’ is alias for ‘a’
```

1 Better C

●引用

```
void incr(int &a)
{
   ++a;
}
……
int b = 3;
incr(b);
```

```
// In C, Call-by-Value
 void incr(int *a)
{
   ++*a;
}
int b = 3;
incr(&b);
```

1 Better C

●引用

1. 引用被创建的同时必须被初始化（指针则可以在任何时候被初始化）
2. 不能有NULL引用，引用必须与合法的存储单元关联（指针则可以是NULL）
3. 一旦引用被初始化，就不能改变引用的关系（指针则可以随时改变所指的对象）

```
int i = 5;
int j = 6;
int &k = i;
k = j;
```

// k = j并不能将k修改成为j的引用，只是把k的值改变为6

1 Better C

●引用

```
int* f(int* x) {
  (*x)++;
  return x;//Safe, x is outside this scope
}
```

```
int& g(int& x) {
  x++;
  return x;//Safe, x is outside this scope
}
```

1 Better C

- 引用

```
int& h() {
  int q;
//return q;    // Error
  static int x;
  return x;//Safe, x lives outside this scope
}
```

2 类的使用

```
1  class Time {
2
3      public:
4         Time();           // constructor
           // set hour, minute, second
5          void setTime( int, int, int );
           // print universal-time format
6          void printUniversal();
           // print standard-time format
7         void printStandard();
8
9      private:
10       int hour;     // 0 - 23 (24-hour clock format)
11       int minute;   // 0 - 59
12       int second;   // 0 - 59
13
14    }; // end class Time
```

2 类的使用

- 防止类的重复定义

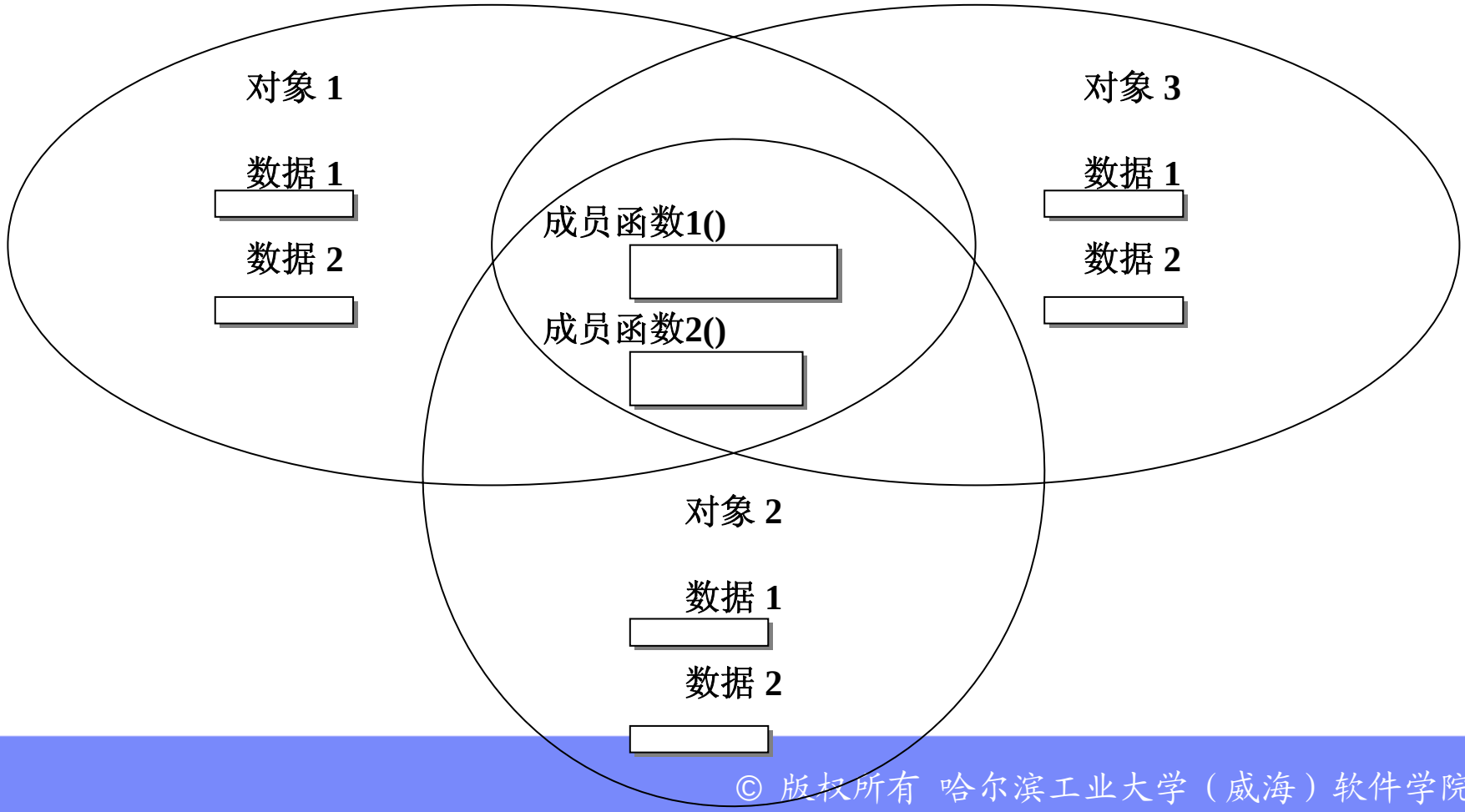
```
#ifndef TIME1_H
#define TIME1_H

    ……

#endif
```

2 类的使用

- 对象的内存布局



2 类的使用

- 构造函数和析构函数的调用顺序
  - 全局对象
  - 静态对象
  - 局部对象

2 类的使用

- const 对象和 const 成员函数
  - const 对象需要 const 成员函数
  - const 数据成员的初始化

2 类的使用

- 合成：对象作为类成员
  - 使用成员初始化语法来初始化成员对象的优点
  - 构造函数和析构函数的调用顺序

2 类的使用

- 友元函数和友元类
  - 目的
  - 性质
  - 缺点

2 类的使用

- this 指针
  - this指针是什么
  - 如何通过this指针访问数据
  - 如何实现函数的级联调用

2 类的使用

- static 成员变量和成员函数
  - 作用范围
  - 访问方式

2 类的使用

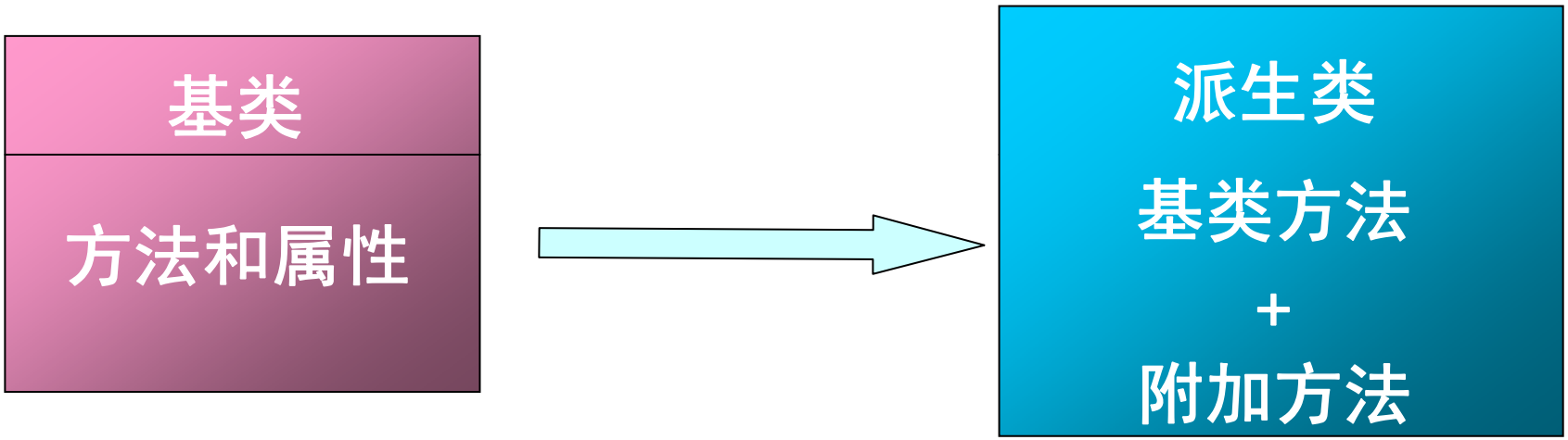
- new 和 delete
  - 隐含的做了哪些工作
  - delete 的注意事项

3 运算符重载

- 目的
- 方式：友元函数、成员函数
- 默认赋值函数、默认拷贝构造函数
- 案例分析：string 类

4 继承

继承是允许重用现有类来构造新类的特性



4 继承

- public 继承
- protected 访问控制的作用
- 基类构造函数的调用
- 构造函数和析构函数的调用顺序

5 多态

- 派生类对象可被看作是基类对象
- 没有多态概念时：
  - 指针或引用的类型决定其调用的函数
- 虚拟函数
- 抽象类、纯虚函数
- 指针或引用指向的对象类型来决定其调用的函数版本
- 虚析构函数

6 模板

- 模板的概念和作用
- 会书写简单的模板类

7 异常

- 什么是异常
- 捕捉异常的意义
- try、catch、throw 的书写

8 文件处理

- 文件的打开
- 文件的读写
- 文件的关闭

考试题型

- 选择
- 判断
- 改错
- 阅读代码
- 问答题
- 编程