

C++面向对象程序设计

第一讲：面向对象程序设计概述

HELLOWORLD

- C

- `#include <stdio.h>`
- `int main() {`
 - `printf("Hello World\n");`
 - `return 0;`
- `}`

- C++

- `#include <iostream>`
- `using namespace std;`
- `int main() {`
 - `cout << "Hello World"`
 - `<< endl;`
 - `return 0;`
- `}`

打印一个由“*”组成的三角形

- C

```
#include <stdio.h>
```

```
int main(void){
```

```
    int i, j, k;
```

```
    for(i = 0; i < 11; i++) {
```

```
        k = (21 - (i * 2 + 1)) / 2;
```

```
        for(j = 0; j < k; j++) {
```

```
            printf(" ");
```

```
        }
```

```
        for(j = 0; j < i * 2 + 1; j++) {
```

```
            printf("*");
```

```
        }
```

```
        printf("\n");
```

```
    }
```

```
    return 0;
```

```
}
```

打印一个由“*”组成的三角形

- C++

```
#include <iostream>
using namespace std;
int main() {
    int i, j, k;
    for(i = 0; i < 11; i++) {
        k = (21 - (i * 2 + 1)) / 2;
        for(j = 0; j < k; j++) {
            cout << " ";
        }
        for(j = 0; j < i * 2 + 1; j++) {
            cout << "*";
        }
        cout << endl;
    }
    return 0;
}
```

程序设计范型

程序设计范型是设计程序的规范、模型和风格，它是一类程序设计语言的基础。

- 过程范型
 - 程序 = 过程 + 调用
 - C、Pascal、Fortran等
- 对象范型
 - 程序 = 对象 + 消息
 - C++、Java、C#等
- 其他
 - 函数范型（Lisp）、模块范型（Modula）、逻辑式范型（PROLOG）等

过程范型的局限性

- 生产效率低下
 - 缺乏大粒度、可重用的软件实体
 - 不利于软件生产的工程化和自动化
 - 数据与其操作分离，且数据的操作分散于程序的不同地方
 - 不利于程序的可修改性和可维护性
- 难以应付日益庞大的信息量和多样的信息类型
 - 不再是单纯的计算问题，而是涉及文本、图形、图像、音频、视频等更为复杂的自然信息和社会信息处理
- 难以适应各种新环境
 - 并行处理、分布式处理、网络、多机系统等

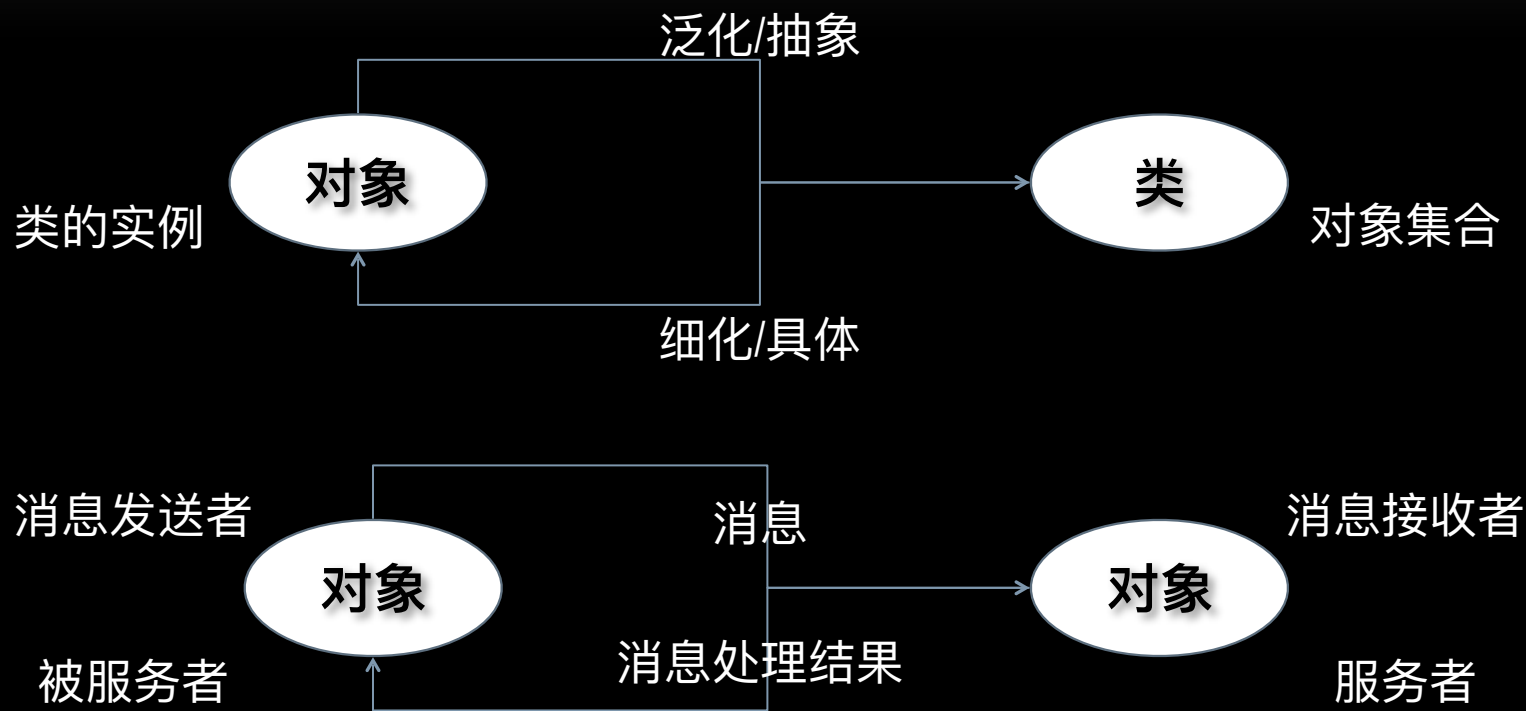
示例程序：P0101.exe 字符串拼接的C程序

对象范型（现实世界）

- 对象
 - 对象 = 静态属性 + 动态行为
 - 对象标识
 - 静态属性：描述对象的某些特征
 - 动态行为：对象的功用（作用于自身的行为和作用于其他对象的行为）
- 类
 - 具有相同属性和行为的对象的抽象（相似对象集的整体性描述）
- 消息传递
 - 对象之间的交互途径和方法

示例程序：P0102.exe 字符串拼接的C++程序

对象范型（现实世界）



对象范型（计算机世界）

- **抽象**

- 将有关事物（对象集）的共性归纳、集中的过程
- 简化复杂世界的表示，仅强调问题域感兴趣的信息
 - 学生成绩管理 VS 学生健康管理
- 抽象的目的是获取对象的属性和行为

- **封装**

- 封装：数据及其处理代码包装在一个实体中，对象间相互独立
- 信息隐藏：隐藏内部细节，只留下便于与外界联系和接收外界消息的接口
- 类 = 成员变量（对象的属性） + 成员方法（对象的行为）

对象范型（计算机世界）

- **继承**
 - 表达对象类之间相关的关系
 - 类间具有共享的属性和行为
 - 类间具有差别或新增的属性或/和行为
 - 类间具有层次结构
 - 父类与子类，直接父类与间接父类
 - 单继承与多继承
- **多态**
 - 不同对象对相同行为的处理差异（收到相同消息时，执行的操作不同）
 - 编译时的多态性
 - 运行时的多态性

对象范型的主要优点

- 可提高程序的重用性
- 可控制程序的复杂性
- 可改善程序的可维护性
- 能够更好地支持大型程序设计
- 增强了计算机处理信息的范围
- 能很好地适应新的硬件环境

面向对象程序设计语言

- 发展概况
- 典型的语言
 - Smalltalk
 - Simula
 - C++
 - Java
 - C#
 - Ruby
 - Object C
 - Object Pascal
 -

学习资源

- 课程资源
 - 讲义及示例程序 (<https://github.com/vchs6/MyCourse>)
- 编程工具
 - Qt+MinGW (<http://download.qt.io/archive/qt/>)
- 学习网站
 - The C++ Resources (Network <http://www.cplusplus.com/>)
 - C++技术网 (<http://www.cjjs.com/>)
 - 其他
 - <http://www.cprogramming.com/>、<http://www.cppreference.com/>
- 进阶学习
 - <http://www.boost.org/>、<http://www.cpp-home.com/>
 - <http://www.cnhonkerarmy.com/forum-36-1.html>