華中科技大學

面向对象程序设计

第一章 引论

许向阳

xuxy@hust.edu.cn

第一章 引论

華中科技大學

1.1 程序设计语言

1.2 程序编译技术

1.3 面向对象的语言及程序设计

1.4 面向对象的基本概念

1.5 C++语言的特点

1.6 C++程序结构

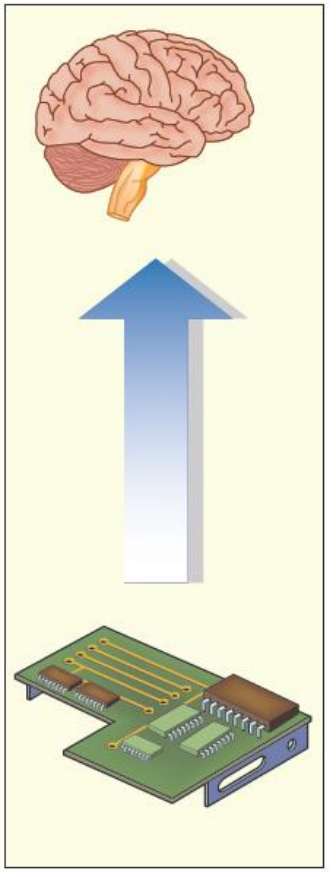
1.1 程序设计语言

- 已学过哪些程序设计语言？
- 创造这些语言的先后顺序是什么？
- 语言发展有什么规律？

1.1 程序设计语言

朝着更高级别的抽象发展

- ?
- declarative languages (Haskell, ML, Prolog...)
- OO languages (C++, Java, Python...)
- procedural languages (C, Fortran, COBOL...)
- assembly languages
- binary code



1.2 程序编译技术

- 能够直接被CPU识别的语言是机器语言
- 机器语言程序是过程性语言，step by step
- 高级语言程序如何变成机器语言程序？

1.2 程序编译技术

编译过程：

- 预 处 理：#define宏替换
  插入 #include 文件内容
  #if 条件编译处理 等
- 词法分析：产生一个程序的单词序列
- 语法分析：检查程序语法结构
- 代码生成：生成中间代码
- 模块连接：中间代码与库连接，形成一个可执行的程序

1.2 程序编译技术

- 模块连接：中间代码与库连接，形成一个可执行的程序
- 库：标准库、非标准库（自己的、第三方的）
- 静态连接

  被调用函数的函数体拷贝到可执行程序中，
  在编译时，由编译程序完成的连接。

- 静态链接库(Static Library，lib)

  包含了实际执行代码、符号表等

1.2 程序编译技术

- 动态连接

  在可执行程序中，保存被调用函数的描述信息，

  在运行时，由操作系统完成的连接。

- 动态链接库(DLL)
- 动态链接库的导入库(Import Libary，lib)

执行代码位于动态库中

导入库只包含了地址符号表等，确保程序找到对应函数

使用动态库、或静态库，各有什么优缺点？

1.2 程序编译技术

- 函数绑定（binding）

  函数的入口地址与函数调用相联系的过程。

- 静态绑定

  在程序执行前完成。

  由编译程序或操作系统装入程序计算函数的入口地址。

- 动态绑定

  在程序执行过程中完成。

  由程序自身计算函数的入口地址。

1.2 程序编译技术

- 函数绑定（binding）
- 静态绑定
- 动态绑定

- 使用动态链接库是否为动态绑定？

非也。使用动态链接库，也是静态绑定。

动态绑定是在程序运行前，都不知道会调用哪一个函数。

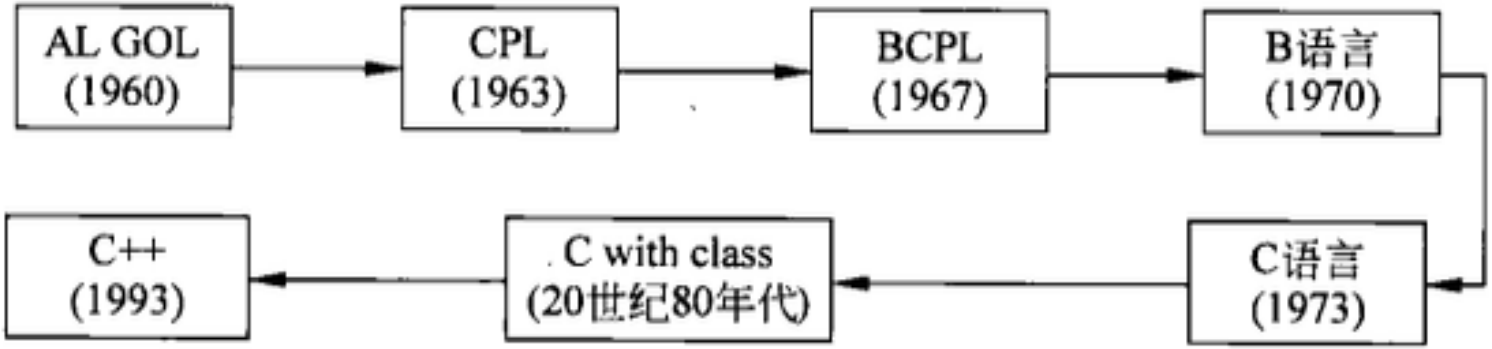
1.3 面向对象的语言及程序设计

面向问题的语言，离硬件比较远，不宜编写系统文件

剑桥大学推出，接近硬件，但规模较大，难以实现

剑桥大学Martin Richards对CPL进行了简化

AT&T Ken Thompson进一步简化，设计了很简单且很接近硬件的B语言，但功能有限



1983年正式命名为C++，1994年制定了ANSI C++标准草案。1998年批准为ISO C++，成为一个统一、完整稳定的系统，并拥有一个强大的标准程序库

AT&T Bjarne Stroustrup设计并实现了C语言的扩充，称为“带类的C”

AT&T D.M.Ritchie在B语言的基础上设计了C语言。既保持了B语言的优点，又克服了它们的缺点(过于简单，无数据类型)

1.3 面向对象的语言及程序设计

1985年， C++1.0 AT&T 前端翻译器，将C++转换成C代码

1988年，真正的C++编译器诞生

1989年， C++2.0 AT&T 类的多继承

1991年， C++3.0 模板、异常处理 VC++1.0

1993年， 运行时类型识别（RTTI) 、名字空间

1997年， 美国国家标准

1998年， C++4.0， 国际 C++标准

VC++ 6.0

2003年后，VC++ 各版本 集成在Visual Studio中

最新版本 VS 2019 本课程的实验平台

1.3 面向对象的语言及程序设计

- **纯OO型语言**

  程序全部由类构成。

  JAVA、 C# 、SMALLTALK等

- **混合型OO语言**

  程序由类、过程或函数以及变量定义构成。

  C++、VB.NET。

1.3 面向对象的语言及程序设计

- **面向对象程序设计的四个阶段**

  系统分析、系统设计、对象设计和对象实现

- **系统分析**

  建立对象模型、动态模型、功能模型

  对象模型最为重要，描述对象、类型之间的关系。

- **系统设计**

- **对象设计**

  细化系统分析建立的三个模型

- **对象实现**

  与采用的程序设计语言有关

1.4 面向对象的基本概念

对象：现实世界具体的或抽象的“事物”。

类：描述对象**特征**和**行为**的集合体。

是一种复杂的数据**类型**，是一种**模板**；

将不同类型的数据及对它们的操作封闭在一起。

抽象：

从对象(事物)到类型(概念)

从低级类型(概念)到高级类型(概念)。

1.4 面向对象的基本概念

封装（Encapsulation）

继承 （Inheritance ）

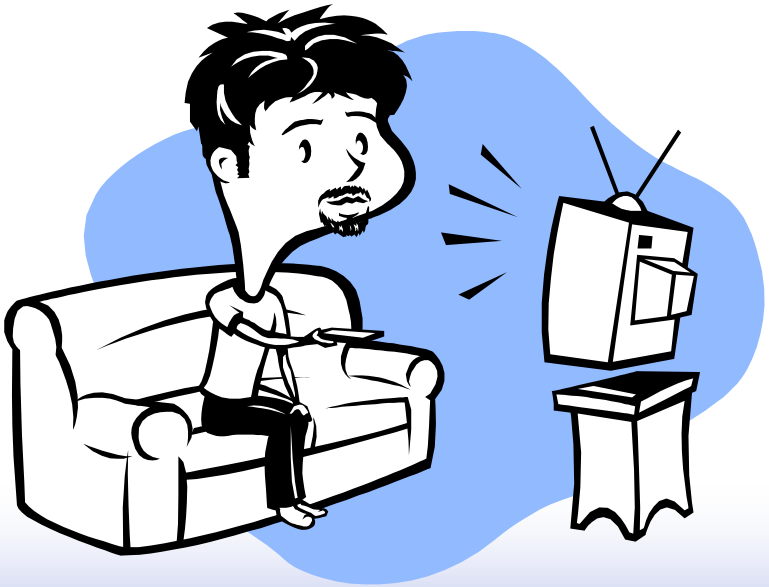
多态（Polymorphism）

1.4 面向对象的基本概念

交互

对 象：观众、遥控器、电视机

对象之间的交互：消息机制



消息：一个对象向另一个对象发出的请求。

要求某个对象执行某个功能操作的规格说明。

1.4 面向对象的基本概念


```
struct  student
{         int  name[15];
          int  age;
          void getold() {  age=age+1;  }
          void tellname()   {
                cout<<"My name is "<<name<<endl;
          }
};

struct  student  zhang;
strcpy_s(zhang.name,"ZhangSan");
zhang.age = 22;
zhang.tellname();
zhang.getold();
```

例程：
C1_simple_oo…

zhang . tellname ( )
接受消息的对象 消息名 参数

1.4 面向对象的基本概念

- 函数调用：实现消息传递机制
- 参数和返回值：实现信息传递
- 函数体：完成收到消息后的处理

直接交互：指一对象调用另一对象的操作、功能或函数；

间接交互：通过发送或监听消息完成。

C++程序的对象既可以直接交互，也可以通过操作系统提供的消息机制间接交互。

1.4 面向对象的基本概念

面向对象概念	C++术语
对象	对象
类	类
方法、行为	成员函数
特征	数据成员
消息	调用成员函数
子类	派生类
继承	派生

总结

面向对象的概念和关键特性

- 类、 对象
- 抽象、实化
- 封装、继承、多态
- 对象交互、消息机制
- 静态连接、动态链接
- 静态绑定、动态绑定