面向对象技术与编程

C++ How to Program（9th）

西安财经学院 信息学院

关于课程

- **课程名称**

  面向对象技术与编程

  面向对象技术与编程课程设计

- 课时：36(授课)+36(实验) +72(课后)
- 课程目标
  - 了解面向对象的基础理论
  - 充分掌握C++面向对象的编程技术和编程机制
  - 利用C++编写面向对象的程序
  - 初步掌握STL编程基础

课程内容-授课内容

- 概述
- C++的变迁
- 类和对象
- 运算符重载
- 继承
- 多态
- 模板
- 流
- 异常处理
- STL编程基础

课程内容-上机内容

- 基本要求
  - 一个完整的小项目，分多次完成
- 主要内容
  - 类的构造函数和析构函数
  - 运算符重载机制
  - 完成类的继承机制
  - 利用虚函数实现多态机制
  - 异常处理机制
  - C++流技术
  - 应用标准模板库

考核方式

- 作业成绩20%
  - 每次课后上机都要求完成指定的作业
  - 取课后作业的加权和作为平时成绩
- 课堂表现10%
  - 课堂点名，提问（主要针对自学内容）
  - 上机课抽查
- 期末考试70%
  - 课程结束后安排期末考试(上机考试，4小时)

参考资料

C++大学教程（第五版）， Harvey M. Deitel, Paul James Deitel, (C++ How to Program, Fifth Edition), 电子工业出版社【英文版最新第9版】(较易)
C++ Primer，Stanley B Lippman, Josée Lajoie著，潘爱民，张丽译，中国电力出版社【中英文第4版】(较易)
C++语言程序设计（第4版）郑莉，董渊，清华大学出版社
C++语言程序设计教程（第二版）沈显君，杨进才，清华大学出版社
更多的网络资源：MSDN、搜索引擎、……

为什么要学程序设计？

- 我们学习的是计算机类（新工科）专业
- 程序设计是计算机类课程的基础
- 程序设计是计算机类工科技术人员**必备的**基本技能


如何学好程序设计

- 原理为纲
  - 语言复杂的表面都是简单原理的外在表现
- 面向应用
  - 书本和课堂只能教会基本原理，写不出自己的程序
  - 只有面向实际应用，在实践中有目的去学、去用才能真正掌握
- 实用为美
  - 写简单实用的程序，不应过分追求复杂、完美。就如同平时的说话不可能采用莎士比亚歌剧中的表达方法
- 无需背诵
  - 该记的，用着就记住了；用不着的，背下来也会忘
- 没有绝对
  - Match is best!

做一个成功的开发人员…

- 以学习英语的方式学习程序设计语言，培养“语感”
- 培养自己的耐心，特别是在调试和学习阶段
- 懂得坚持自己的开发思路，并懂得理解与吸收别人的思想
- 懂得向你周围的人学习，不论是开发能力上的还是开发经验上的
- 拓宽自己的知识面，并且能够及时补充自己的知识和完善自己的知识结构。对于大部分人来说，如果要写好程序，还需要其它领域的知识

对于希望从事软件系统开发的新手

- 精通一门语言：C / C++ / Java / C# / ……
- 掌握一种开发工具： Visual C# .NET、 Visual C++、Eclipse、NetBeans……
- 熟悉“和选定的语言、开发工具相关”的平台： .NET、J2EE、MFC……
- 熟悉一种数据库产品： SQL Server、 Oracle、DB2、MySQL、……
- 了解某个行业的业务知识：电信、银行、电力、制造业等（可以工作后再学。但有时为了得到一份好的工作，可以提前掌握一些）

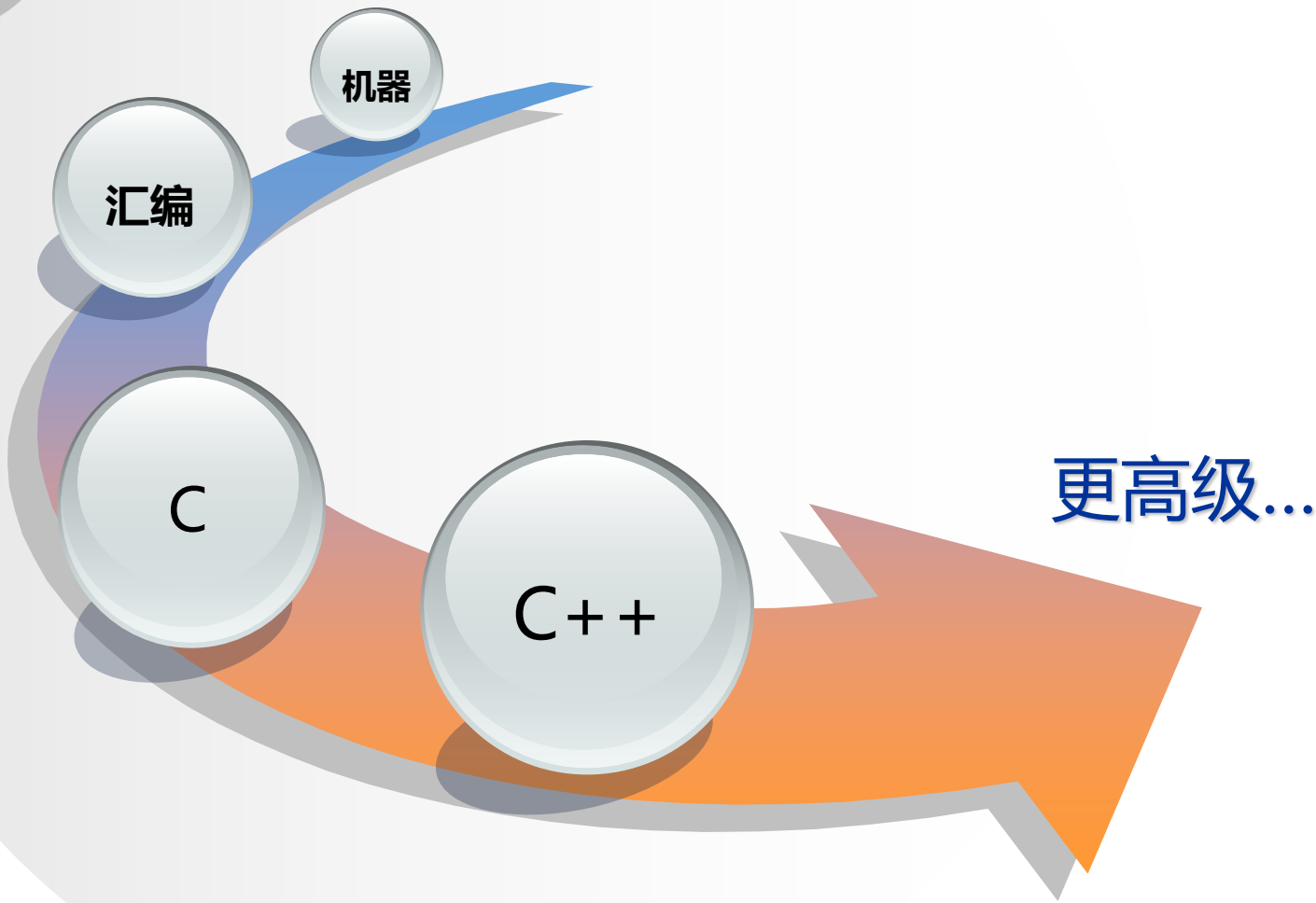
言归正传……
（C语法总结）


第一章 C++的变迁

主要内容

1. 程序设计语言的变迁
2. C++发展之路
3. 了解标准C++
4. 从字符串处理看C++变化

1. 程序设计语言的变迁



程序设计语言：人与计算机对话

- 两个说不同语言人的对话方式：
  - 一方学习另一方的语言
  - 双方都学习另一种第三方语言
- 人与计算机的对话方式：
  - 计算机学习人的语言：自然语言
  - 人学习计算机的语言：机器语言
  - 学习第三方语言：高级程序设计语言

计算机程序的特点

- 计算机的工作是用程序来控制的。
- 程序是指令的集合。
- 指令是计算机可以识别的命令。

机器语言

- 计算机可以直接识别（二级制字符串）

```
10111000
01000001
00000000
00000101
01000001
00000000
```

汇编语言

- 用英语缩写助记符来表示

MOV AX, 1	10111000 01000001 00000000
ADD AX, 1	00000101 00100001 00000000

与人类的思维相差甚远。抽象层次太低，程序员需要考虑大量的机器细节。

- 用类似于日常用语的形式指令，单条语句可以包含很多任务。

```
#include <stdio.h>
main()
{
    int sum=0;
    for(int i=0;i<10;i++)
        sum+=i;
}
```

面向对象的语言

- 出发点：
  - 更直接地描述客观世界中存在的事物（对象）以及它们之间的关系。
- 特点：
  - 是高级语言。
  - 将客观事物看作具有属性和行为的对象。
  - 通过抽象找出同一类对象的共同属性和行为，形成类。
  - 通过类的继承与多态实现代码重用

C++ / 标准C++

```
#include <iostream.h>
int main()
{
    cout<<1+1<<endl;
}
```

C++

```
#include <iostream>
int main()
{
    std::cout<<1+1<< std::
    endl;
}
```

标准C++

高级程序设计语言的发展

- 50年代高级语言出现
  - 1951 Fortran I ，1954 Fortran II
  - ALGOL 58，ALGOL 60
  - COBOL 60
- 60年代奠基性研究
  - 编译技术的完善
  - 1967 BASIC （对促进中国的科技人员使用计算机功不可没）
  - 1971 PASCAL（在中国作为教学语言，没有推广使用）
- 70年代完善的软件工程工具
  - 1972 C
  - Ada 1975年，美国军方，历时8年

续

- 80年代面向对象发展
  - 1980 Smalltalk-80
  - 1982-1986 Object Pascal、Objective-C、Object Assemble
  - 1985 C++
- 90年代网络计算语言
  - 多范型、持久化、多媒体、平台无关
  - 1996 Java
- 本世纪
  - .Net和C#

程序设计语言现状

- 语言:
  - 高级语言： C、C++、Java、C#、Basic、Pascal...
  - 脚本语言：PHP、 Python、Ruby (On Rails)、JavaScript、Asp .Net、Perl、...
- 工具:
  - Microsoft
    - Microsoft Visual Studio .NET: C#、C++、Basic、Asp .Net
  - Open Source(开源)项目：gcc、Eclipse、NetBeans
  - Dev C++
  - Code Blocks

选择语言？

这么多语言，我该学哪个呢？

看两组统计数据(一)

四地区通用编程技术市场需求量对比				
语言	硅谷	北美	澳洲	中国
C/C++	45.8%	33.5%	20.4%	34.2%
Java	34.3%	34.4%	34.1%	36.8%
.Net	11.9%	29.1%	43.6%	27.5%
Python	8.0%	2.1%	1.9%	1.5%

1. 来自CSDN 2008年的统计数据
2. 未考虑Perl、Delphi等其他通用语言和工具

看两组统计数据(二)

四地区Web服务端技术市场需求量对比				
Web技术	硅谷	北美	澳洲	中国
Java	57.5%	50.4%	33.0%	34.4%
ASP .Net	9.0%	29.4%	44.7%	21.4%
PHP	25.0%	15.6%	19.3%	43.3%
Ruby	8.4%	4.6%	3.0%	0.9%
1. 来自CSDN 2008年的统计数据				

再看一组编程语言排名：TIOBE

Programming Language	2017	2012	2007	2002	1997	1992	1987
Java	1	1	1	1	12	-	-
C	2	2	2	2	1	1	1
C++	3	3	3	3	2	2	4
C#	4	4	7	13	-	-	-
Python	5	7	6	10	27	-	-
PHP	6	5	4	6	-	-	-
JavaScript	7	9	8	7	19	-	-
Visual Basic .NET	8	21	-	-	-	-	-
Perl	9	8	5	4	4	13	-
Assembly language	10	-	-	-	-	-	-
COBOL	24	32	16	5	3	15	8
Lisp	31	12	13	8	9	10	2
Prolog	33	38	25	21	18	14	3
Pascal	97	13	19	19	7	3	5

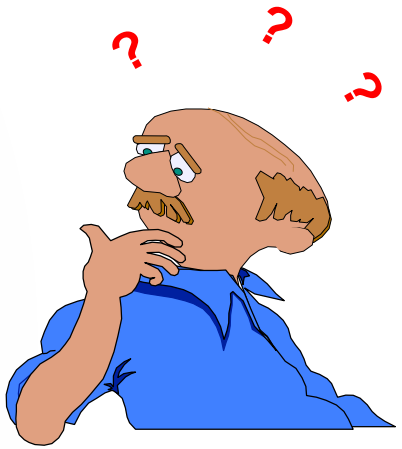
1. 来自TIOBE Programming Community (http://www.tiobe.com/)

结论：选择哪种语言

- C/C++、Java、.Net(C#)
  - C/C++：程序性能高、支持底层应用
    - 主要用于系统级软件、资源受限环境软件
    - 典型应用：通用软件、主机游戏、与硬件相关底层应用等
  - Java：跨平台，更好的互操作性
    - 主要用于大规模企业级应用软件，随着硬件的发展使得其在嵌入式领域(如手机软件)应用更加广发
    - 典型应用：电信、银行等行业管理信息系统
  - .Net：开发、部署效率高，成本低
    - 主要用于中小规模企业级应用软件
    - 典型应用：部门级信息系统、桌面应用软件
  - 其它脚本语言也有其应用环境，如Python、P
- 结论
  - 至少精通一种语言，深入理解该语言的各种特性。
  - 实际工作后，用什么学什么！

2. C++发展之路

为什么会有C++??



- C不够用了
  - 计算机的应用日益广泛
  - 大规模的复杂系统
  - 易用性、复用等问题
- 在C之上添加新的机制→ C++
  - 解决易用问题，如指针、内存分配等
  - 类和对象的概念→面向对象的编程机制
  - ……

C++的发展

- C语言(Dennis M. Ritchie)
  - B语言、Unix操作系统、C语言
  - 60年代末、70年代初
- C with class
  - 70年代末、80年代初
- C++语言(Bjarne Stroustrup)
  - 80年代
- ISO C++：标准C++
  - 1998年正式发布
  - The C++ Standard Library
  - 最新的标准C++ 0x版本已经发布

标准C++2.0现状

- C++0X (即标准C++ 2.0)的草案在2007年10月完成；目标是更好地应对多核时代
- C++0x的几个重大进展：
  - 1. 库增强：标准库TR1和TR2中的库增强组件
  - 2. 垃圾收集：符合C++0x的编译器必须提供垃圾收集器
  - 3. Concept：可简化泛型程序的开发
  - 4. 并发内存模型和并发库：使C++能够在多核时代健康成长
- 未被包括在C++0x中的重要特征包括：
  - 1. 统一的动态加载模型
  - 2. 模块

C++开发工具

- 当前可用的工具
  - Visual Studio (.Net)
  - 基于gcc编译器的各类共享软件: Eclipse CDT 、CodeBlocks、Dev C++...
- 历史上曾出现的开发工具
  - Borland C++、C++ Builder
  - Symantec C/C++
  - Watcom C/C++
  - IBM VisualAge C++
  - Sybase Optima++
  - ……
- “工欲善其事，必先利其器”。利用好的开发工具可以达到事半功倍的效果

3. 了解标准C++

- 标准C++完全兼容C语言、早期的C++语言
  - 相同功能的库函数可能在多个版本的C++存在
  - 程序设计人员应尽量使用标准C++中最新的头文件、库函数等内容
  - 有时为了兼容旧的C++编译器，可以在程序中应用一些预处理指令或其它手段

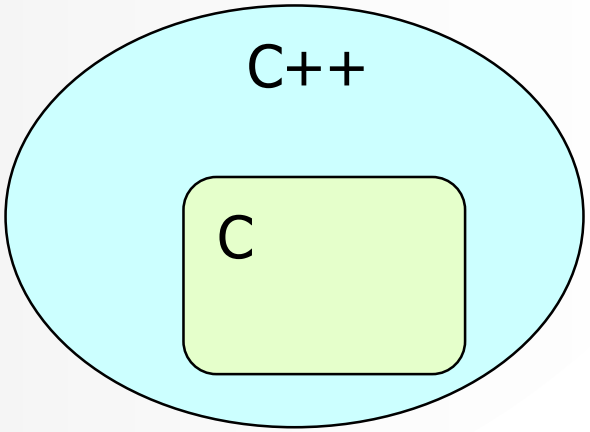
续

- C++是C的超集(Superset)

  Provides capabilities for object-oriented programming(OOP, 面向对象的程序设计方法)

- 一种混合语言(Hybrid language)

  Can program with, C-like style, Object-oriented style, or both



标准C++新增元素

- 名字空间
- 新的类型转换操作符
- bool数据类型
- 运算符关键字
- 与类相关的新特性
  - 运行时类型信息RTTI
  - explicit构造函数
  - mutable类成员
  - 类成员指针.*和->*
  - virtual基类
- 丰富的模板库STL

第一个C++程序

```
//2_1.cpp
#include <iostream>
using namespace std;
int main() {
      cout << "Hello!" << endl;
      cout << "Welcome to c++!" << endl;
      return 0;
}
```

运行结果：

```
Hello!
Welcome to c++!
```

标准C++头文件现状

- 标准C++头文件
  - 不再使用.h扩展名，全部在std名字空间
  - 如<iostream>等，所包含功能和对应旧头文件类似
- 旧的C++头文件名
  - 早期以.h为扩展名的C++头文件，没有名字空间
  - 如<iostream.h>等，但已不在官方标准中
- 标准C头文件
  - 标准C语言所提供的头文件，以.h为扩展名
  - 如<stdio.h>等
- 具有C库功能的新C++头文件
  - 无.h扩展名，文件名以c开头，并全部在名字空间std中
  - 如：<cstdio>代替<stdio.h>头文件，并提供类似功能

名字空间

- 程序在不同的范围包括许多不同的标识符
  - 当不同的范围重叠时，则可能导致问题
  - 名字空间污染（namespace pollution）
- 名字空间(namespace)定义了范围
  - 将变量和标识符名字空间里面

```
namespace namespace_name
{
  members
}
```

  - 通过“*名字空间::成员名*”的方式访问

```
namespace_name::member_name
```

  - 未命名的名字空间是全局
  - 名字空间可以嵌套

using关键字

- using关键字
  - 语法：using namespace *namespace_name*;
  - 作用：在该名字空间里面的成员可以直接访问，而不需要使用namespace_name::
  - 范例：
    - using namespace std;
      - 直接访问名字空间std所有标识符，而不需要前缀std
    - using std::cout
      - 仅开放名字空间std中的cout标识符，可以直接使用cout来代替std::cout
      - 若想使用std中的cin，则必须std::cin

cin，cout，endl，cerr

- 标准命名空间中定义了cin、cout、cerr、endl等

cin: 标准输入流（一般默认从键盘输入）

cout: 标准输出流（默认向显示屏输出）

cerr: 标准错误流

endl: 结束并换行（endline）

提取运算符>>和插入运算符<<

- 提取运算符>>与cin配合使用，表示从输入设备上提取输入信息到内存中。

  例如：

  ```
  cin>>x>>y>>z;
  ```

- 插入插入运算符<<一般与cout配合使用，表示将待输出内容插入到（屏幕）当前输出位置。

  例如：

  ```
  cout<<x<<“,”<<y<<endl;
  ```

4. 从字符串处理看C++变迁

- 有三个头文件中可用于处理字符串
  - <string.h>：旧的C头文件
  - <cstring>：旧C头文件的std版
  - <string>：新的标准C++头文件

```
#include <string.h>
#include <stdlib.h>
main(){
    char *str1;
    char *str2="Test" ;
    str1=malloc(5);
    strcpy(str1,str2);
    printf("%s", str1);
    free(str1);
}
```

```
#include <cstring>
#include <iostream>
using namespace std;
main(){
    char *str1;
    char *str2="Test" ;
    str1=new char[50];
    strcpy(str1,str2);
    cout<<str1;
    delete[] str1;
}
```

```
#include <string>
#include <iostream>
using namespace std;
main(){
    string str1;
    string str2("Test");
    str1=str2;
    cout<<str1;
}
```

编程练习

- 编写一程序，接受用户输入的一个字符串，以相反的顺序存储下来，然后输出到控制台。

- 主要难点
  - 如何接受用户输入：命令行参数
  - 如何存储字符串：字符串的处理
  - 如何输出到控制台

（1） 命令行参数

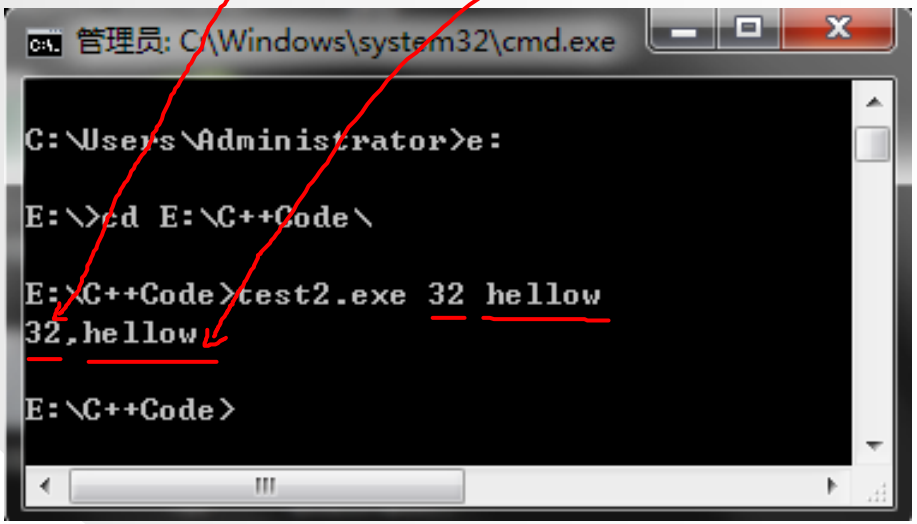
- 通常，main函数定义为int main( )，如果带参数，则格式为：

  int main(int argc, char* argv[]);

  - argc：命令行参数的个数，
  - argv：字符型指针数组，其各指针分别指向命令行中命令名和各个参数的字符串
  - 其中argv [0]指向命令名字符串，argc的取值从argv[1]开始（下标从1开始）

例如：

```
#include <iostream>
int main(int argc, char** argv) {
        std::cout<<argv[1]<<","<<argv[2]<<std::endl;
        return 0;
}
```



（2） 字符串处理

- C语言中没有“字符串”数据类型
  - 字符串被处理为由字符指针指向的、存储在字符数组里的字符序列
  - 最后加上了一个空字符‘\0’，作为字符串的结束标志
  - 字符串常量被编译程序自动转换成具有这种形式的数组，该数组的开始地址被作为字符指针值使用

C语言的字符串处理函数（回顾）

- #include <string.h>
  - strcpy(char *, const char *)
    - 字符串拷贝
  - strlen(const char *)
    - 取字符串长度
  - strcat(char *, const char *)
    - 字符串连接
  - strcmp(const char *, const char *)
    - 字符串比较
  - strstr(const char *, const char *)
    - 在主串中查找指定的子串

C语言中的动态内存分配（回顾）

- 头文件<stdlib.h>
- void* malloc(size_t size);
  - 向系统申请大小为size的内存块，把指向首地址的指针返回。如果申请不成功，返回NULL（一定要检查返回值）
- void free(void* block);
  - 释放由malloc()申请的内存块。block是指向此块首地址的指针（malloc()的返回值）
- 关于动态分配的内存
  - 在“堆（heap）”中分配，内容随机
  - 被free/delete之前，永久有效
  - 在被free/delete之后，该块内存不再属于你

C程序实现

```c
/*file: inverse.c*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main(int argc, char *argv[])
{
    int i,len;  char *str;
    if (argc!=2) {
        printf("Usage: inverse <string>\n");
        exit(1);
        }
    len=strlen(argv[1]);
    str=(char*)malloc(len+1);
    for(i=0;i<len;i++) str[i]=argv[1][len-i-1];
    str[i]='\0';
    printf("%s\n", str);
    free(str);
    exit(0);
}
```

C++中的字符串处理

- C++保留了C语言的字符串处理机制和相关的处理函数
  - 以’\0’结束字符序列，头文件仍为<string.h>
  - 并提供了完全兼容的<cstring>头文件，相关函数定义在std名字空间
- C++同时提供新的字符串类，封装对字符串的处理
  - 字符串为string类，头文件为<string>，在std名字空间
  - 该类直接支持各种字符串操作

C++中的动态内存分配

- 提供了动态内存管理的运算符
  - new分配内存
  - delete释放内存

new语法格式：

格式1：指针标识符 = new 类型标识符;

格式2：指针标识符 = new 类型标识符(初始化值);

格式3：指针标识符 = new 类型标识符[数组维数];

delete语法格式：

格式1：delete 指针标识符;

格式2：delete[] 指针标识符;

C++程序实现-1

```
1  /*file: inverse.cpp*/
2  #include <iostream.h>
3  #include <string.h>
4  int main(int argc, char *argv[])
5  {
6      int i,len;
7      char *str;
8      if (argc!=2) {
9          cout<<"Usage: inverse <string>\n";
10         return 1;
11     }
12     len=strlen(argv[1]);
13     str=new char[len+1];
14     for(i=0;i<len;i++)
15         str[i]=argv[1][len-i-1];
16     str[i]='\0';
17     cout<<str<<"\n";
18     delete[] str;
19     return 0;
20 }
```

C++程序实现-2

```
1  //file: inverse.cpp
2  #include <iostream>
3  #include <string>
4  using namespace std;
5  int main(int argc, char *argv[])
6  {
7      string str;
8      if (argc!=2) {
9          cout<<"Usage: inverse <string>";
10         return 1;
11     }
12     str=argv[1];
13     ……
14 }
```

新的类型转换操作符

- 标准C++中引入4个强制类型转换运算符
  - static_cast：标准转换及其逆转换
    - void*转换为char*、int转换为float
  - const_cast：转换const或volatile，将转换掉表达式的常量性
  - reinterpret_cast：非标准强制转换
    - 如void*转换为int、double转换为int
  - dynamic_cast：进行类对象间的转换

旧式强制类型转换

语法：

```
// C++强制转换符号
type_name (expr);
// C语言强制转换符号
(type_name) expr;
```

范例：

```
const char *pc = (const char*) pcom;

int ival = (int) 3.14159;

extern char *rewrite_str(char* );//存储类型
char *pc2 = rewrite_str( (char*) pc );

int addr_value = int(&ival );
```

显式类型转换

语法：

```
cast_name<type_name>(expression)
```

范例1：

```
int ival; double dval;
ival += dval;
ival += static_cast<int>( dval );
```

范例2：

```
int x=22, *unsignedPtr;
void *voidPtr=&x;
unsignedPtr= reinterpret_cast<int *>(voidPtr);
```

static_cast和reinterpret_cast的区别

- static_cast（静态转换）执行非多态的转换，用于代替C中通常的强制转换操作。

注意：不能用static_cast把struct转换成int类型或者把double类型转换成指针类型

- reinterpret_cast（重述转换）主要是将数据从一种类型的转换为另一种类型。通常用在函数指针类型之间进行转换，而且使用reinterpret_cast的代码很难移植。

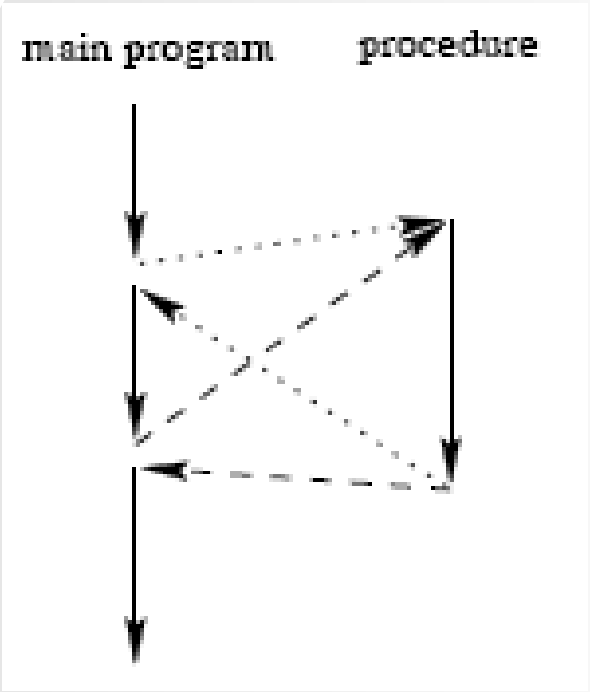
5. 程序设计方法

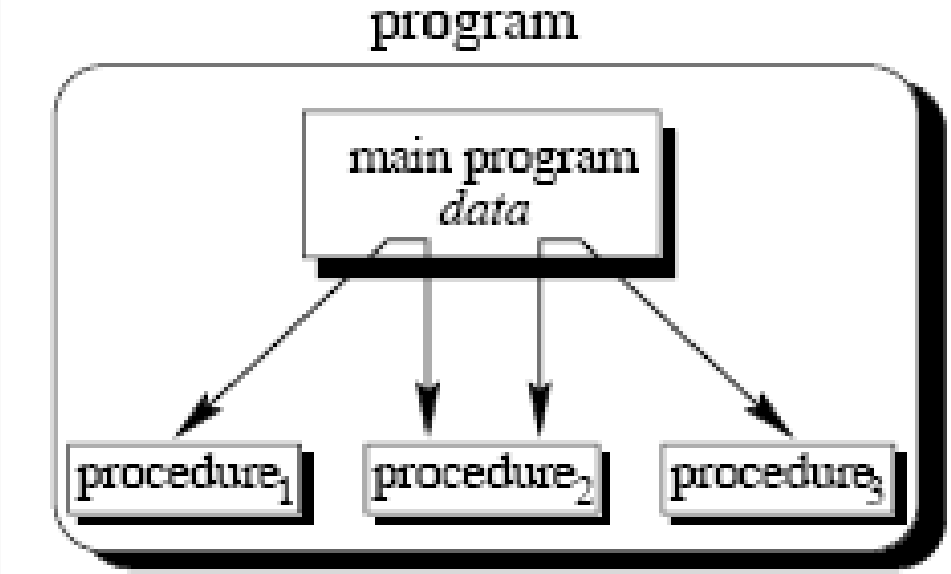
- Unstructured programming（无结构的）
- Procedural programming（面向过程的）
- Modular programming （模块化的）
- Object oriented programming（面向对象的）

5.1 Unstructured programming

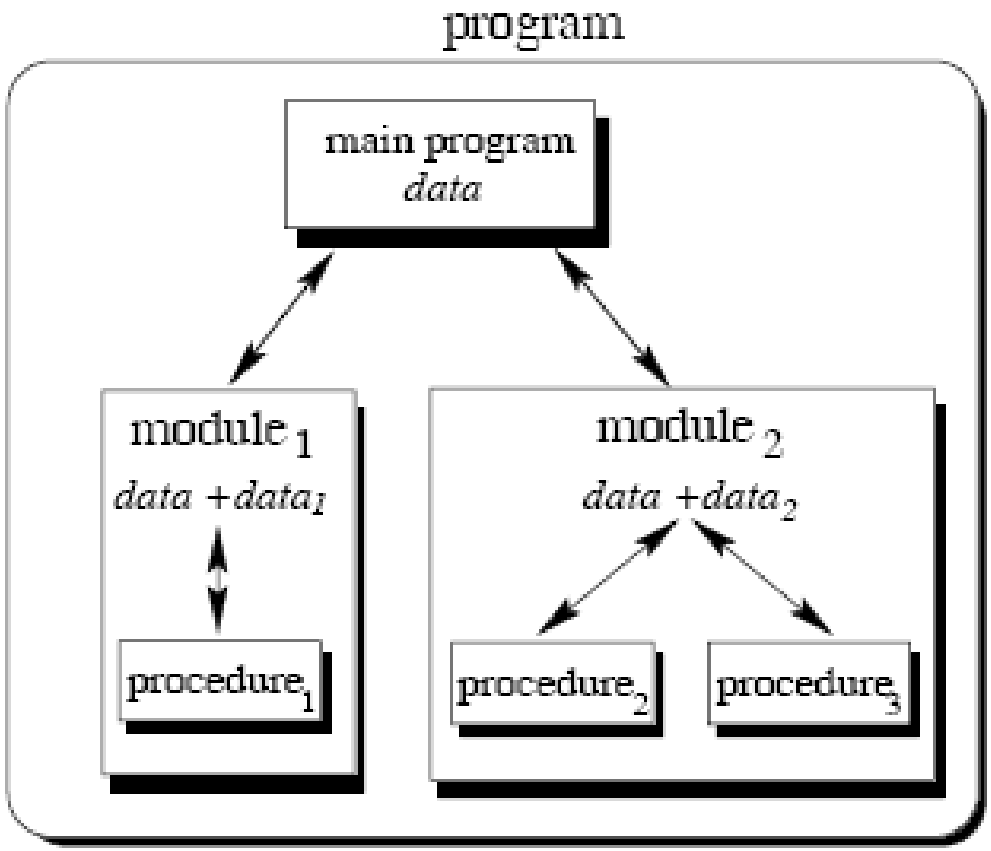
The main program directly operates on global data

5.2 Procedural programming





5.3 Modular programming



续

例：

从键盘输入一个学生的信息（包括姓名、年龄、性别、学号等）和一个老师的信息（包括姓名、年龄、性别、是否授课等），然后将信息输出到屏幕。

续

分析：

根据需求（题目要求），我们可以把问题划分为两个功能模块，一个是**输入模块**，一个是**输出模块**。

我们用C语言来写，参看下面的代码：

```
// ……
int main()                                          // 主函数开始
{
    // 声明用于存储学生信息的变量
    char strStudentName[20];                        // 学生姓名
    int    nStudentAge;                             // 学生年龄
    char cStudentSex;                               // 学生性别
    int    nStudentNumber;                          // 学生学号

    // 声明用于存储老师信息的变量
    char strTeacherName[20];                        // 老师姓名
    int    nTeacherAge;                             // 老师年龄
    char cTeacherSex;                               // 老师性别
    int    nIsTeaching;                             // 是否授课

    // 输入模块
    GetStudentInfo(…);                              // 输入学生信息
    GetTeacherInfo(…);                              // 输入老师信息

    // 输出模块
    PrintStudentInfo(…);                            // 输出学生信息
    PrintTeacherInfo(…);// 输出老师信息

    return(0);
}
```

描述学生的数据

描述老师的数据

函数

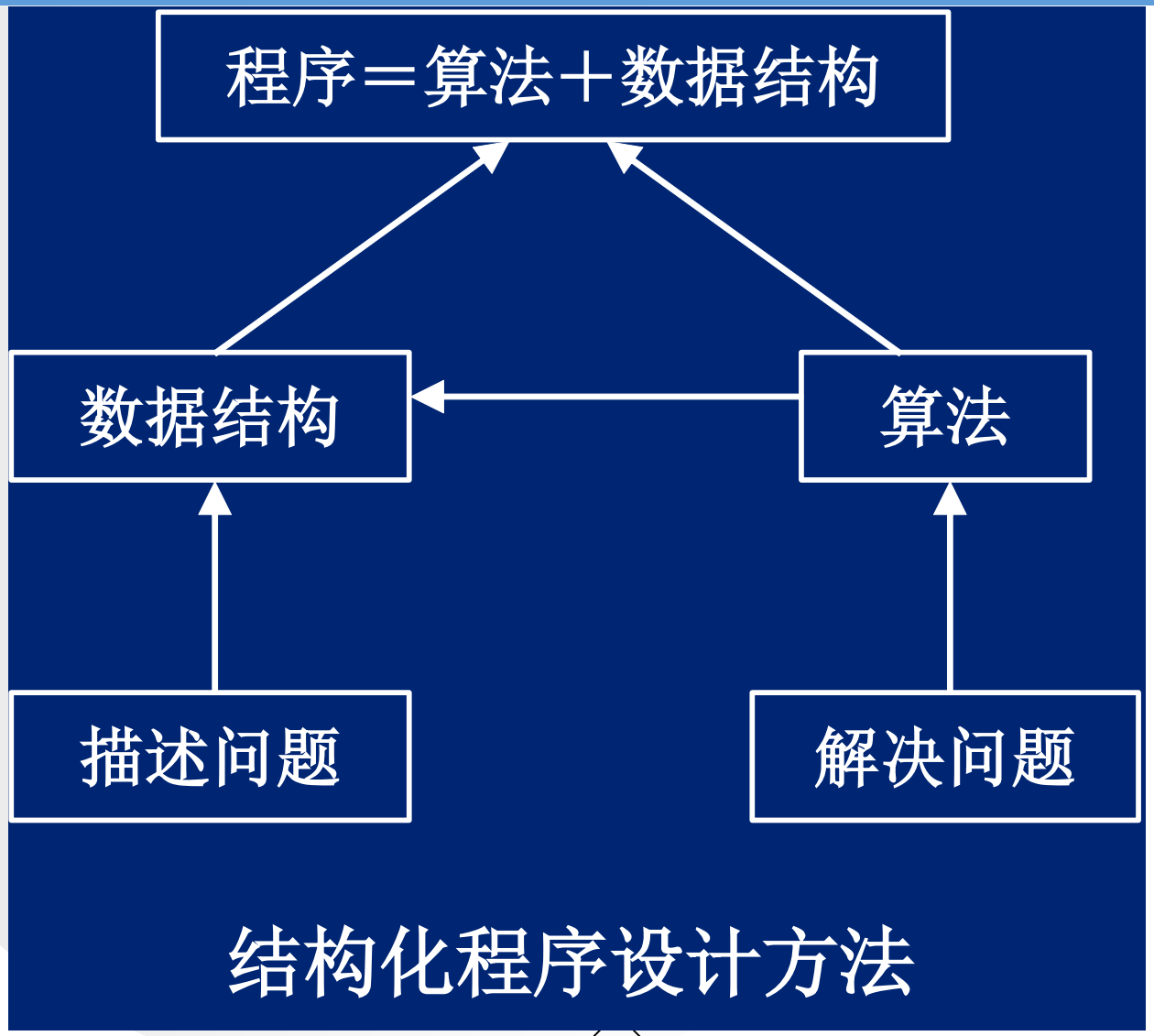
函数

思考：如果需要处理多个学生和教师信息怎么办？

上面的例子中，我们可以进一步将属于学生和老师的变量放入结构中。这样可以在一定程度上完成对数据的封装。但在结构化程序设计中，数据与对其进行操作的函数仍是分离的。

```
// 声明学生结构Student
struct Student
{
    char strStudentName[20];      // 学生姓名
    int  nStudentAge;             // 学生年龄
    char cStudentSex;             // 学生性别
    int  nStudentNumber;          // 学生学号
};
// 声明老师结构Teacher
struct Teacher
{
    char strTeacherName[20];      // 老师姓名
    int  nTeacherAge;             // 老师年龄
    char cTeacherSex;             // 老师性别
    int  nIsTeaching;             // 是否教书
};
```

续



续

问题：

函数用于完成一定的功能，它们都是针对特定的数据进行操作的。那么我们能不能以特定的数据为中心，将数据与对其进行操作的函数封装起来呢？

5.4 Object oriented programming

- 面向对象程序设计出现在80年代中后期
- 面向对象程序设计是建立在结构化程序设计基础上的，但它不再是从功能入手，而是从对象（人、地方、事情等）入手
- 面向对象程序设计以类作为构造程序的基本单位，它具有封装、数据抽象、继承、多态等特点

续

什么是对象？

简单地说，对象就是现实世界中的各种实体，包括人、地点和事物等。例如，桌子、椅子、教室、学生、老师、电话、汽车等等。一般都要从属性和行为两个方面来对它们加以描述。

续

什么是类？

类描述了一组具有相同属性（数据元素）和相同行为（函数）的对象。

类的数据成员是对对象属性的抽象，类的函数成员是对对象行为的抽象，而类本身就是对对象的抽象。

例：C++中类的声明——Student类

```
class Student                                   // Student类的声明
{
public:                                         // 公有成员
        Student();                              // 构造函数
        ~Student();                             // 析构函数

    char*  GetName();                           // 查询姓名
    int    GetAge();                            // 查询年龄
    char   GetSex();                            // 查询姓名
    int    GetNumber();                         // 查询学号

    bool   SetName(char* n);                    // 设置姓名
    bool   SetAge(int age);                     // 设置年龄
    bool   SetSex(char* s);                     // 设置性别
    bool   SetNumber(int num);                  // 设置学号
protected:                                      // 保护成员
    char   m_strName[20];                       // 姓名，字符串数组
    int    m_nAge;                              // 年龄，整型
    char   m_cSex;                              // 性别，字符型
    int    m_nNumber;                           // 学号，整型
};
```

成员函数

成员函数

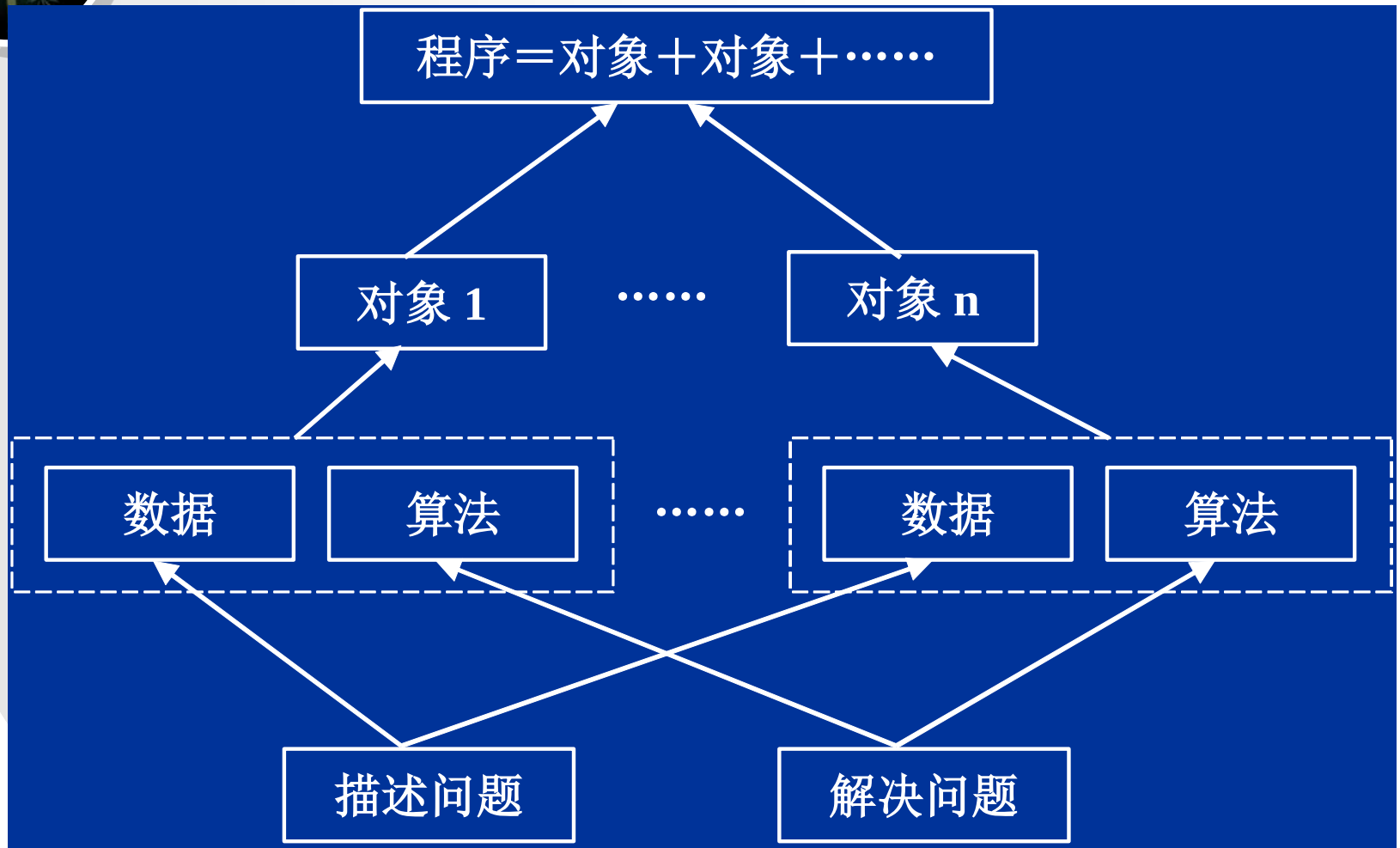
成员变量

续

例：C++中类的使用

```
……
Student A;                          // 声明Student的对象A
A.SetName(“张三”);                  // 设置A的名字
A.SetAge(20);                       // 设置A的年龄
……
```

续



面向对象程序设计方法

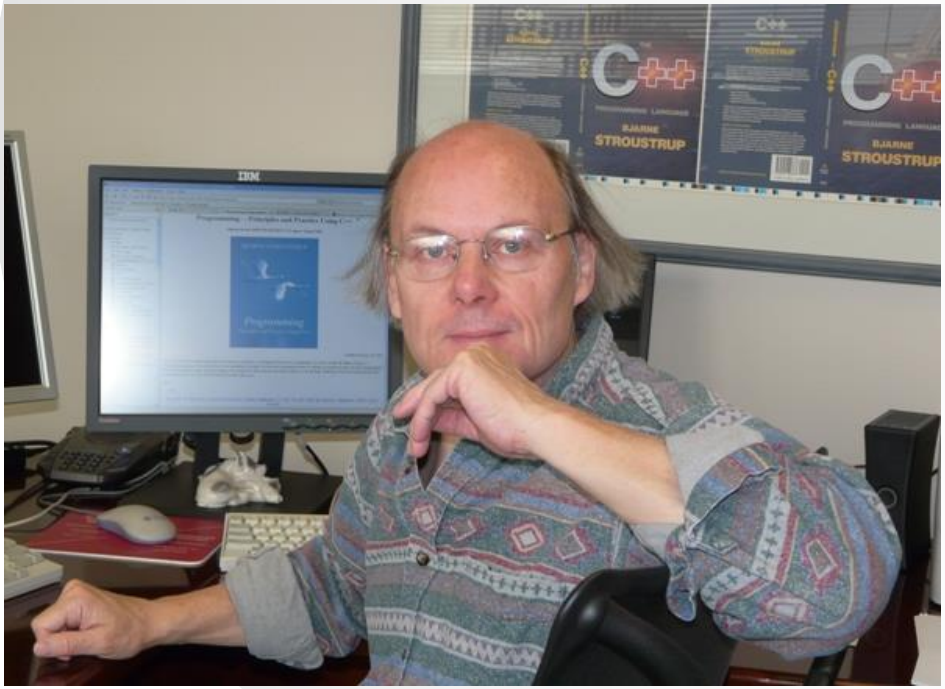
续

- 结构化程序设计方法是一种模块化程序设计方法，它在解决问题时是以功能为中心的，一定的功能模块虽然也作用于特定的数据，但它们并没有被封装在一起。
- 面向对象程序设计方法则是以对象为中心来解决问题的。属于同种对象的属性（数据）和服务（功能）被抽象出来封装到一起。

这是思维方式的一种进步！

Bjarne Stroustrup

- 个人网页:
  - http://www.stroustrup.com/



Thank You !