



第1章 预备知识

——一双明眸：欣赏数据结构之美



计算机学院

主要内容

- 课程介绍
 - ①基本信息 ②考核方式
 - ③教学内容 ④课程目的
- C++核心语法
 - 函数与参数
 - 动态存储分配
 - 类和对象
- 测试与调试



基本信息

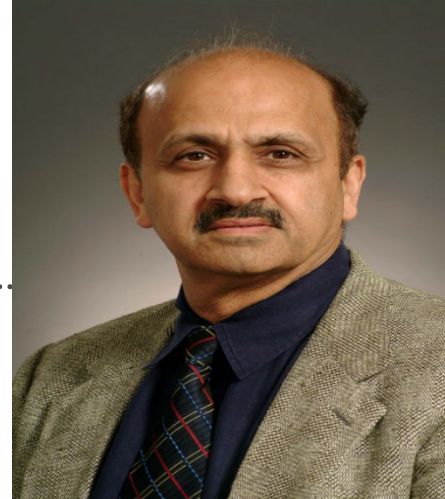
- 课程名称：数据结构
- 额定学时：

讲授51 (3×17) + 实验45 (3×15) = 总计96

- 指定教材：数据结构、算法与应用——C++语言描述（原书第2版）
 - 作者：Sartaj Sahni（萨尼）
 - 译者：王立柱 刘志红
 - 机械工业出版社



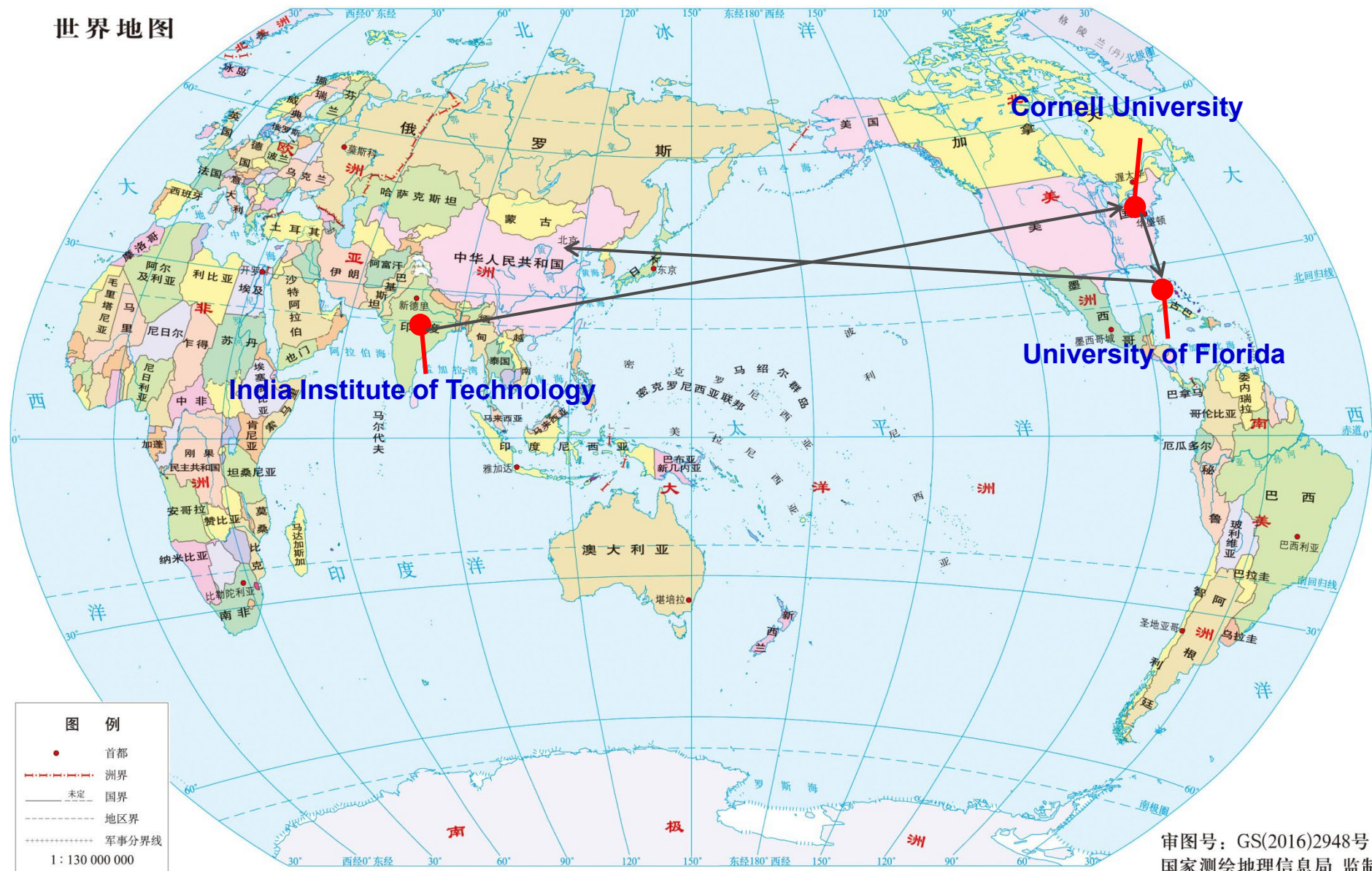
萨尼其人



- Sartaj Sahni
 - <https://www.cise.ufl.edu/~sahni/>
 - Distinguished Professor at the University of Florida
 - Member of the European Academy of Sciences
 - Fellow of IEEE, ACM, AAAS
 - Editor-in-Chief of ACM Computing Surveys



世界地图



计算机学院

如何选择参考书?

- 相关书籍种类繁多

- 南开馆藏871种
- 京东销售57000余种

- 几条小建议

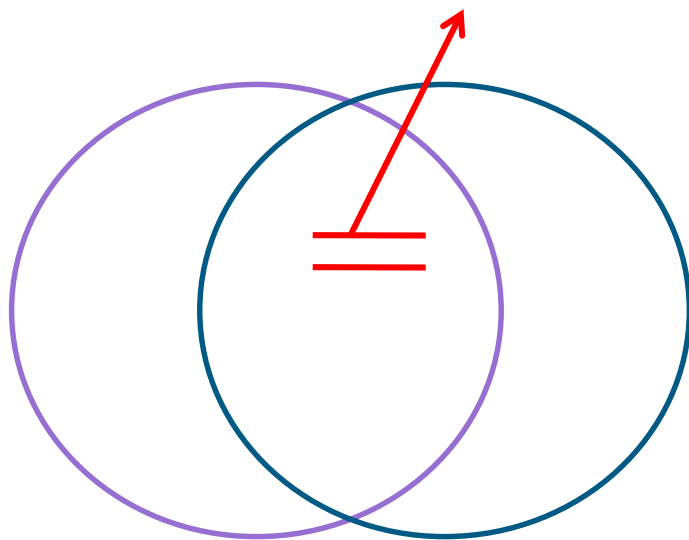
- 最好读一本英文原版书 → 理解更深
- 有空找几本习题集做做 → 益于考试
- 可以自学严蔚敏的教材 → 通俗易懂
- <http://poj.org/>

<input type="checkbox"/>	经济分馆普通书架	12
<input type="checkbox"/>	经济分馆中文库本书架	11
<input type="checkbox"/>	逸夫馆密集书库外文普通书架	10
<input type="checkbox"/>	逸夫馆密集书库外文库本书架	10
<input type="checkbox"/>	逸夫馆新书架	9
<input type="checkbox"/>	周恩来政府管理学院	7
<input type="checkbox"/>	中心馆外文库本书架	3
<input type="checkbox"/>	文中馆周转书库	3
<input type="checkbox"/>	逸夫馆密集书库	3
<input type="checkbox"/>	中心馆南开文库专架	2
<input type="checkbox"/>	金融学院	2
<input type="checkbox"/>	环境科学与工程学院	2



如何使用教材？

交叉部分是重中之重



教材

讲授



考核方式

- 平时成绩占30%
 - 完成4-6次书面作业
 - 完成10次上机作业
- 期末成绩占70%



课程内容

- 预备知识

- 5%

- 程序性能分析、**调试测试技术（不考）**

- 数据结构

- 70%

- 表（1维）、树（2维）、图（N维）

- 算法设计

- 25%

- 查找、排序、... ..

相互融合



地位和作用

- 数据结构+算法：是编写程序的核心和基础
- 下里巴人的追求
 - 研究“数据”的存储、表达、操作方式
 - 学习经典的数据结构
 - 学习经典数据结构在经典问题中的应用
- 阳春白雪的冀望
 - 将经典数据结构巧妙改进以适应新问题
 - 发明新的具有普适意义的数据结构



课程目的

- 学习基本数据结构，
其上的操作，
掌握其实现方法，
以及如何利用它们**解决实际问题**
- 学习一些经典的算法设计方法及其应用，具备基本的算法设计能力



课程目的？

- 教大家如何编写程序？
- 不是学过C++了吗？已经会写程序了
- 问题是：什么叫“会”写程序？
- 另一个相似的问题：什么叫“会”下象棋？
- 知道“马走日”、“象走田”、“炮打隔一位”、…就是“会”下象棋吗？



课程目的？

- 下象棋的目的是什么？——**不是不违反规则，而是击败对手！**
- 因此，一般意义的“会”下象棋，至少是
 - 了解一些布局的方法
 - 知道一些中局攻防的基本方法
 - 掌握一些残局的下法
 - 总之，系统地学习过一些如何“击败对手”的方法



课程目的？

- 同样，我们编写程序的目的是什么？——**不是按照语法规则堆砌代码，而是要解决实际问题**
- 因此，“会写程序” 应该是，对于一个要解决的实际问题
 - 利用一些学过的知识和经验，建立数学模型
 - 抽象出要处理的数据，设计数据结构解决数据如何在计算机中保存
 - 设计算法，能对数据进行处理得到期望的结果



例子—搜索（search：查找）

- 已有一组数据，在其中找到指定数据
- 简单方法：数据无序存储，顺序搜索
 - 如，数据集合为：26 33 35 29 19 12 22
 - 搜索26，1次比较操作
 - 29，4次
 - 22，7次
 - 与列表长度 n 成比例



例子——搜索

- 更好的方式：数据按大小次序存储，二分搜索方式

- 12 19 22 26 29 33 35

- 搜索22

- 与中心元素比较， <26 ——继续搜索前半部分
- 同样与中心元比较， >19 ——继续搜索后半部分
- 与22比较，相等，成功！3次比较！

- 每次比较搜索范围减小一半

- 充分体现了数据结构和算法的紧密关系



主要内容

- 课程介绍
- C++核心语法（以实例为主）
 - ①传值过程 ②拷贝构造函数
 - ③递归函数
- 测试与调试



H1. 传值过程

调用int的copy constructor将实参值拷贝给形参，即借用了实参的副本

```
void main()
{
    int i=Abc(1,2,3);
    cout<<i<<endl;
}
```

```
int Abc(int a, int b, int c)
{
    return a+b+b*c+(a+b-c)/(a+b)+4;
}
```

计算完成，释放副本

```
class person
{
    char* pName;
    ....
}
```

```
void main()
{
    person p1("zhang");
    person p2=p1;
    print(p1);
}
```

```
void print(person p)
{
    cout<<p.pName<<endl;
}
```



传值过程 (cont.)

- 复制构造函数

- 也叫拷贝构造函数, copy constructor
- 当: 用已存在的对象来创建一个新对象,
- 或: 对象作为传值参数,
- 或: 对象作为返回值 时, 触发复制构造函数
- 隐式的复制构造函数**仅提供浅拷贝**
- 显式的复制构造函数**可提供深拷贝**
- 当类包含指针成员时一般应自定义复制构造函数



传值过程 (cont.)

```
class person
{
    char* pName;
public:
    person(char* pN)
    {
        pName=new char[strlen(pN)+1];
        strcpy(pName,pN);
    }
    person(person &p)
    {
        pName=new char[strlen(p.pName)+1];
        strcpy(pName, p.pName);
    }
    ~person(){
        delete pName;
    }
}
```

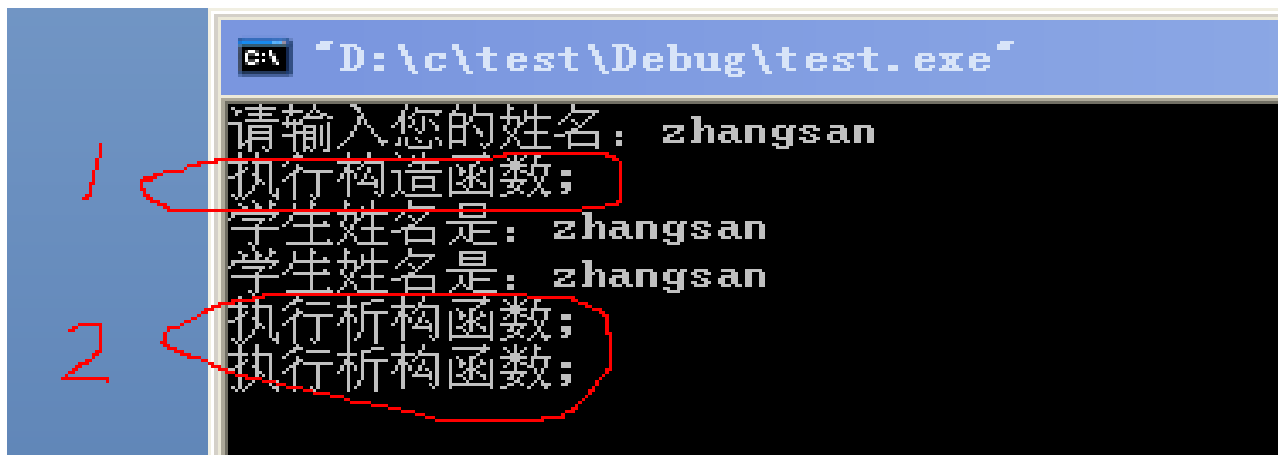


拷贝构造函数

- 疑问
 - 设自定义类MyClass含有指针类型的数据成员，obj_A是MyClass的一个对象，如果程序中有语句MyClass obj_B=obj_A；则执行时会出现错误，为什么？如何解决？
- 示例：Program_1



拷贝构造函数

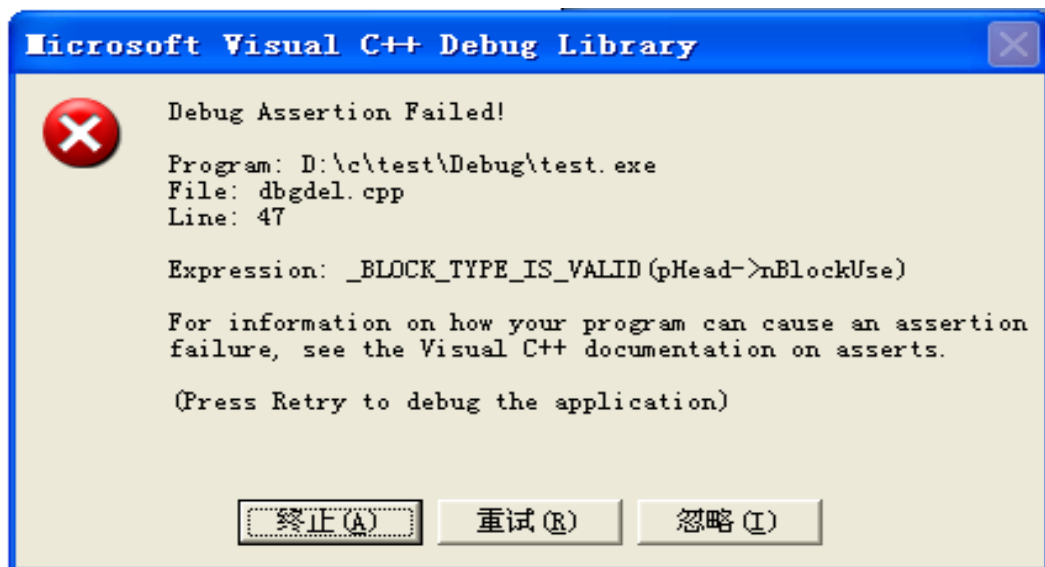


```
C:\ "D:\c\test\Debug\test.exe"
请输入您的姓名: zhangsan
1 执行构造函数;
   学生姓名是: zhangsan
   学生姓名是: zhangsan
2 执行析构函数;
   执行析构函数;
```

- 错误情形（浅拷贝）

- 由红色圆圈可见：对象obj_A和obj_B各执行了一次析构函数，这是正常的；但是**类的构造函数只被执行了一次！**这是因为，创建obj_A的时候执行了构造函数，而在创建obj_B的时候执行的是隐藏的拷贝构造函数。

拷贝构造函数



- 程序虽然可以正常编译和执行，但是最后会跳出上面的错误提示。这是由于obj_A和obj_B的指针成员name指向了内存中的同一片区域，第一次执行析构函数时将其释放了，第二次执行析构函数还想再释放一次，显然会发生错误。

拷贝构造函数

- 示例：Program_2

```
C:\ "D:\c\test\Debug\test.exe"
请输入您的姓名: zhangsan
执行构造函数;
学生姓名是: zhangsan
学生姓名是: zhangsan
执行析构函数;
执行析构函数;
```

```
C:\ "D:\c\test\Debug\test.exe"
请输入您的姓名: zhangsan
执行构造函数;
执行拷贝构造函数;
学生姓名是: zhangsan
学生姓名是: zhangsan
执行析构函数;
执行析构函数;
Press any key to continue
```

- 自定义拷贝构造函数（深拷贝）

- 新的程序执行结果显示obj_A执行了构造函数，obj_B执行了自定义的拷贝构造函数，并且不再发生错误。



拷贝构造函数

- 基本认识

- 拷贝构造函数是一种特殊的构造函数，具有一般构造函数的特性。只含有一个形参，且为本类对象的引用。拷贝构造函数的原型为：

<类名> (<类名> &);

- 作用是使用一个已存在的对象去初始化另一个正在创建的对象。
- 当类中含有指针类型的数据成员时，一般都应该自定义一个拷贝构造函数。



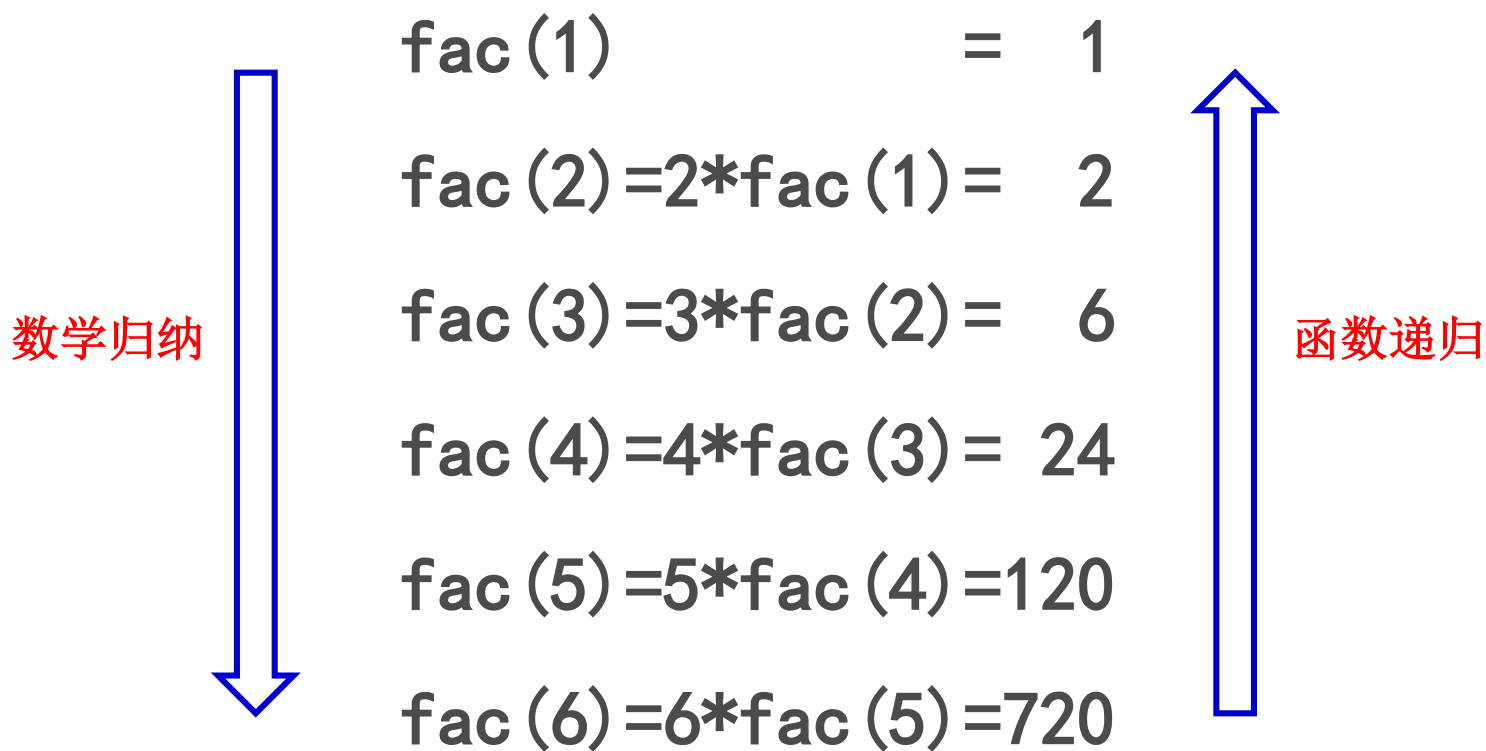
H2. 递归函数

- **直接递归**
 - 函数F的代码中直接包含了调用F的语句
- **间接递归**
 - 在函数F、G、H、... 之间形成调用回路



递归函数 VS 数学归纳

- 计算阶乘 $n!$



递归函数示例：反序输出

```
void inv(int n) {  
    int i; cin>>i;  
    if (n>1) {inv (n-1) ;}  
    cout<<i<< " " ;  
}
```

```
void main() {  
    inv(10) ;  
}
```

Input:

1 2 3 4 5 6 7 8 9 10

Output:

10 9 8 7 6 5 4 3 2 1



主要内容

- 课程介绍
- C++核心语法
- 测试与调试
 - ①测试基本知识
 - ②设计测试用例



程序的属性

- 基本属性

- 正确性：首要的和必备的属性
- 确定性
- 有穷性：程序终止性证明和检验

- 扩展属性

- 鲁棒性
- 通用性
- 易读易修改
- 有效性：程序效率分析（下次课）



程序正确性检验方法

- 静态方法
 - 符号执行
 - 定理证明
 - 模型检测
- 动态方法
 - 黑盒测试
 - 白盒测试



程序测试

- 关于测试最重要的一句话
 - 测试的目的是发现尽可能多的错误，而非证明程序正确
- 基本认识
 - 所谓测试就是进行对比，将基于一组实际数据的程序执行结果与理想结果比较
 - 如果不一致，则发现错误
 - 如果一致，只说明没有发现错误，而非程序正确
 - 很难实现穷举测试



软件测试的理论体系

- 从流程上分
 - 单元测试、集成测试、确认测试、系统测试
- 从技术上分
 - 黑盒测试、白盒测试、灰盒测试
- 从管理上分
 - 测试计划、测试人员、测试配置、测试文档、测试工具
- 从目的上分
 - 功能测试、性能测试、安全测试



测试的流程（简化）



设计测试用例

- 等价类划分
- 语句覆盖
- 分支覆盖
- 从句覆盖（一般）
- 从句覆盖（加强）
- 执行路径覆盖
- 边界值
- 经验推测



二次方程求解例

$$ax^2 + bx + c$$

$$\text{根: } \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

$$d = b^2 - 4ac$$

$$d = 0 \quad \text{两个根相等}$$

$$d > 0 \quad \text{两个不同实数根}$$

$$d < 0 \quad \text{两个虚根 } \frac{-b \pm \sqrt{-d}i}{2a}$$



求二次方程根的函数

```
template<class T>
```

```
void OutputRoots(T a, T b, T c)
```

```
{// Compute and output the roots of the quadratic.
```

```
    T d = b*b-4*a*c;
```

```
    if (d > 0) {// two real roots
```

```
        float sqrtd = sqrt(d);
```

```
        cout << "There are two real roots "
```

```
            << (-b+sqrtd)/(2*a) << " and "
```

```
            << (-b-sqrtd)/(2*a)
```

```
            << endl;}
```



求二次方程根的函数

```
else if (d == 0)
```

```
    // both roots are the same
```

```
    cout << "There is only one distinct root "
```

```
        << -b/(2*a)
```

```
        << endl;
```

```
else // complex conjugate roots
```

```
    cout << "The roots are complex"
```

```
        << endl
```

```
        << "The real part is "
```

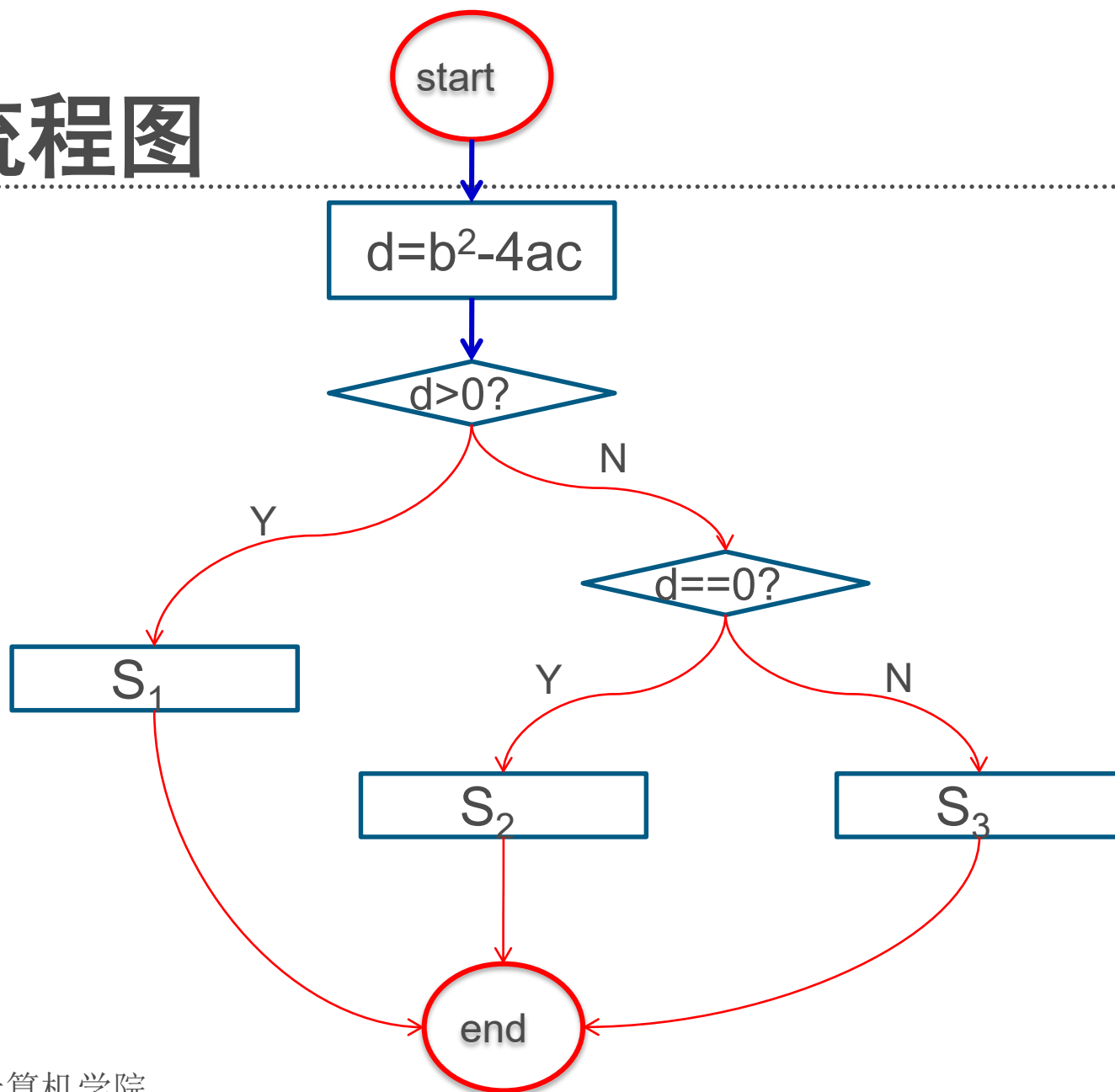
```
        << -b/(2*a) << endl
```

```
        << "The imaginary part is "
```

```
        << sqrt(-d)/(2*a) << endl; }
```



流程图



等价类划分

- 输入数据、输出数据划分若干类
- 不同类数据→程序行为应表现出本质差别
同类数据→程序表现出本质类似的行为
- 测试数据集：每类至少抽取一个数据
- 上例：产生复数根、产生相同实数根、产生不同实数根



语句覆盖

- 用例使程序的每一条语句都至少执行一次
- 是设计测试用例最基本的要求
- 如果做不到，则说明
 - 用例设计不达标
 - 或 程序本身存在不可达路径



分支覆盖

- 要求测试集要能使每一个条件都分别出现 true 和 false 两种情况
- 也就是说每一个条件分支都至少执行一次



一般从句覆盖

- 如果条件包含从句，如

`if ((C1 && C2) || (C3 && C4))`

- 要求每个条件中的每个从句均出现了true和false两种情况
- 则满足一般从句覆盖的测试用例集是

`(T, T, T, T)`

`(F, F, F, F)`



加强从句覆盖

- 如果条件包含从句，如

`if ((C1 && C2) || (C3 && C4))`

- 要求每个条件中的各种从句取值组合均出现
- 则满足加强从句覆盖的测试用例数量显然是
 $2*2*2*2=16$ 个



执行路径覆盖

- 顾名思义，要求测试用例可以覆盖程序所有可能的执行路径
- 所以关键是列举执行路径



测试用例设计原则

- 至少语句覆盖
- 尽量路径覆盖
- 最好从句覆盖



例子：求最大元

```
template<class type>
```

```
int Max(type a[], int n)
```

```
{// Locate the largest element in a[0:n-1].
```

```
    int pos = 0;
```

```
    for (int i = 1; i < n; i++)
```

```
        if (a[pos] < a[i])
```

```
            pos = i;
```

```
    return pos;
```

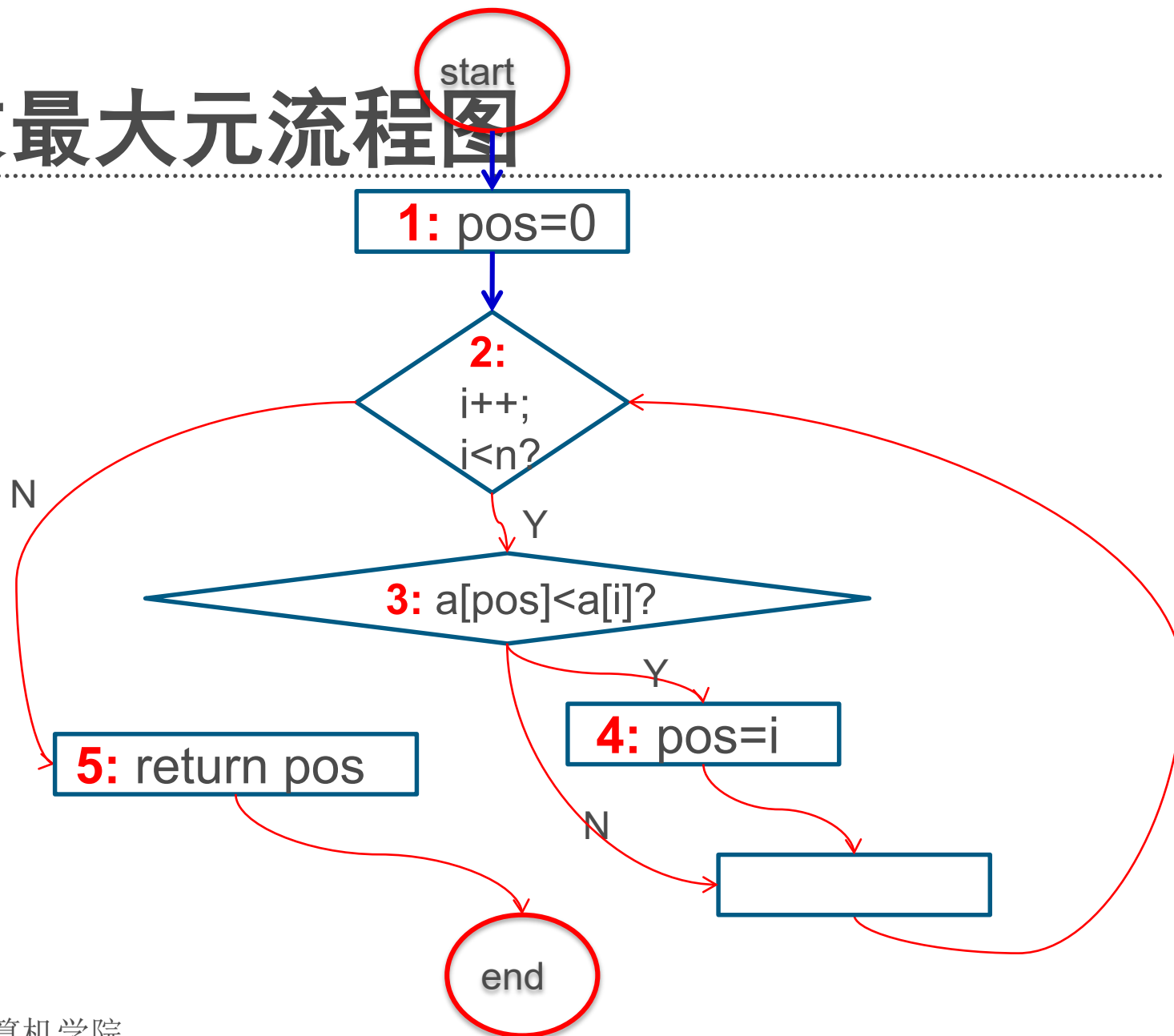
```
}
```

$a[0:4]=[2, 4, 6, 8, 9]$ ：语句覆盖，但没有分支覆盖

$a[0:4]=[4, 2, 6, 8, 9]$ ：语句覆盖，且分支覆盖



求最大元流程图



思考

- P_{34} 例1-7最后的测试集 $(1, -5, 6)$ $(1, -8, 16)$
 $(1, 2, 5)$ 是最好的一组用例吗？为什么？
 - 对于漏写a的错误无法识别；
 - 缺少边界值检查。



小结

- 回顾了两个C++知识
 - 拷贝构造函数
 - 递归函数
- 了解了测试的基本流程，尤其是测试用例的设计方法



思考

- 设计一个梭哈游戏的子模块，判断任意5张牌是否构成“同花顺”？
 - 合理运用语法知识写出C++源代码
 - 设计测试用例以检验代码的正确性
 - 使用等价类划分的方法
 - 请说明这组用例是否满足以下准则
 - 语句覆盖
 - 分支覆盖
 - 从句覆盖
 - 执行路径覆盖



深入思考

- 你能设法实现一款自动检测和判分程序吗，要求能检验任何人用C++写的“同花顺”程序？有哪些难点？
- 你能用最简洁的形式完成一个人机对战的梭哈游戏吗？有哪些关键之处？



本章结束

