第一章 绪论

——刘亮亮

上海对外经贸大学

SHANGHAI UNIVERSITY OF INTERNATIONAL BUSINESS AND ECONOMICS

主要内容

- 什么是数据结构
- 基本概念和术语
- 抽象数据类型的表示与实现
- 算法的概念与性质
- 算法分析
- 小结

主要内容

- 什么是数据结构
- 基本概念和术语
- 抽象数据类型的表示与实现
- 算法的概念与性质
- 算法分析
- 小结

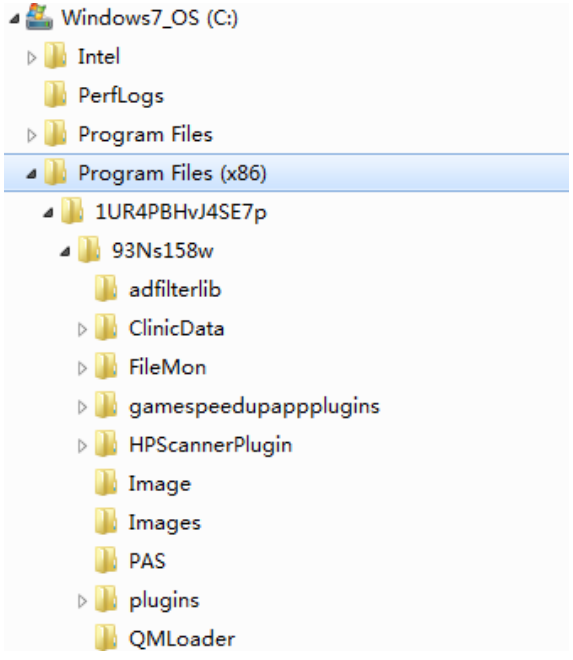
什么是数据结构

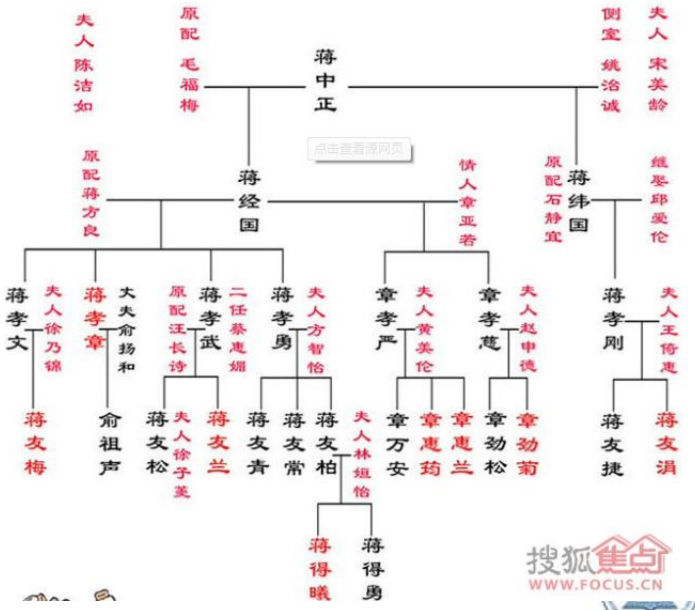
- **示例1:电话号码查询系统**
  - 电话号码簿，记录了N个人的名字和其电话号码。形如（a_1,b_1）、（a_2,b_2）、...(a_n,b_n)。——一种简单的线性关系（一对一）。
  - 其它示例：图书查询系统中的图书数据结构等

姓名	电话号码
张三	13612345588
李四	13562300118
……	……

什么是数据结构

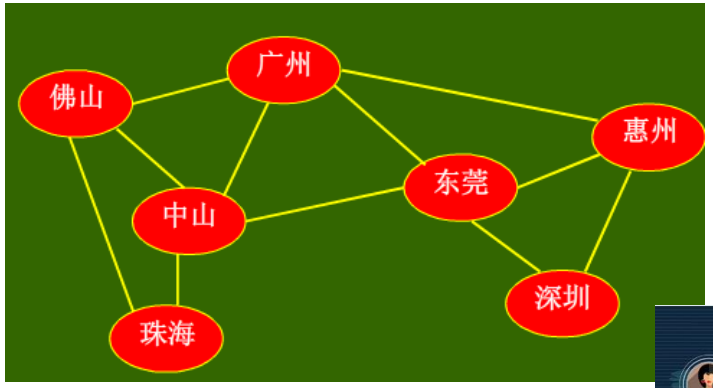
- **示例2:磁盘目录文件系统**
  - 磁盘目录：磁盘根目录下有很多子目录及文件，每个子目录下面有很多的子目录等，但是，每个子目录都只有一个父目录。——树型结构（一对多）
  - 其它示例：行政关系、家谱等

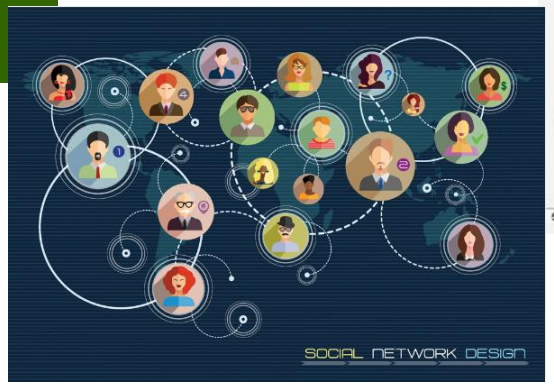




什么是数据结构

- **示例3:交通图**
  - 从一个地方到另一个地方有很多条路径——网状结构（多对多的关系）
  - 其它示例：社交网络、快递网点、京东网点等





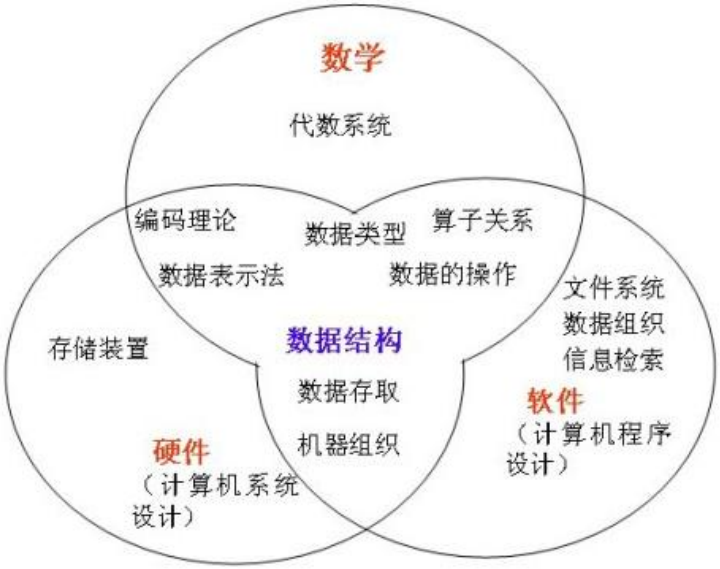
什么是数据结构

- **结论：**
  - 以上例子无法用数学公式或方程来描述。
  - 数据之间是存在着某种特定的关系的.
  - 数据结构在日常生活中无处不在！

- **问题：**
  - 什么是数据结构?
  - 数据结构是干什么的?

什么是数据结构

- 数据结构是一门研究**非数值计算**的程序设计问题中计算机的操作对象以及它们之间的**关系和操作**等的学科。
- 数据结构是介于**数学、计算机硬件**和**计算机软件**三者之间的一门核心课程。



什么是数据结构

- **数据结构讨论的范畴**
  - Niklaus Wirth提出：

    Algorithm+Data Structures=Programs

    ✓程序设计：为计算机处理问题编制的一组指令集

    ✓算法：处理问题的策略

    ✓数据结构：问题的数学模型

  - 数据结构研究的范围：数据的逻辑结构、数据的存储结构（物理结构）、以及这种结构定义相关的运算（或操作）、设计并实现相应的算法、分析算法效率。

主要内容

- 什么是数据结构
- 基本概念和术语
- 抽象数据类型的表示与实现
- 算法的概念与性质
- 算法分析
- 小结

主要内容

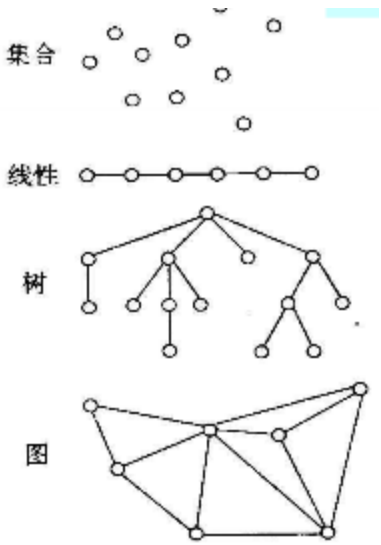
- 什么是数据结构
- 基本概念和术语
- 抽象数据类型的表示与实现
- 算法的概念与性质
- 算法分析
- 小结

基本概念和术语

- **数据**
  - 客观事物的符号表示，所有能输入到计算机中被被计算机程序处理的符号总称。
  - 例如：字符串、图像、声音等
- **数据元素**
  - 数据的基本单位，在计算机程序中作为一个整体进行操作和处理。
  - 例如：树或图中的一个结点等
  - 由若干个**数据项**组成。
    - ✓例如：（数据结构，严蔚敏，...）中的每一项称为数据项
- **数据对象**
  - 性质相同元素的集合，数据的子集；
  - 例如：Z={0,1,2,...}

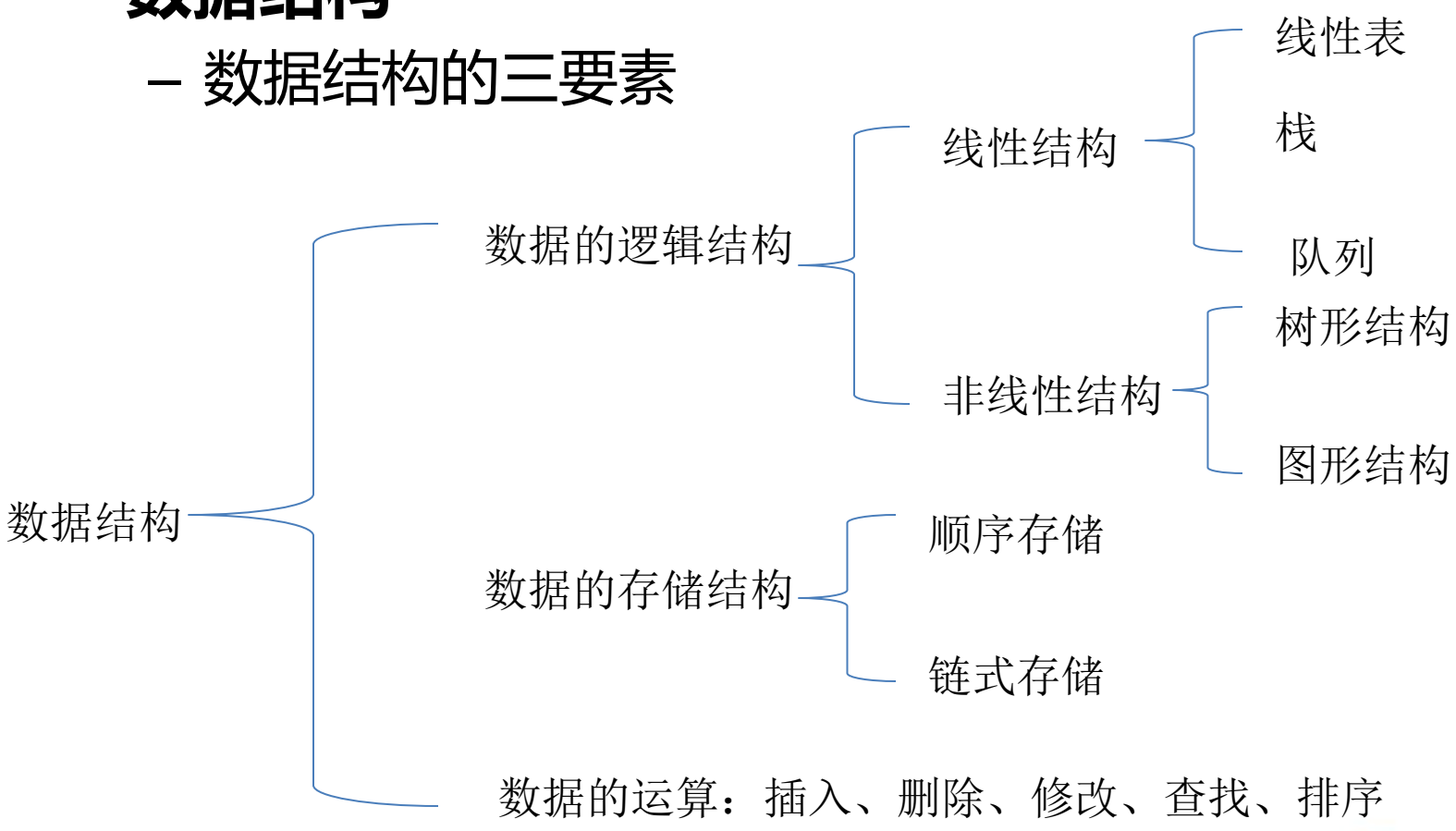
基本概念和术语

- **数据结构**
  - 相互之间存在一种或多种特定关系的数据元素的集合。
  - 包括数据元素和关系，数据元素之间的关系也称为结构。
  - 数据结构+算法=程序
  - 四种基本结构：
    - ✓集合：同属于一个集合
    - ✓线性结构：一对一的关系
    - ✓树形结构：一对多的关系
    - ✓图状结构（或网状结构）：多对多的关系
  - 形式化定义：*Data_Structure=(D, S)*
    - ✓ *D：元素的有限集*
    - ✓ *S：关系的有限集*



基本概念和术语

- **数据结构**
  - 数据结构的三要素



基本概念和术语

- **数据结构示例：**
  - 示例1：复数
    - ✓Complex=(C,R)
      - C={c_1,c_2}
      - R={<c_1,c_2>}
  - 示例2：课题小组
    - ✓Group={P,R}
      - P={T,G_1,…,G_n,S_{11}….S_{nm},1<=n<=3,1<=m<=2}
      - R={R_1,R_2}
        - R_1={<T,G_i>|1<=i<=n,1<=n<=3}
        - R_2={<G_i,S_{ij}>| 1<=i<=n,1<=j<=m,1<=n<=3,1<=m<=2}

基本概念和术语

- **逻辑结构**
  - 定义中的“关系”描述数据元素之间的逻辑关系
- **存储结构（数据的物理结构）**
  - 数据结构在计算机中的表示（映像）
  - 两种存储结构：
    - ✓**顺序存储结构**:存储器中的相对位置来表示数据元素之间的逻辑关系；
    - ✓**链式存储结构**：利用存储地址的指针来表示元素之间的逻辑关系；

基本概念和术语

- **数据类型**
  - 一个值的集合和定义在这个集合上的一组操作的总称。
  - 两种类型
    - ✓原子类型
    - ✓结构类型
- **抽象数据类型**
  - 一个数学模型以及定义在该模型上的一组操作。
  - 三种类型：
    - ✓原子类型
    - ✓固定聚合类型
    - ✓可变聚合类型

基本概念和术语

- **抽象数据类型的形式定义：**
  - (D,S,P)
    - ✓ D是数据对象
    - ✓ S是D上的关系集
    - ✓ P是对D的基本操作集合
  - 定义描述

    ADT 抽象数据类型名称{
    数据对象：<数据对象的定义>
    数据关系：<数据关系的定义>
    基本操作：<基本操作的定义>
    }
    基本操作名(参数表)
    初始条件：<初始条件描述>
    操作结果：<操作结果描述>

基本概念和术语

- **抽象数据类型定义示例**
  - 例1　三元组的定义

ADT Triplet{
　　数据对象：D={e1,e2,e3|e1,e2,e3∈ElemSet}
　　数据关系：R1={<e1,e2>,<e2,e3>}
　　基本操作：
　　Init(&T,v1,v2,v3)
　　操作结果：构造了三元组
　　Destroy(&T)
　　操作结果：三元组T被销毁
　　Get(T,i,&e)
　　初始条件：三元组T已存在，1<=i<=3
　　操作结果：用e返回T的第i元的值
　　……
}ADT Triplet

主要内容

- 什么是数据结构
- 基本概念和术语
- 抽象数据类型的表示与实现
- 算法的概念与性质
- 算法分析
- 小结

主要内容

- 什么是数据结构
- 基本概念和术语
- 抽象数据类型的表示与实现
- 算法的概念与性质
- 算法分析
- 小结

抽象数据类型的表示与实现

- **利用类C语言描述：介于伪代码和C语言的描述语言。**
- **也可以采用伪代码描述算法**
- **描述语法：**
  - （1）预定义常量和类型
    - ✓#define TRUE 1
  - （2）类型定义typedef
    - ✓typedef int Status;
  - （3）函数定义
    - ✓函数类型 函数名(函数参数名){
      //算法说明
      语句序列
      }//函数名

抽象数据类型的表示与实现

- （4）赋值语句
  - ✓变量名=表达式;
  - ✓变量1=变量2=…=变量n=表达式;
  - ✓例如：i=3;
- （5）选择语句
  - ✓条件语句1: if(表达式)语句;
  - ✓条件语句2: if(表达式)语句;
    else 语句;
  - ✓switch语句
- （6）循环语句
  - ✓for(赋值语句；条件；修改表达式)语句;
  - ✓while(条件)语句;
  - ✓do{}whlie(条件);
- 其余参考教材1.3节

抽象数据类型的表示与实现

- **示例：Triplet的表示与实现**

```
//采用动态分配的顺序存储结构
Typedef ElemType *Triplet;
Status InitTriplet(Triplet &T, ElemType v1,ElemType v2,ElemType v3);
……
//基本操作的实现
Status InitTriplet(Triplet &T, ElemType v1, ElemType v3, ElemType v3){
        T=(ElemType*)malloc(3*sizeof(ElemType));
        if(!T)exit(OVERFLOW);
        T[0]=v1;
        T[1]=v2;
        T[2]=v3;
        return OK;
}
```

主要内容

- 什么是数据结构
- 基本概念和术语
- 抽象数据类型的表示与实现
- 算法的概念与性质
- 算法分析
- 小结

主要内容

- 什么是数据结构
- 基本概念和术语
- 抽象数据类型的表示与实现
- 算法的概念与性质
- 算法分析
- 小结

什么是算法?

- 示例：程序1+…+100

```
int i=0;
int sum=0;
int n=100;
for(i=1;i<=n;i++)
{
    sum=sum+i;
}
printf( “%d” );
```

这段简单的程序就是一个算法!
问题:真的很好吗？高效吗?

看看少年高斯的解法

```
int i=1;
int n=100;
int sum=(i+n)*n/2;
```

算法

- **定义：对特定问题求解步骤的一种描述，是指令的有限序列。**
- **性质：**
  - 有穷性
    - ✓执行有穷步结束、每一步在有穷的时间内完成。
  - 确定性
    - ✓唯一的一条执行路径，相同输入只能有相同的输出。
  - 可行性
    - ✓所有的操作可以通过基本操作执行有限次实现。
  - 输入
    - ✓有0个或多个的输入
  - 输出
    - ✓有一个或多个输出

算法

- **算法设计的要求**
  - 正确性
  - 可读性
  - 健壮性
  - 效率与低存储量需求

主要内容

- 什么是数据结构
- 基本概念和术语
- 抽象数据类型的表示与实现
- 算法的概念与性质
- 算法分析
- 小结

主要内容

- 什么是数据结构
- 基本概念和术语
- 抽象数据类型的表示与实现
- 算法的概念与性质
- 算法分析
- 小结

算法分析

- **算法分析：对算法质量优劣的评价，被分析的算法必须是正确的。**
- **除正确性外，应从三方面分析一个算法：**
  - 依据算法编写的程序在计算机中运行时间多少的度量，称之为**时间复杂度**。
  - 依据算法编写的程序在计算机中占存储空间多少的度量，称之为**空间复杂度**。
  - 其它方面：如算法的可读性、可移植性等。

算法分析

- **算法的度量**
  - 时间复杂度
    - ✓算法中基本操作的重复执行的次数是问题规模n的某个函数f(n),算法的时间量度记作

      T(n)=O(f(n))
    - ✓随着问题规模n的增大，时间增长率和f(n)增长率相同。
    - ✓示例1：for(int i=0;i<n;i++){++x;s+=x;}
      - 执行的次数：3n，时间复杂度为O（n）
    - ✓示例2：for(int i=0;i<n;i++)

      for(int j=0;j<n;j++){++x};
      - 执行的次数：$n+2n^2$，时间复杂度为$O(n^2)$
  - 时间复杂度T(n)按数量级递增顺序：
    - ✓$O(1)<O(\log_2 n)<O(n)<O(n\log_2 n)<O(n^2)<O(n^3)<\ldots<O(n^k)<O(2^n)$

算法分析

- **关于“O”符号的数学定义：**
  - f(n)=O(g(n))表示存在一个正常的常数c和n_0，使得对所有的n>=n_0，有f(n)<=c*g(n)
- **大写O符号给出了函数f的一个上限。**
- **定理：**
  - 如果f(n)=$a_m n^m$+...+a_1n+a0，且a_m>0,则f(n)=O(n^m)

当n→∞时，有f(n)/g(n)=常数≠0，
则称函数f(n)与g(n)同阶，或者说，f(n)与g(n)同一个数量级，记作

$$f(n)=O(g(n))$$

称上式为算法的时间复杂度，或称该算法的时间复杂度为O(g(n))。
其中，n为问题的规模(大小)的量度。

算法分析

- **示例1：**

```
1)  ++x;  s=0;
2)  for（i=1;  i<=n;  ++i）
      {++x;  s+=x;  }
3)  for（j=1;  j<=n;  ++j）
     for（k=1;  k<=n;  ++k）
        {++x;  s+=x;  }
```

基本操作：x增1

总的执行次数为：$1，n，n^2$

时间复杂度：$O（1），O（n），O（n^2）$

算法分析

- **示例1：n*n矩阵相乘的时间复杂度**

```
for (i=1; i<=n; ++j)                          (1) n次
    for (j=1; j<=n; ++j) {                    (2) n²次
        c[i][j]=0;                            (3) n²次
        for (k=1; k<=n; ++k)                  (4) n³次
            c[i][j]+=a[i][k]*b[k][j]; (5) n³次
    }
```

基本操作：乘法

总的执行次数为：$2n^3+2n^2+n$

时间复杂度：与乘法重复执行的次数n^3成正比

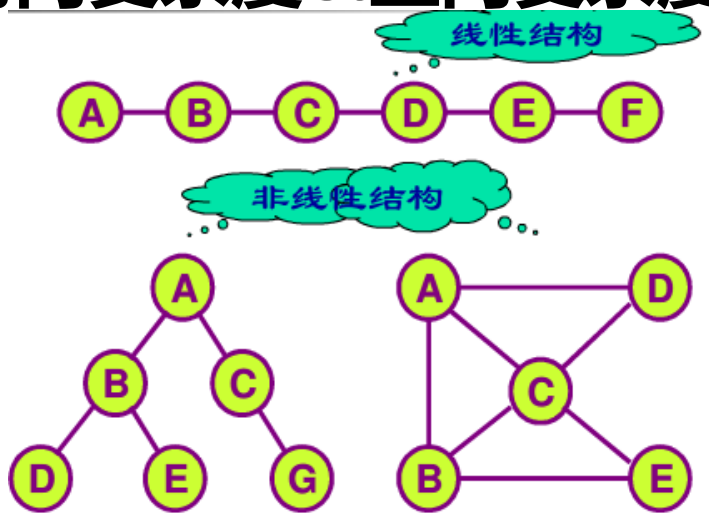
$$T(n)=O(n^3)$$

算法分析

- **算法的度量**
  - 空间复杂度
    - ✓算法所需存储空间的度量记作
      S(n)=O(f(n))
    - ✓n是问题的规模
    - ✓算法的存量包括：
      - 输入数据所占空间
      - 程序本身所占空间
      - 辅助变量所占空间

本章小结

- 数据结构的定义（*）
- 逻辑结构 & 存储结构（*）
- 抽象数据类型ADT的定义（*）
- 算法的基本概念& 性质
- 算法度量：时间复杂度&空间复杂度（*）



课外学习

- **复习C/C++语言**
  - 基本输入/输出
  - 函数与参数传递
  - 结构体及运用

作业

- 1.什么是数据结构？数据结构的三要素是什么?
- 2.算法与程序的区别?
- 3.计算以下程序段的时间复杂度

1）
```
x=n;
y=0;
while(x>(y+1)*(y+1)){y++}
```

2)
```
for(i=1;i<=n;i++)
  {
     for(j=1;j<=i;j++){
        for(k=1;k<=j;k++){
           x=x+1;
        }
     }
  }
```

3)
```
x=91;
y=100;
while(y>0){
    if(x>100){
       x=x-10;
       y=y-1;
    }else{
       x++;
    }
}
```

下一章内容

- **介绍第一种线性结构——线性表**
  - 线性表的定义
  - 线性表的存储结构
  - 线性表的操作