1.2 基本概念和术语

一、数据与数据结构

数据:

所有能输入到计算机中，且能被计算机程序处理的符号的总称。

是计算机操作的对象的总称。

是计算机处理的信息的某种特定的符号表示形式。

数据元素：

是数据（集合）中的一个“个体”，是数据结构中讨论的基本单位。可由若干个数据项组成。

数据项：

是数据结构中讨论的最小单位。数据元素可以是数据项的集合。

例如：描述一个运动员的数据元素可以是

数据对象：

是性质相同的数据元素的集合，是数据的一个子集。

例如：

整数数据对象是集合N={0，1，-1，2，-2，…}，

字母字符数据对象是集合C={‘A’，‘B’，…，‘Z’}

数据结构：

带结构的数据元素的集合。

是相互之间存在一种或多种特定关系的数据元素的集合。

假设用三个4位的十进制数表示一个含 12 位数的十进制数。

例如:

3214,6587,9345 — a1（3214），a2（6587），a3（9345）

则在数据元素 a1、a2 和 a3 之间存在着“次序”关系

<a1,a2>、<a2,a3>

3214，6587，9345 ≠ 6587，3214，9345
a1 a2 a3 a2 a1 a3

又例，在2行3列的二维数组{a1, a2, a3, a4, a5, a6} 中六个元素之间存在两个关系:

a1	a2	a3
a4	a5	a6

行的次序关系：

row = {<a1,a2>,<a2,a3>,<a4,a5>,<a5,a6>}

列的次序关系：

col = {<a1,a4>,<a2,a5>,<a3,a6>}

a1 a3 a5
a2 a4 a6 ≠ a1 a2 a3
a4 a5 a6

再例，在一维数组{a1, a2, a3, a4, a5, a6}的数据元素之间存在如下的**次序关系**:

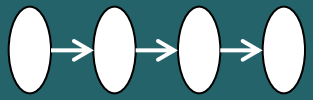
$$\{<a_i, a_{i+1}> | i=1, 2, 3, 4, 5\}$$

可见，不同的“**关系**”构成不同的“**结构**”。

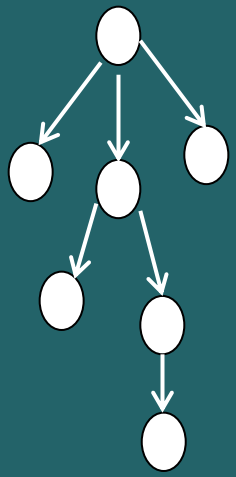
或者说，数据结构是相互之间存在着某种逻辑关系的数据元素的集合。

数据的逻辑结构可归结为以下四类：

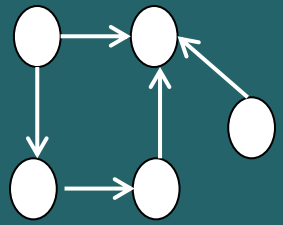
线性结构



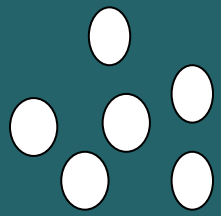
树形结构



图结构或网状结构



集合结构



数据结构的形式定义为:

数据结构是一个二元组:Data_Structures = （D, S）

其中: D 是数据元素的有限集，

S 是 D上关系的有限集。

数据的存储结构：逻辑结构在计算机中的表示。

“数据元素”的映象 ？

“关系”的映象 ？

数据元素的映象方法：

例：用二进制位(bit)的位串表示数据元素

$(321)_{10} = (501)_8 = (101000001)_2$

$A = (101)_8 = (001000001)_2$

关系的映象方法：

在计算机中有两种不同的表示方法：顺序映像和非顺序映像，并由此得到两种不同的存储结构：顺序存储结构和链式存储结构。。

例如：（表示<x, y>的方法**）**

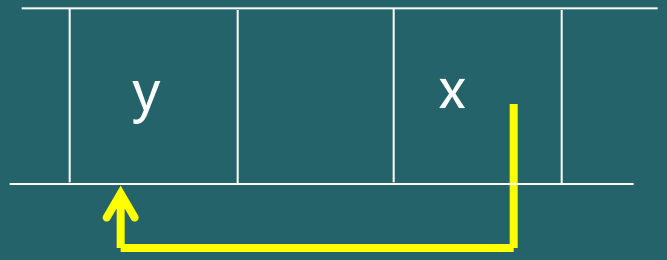
顺序存储结构：

以相对的存储位置表示后继关系。

例如：令 y 的存储位置和 x 的存储位置之间差一个常量 C。而 C 是一个隐含值，整个存储结构中只含数据元素本身的信息。

链式存储结构：

以附加信息（指针）表示后继关系。需要用一个和 x 在一起的附加信息指示 y 的存储位置。



在不同的编程环境中，存储结构可有不同的描述方法。

当用高级程序设计语言进行编程时，通常可用高级编程语言中提供的数据类型描述之。

例如:

以三个带有次序关系的整数表示一个长整数时，可利用 C 语言中提供的整数数组类型。

定义长整数为：

```
typedef int Long_int [3]
```

数据类型：

在用高级程序语言编写的程序中，必须对程序中出现的每个变量、常量或表达式，明确说明它们所属的数据类型。

例如，C 语言中提供的基本数据类型有：

整型 int

浮点型 float

双精度型 double

} 实型（ C++语言）

字符型 char

逻辑型 bool （ C++语言）

不同类型的变量，其所能取的值的范围不同，所能进行的操作不同。

数据类型：

是一个值的集合和定义在此集合上的一组操作的总称。

抽象数据类型（Abstract Data Type 简称ADT）：

是指一个数学模型以及定义在此数学模型上的一组操作。

例如，抽象数据类型复数的定义：

ADT Complex{

数据对象：

D = {e1,e2 | e1,e2∈RealSet }

数据关系：

R1 = {<e1,e2> | e1是复数的实数部分

| e2 是复数的虚数部分}

基本操作：

AssignComplex(&Z, v1, v2)

操作结果：构造复数 Z，其实部和虚部 分别被赋以参数 v1 和 v2 的值。

DestroyComplex(&Z)

操作结果：复数Z被销毁。

GetReal(Z, &realPart)

初始条件：复数已存在。

操作结果：用realPart返回复数Z的实部值。

GetImag(Z, &ImagPart)

初始条件：复数已存在。

操作结果：用ImagPart返回复数Z的虚部值。

Add(z1,z2, &sum)

初始条件：z1， z2是复数。

操作结果：用sum返回两个复数z1，z2 的和值。

} ADT Complex

ADT 有两个重要特征:

数据抽象：

用ADT描述程序处理的实体时，强调的是其本质的特征、其所能完成的功能以及它和外部用户的接口（即外界使用它的方法）。

数据封装：

将实体的外部特性和其内部实现细节分离，并且对外部用户隐藏其内部实现细节。