数据结构与算法

疫情面前，让我们一起努力！

1

数据结构与算法

2

投票 最多可选4项

可爱的同学们，上周线上我们学习了哪些内容？

A 什么是数据结构

B 数据结构的基本概念和术语

C 知道了典型逻辑结构

D 了解了数据的存储结构

9/10/2020

数据结构

由于数据的表示方法和组织形式直接关系到程序对数据的处理效率，而系统程序和许多应用程序的规模很大，结构相当复杂，处理对象又多为非数值性数据。要求人们对计算机程序加工的对象进行系统的研究，即研究数据的特性以及数据之间存在的关系——**数据结构**。

基本概念

1. 数据（Data）：

所有能输入到计算机中并被计算机程序处理的符号的总称。可以是数值数据，也可以是非数值数据。

2. 数据元素（Data element）：

组成数据的基本单位，在计算机中通常作为一个整体进行考虑和处理(又称结点、记录)。

3. 数据项（Data item）：

数据的最小标识单位，属性的描述，域、字段。

基本概念

4.数据结构（Data Structure）：

数据结构
- 逻辑结构
- 存储结构
- 运算的集合

按某种逻辑关系组织起来的一批数据,按一定的存储表示方式把它们存储在计算机的存储器中,并在这些数据上定义了一个运算的集合,就称为一个数据结构.

相互之间存在一种或多种特定关系的数据元素的集合.

6

根据数据元素间的不同特性，通常有以下4种基本逻辑结构：

- 逻辑结构
  - 1. 线性结构
  - 非线性结构
    - 2．集合
    - 3. 树形结构
    - 4．图型或网状结构

数据元素间的逻辑结构可形式描述为：

DS=(D，S)

其中：D 是数据元素的有限集合；

S是 D上的有限关系集合；

逻辑结构

多选题 2分

属于非线性的逻辑结构有哪些？

A 线性结构

B 树形结构

C 集合

D 图形结构

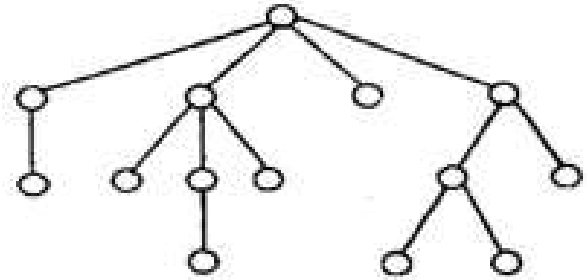
数据结构与算法

线性 

【线性结构】—— 1对1的关系。

线性

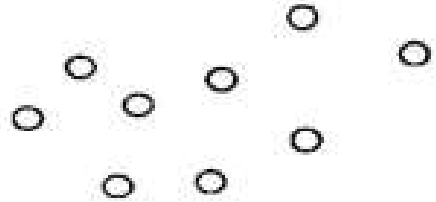
非线性

树 

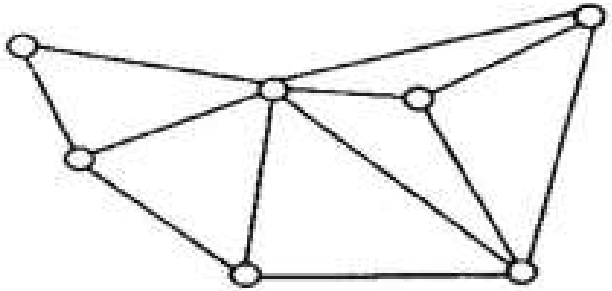
【树形结构】—— 1对多的关系。

层次结构

群结构

集合 

【集合】—— 数据元素间除了“同属于一个集合”外，无其他关系。

图 

【图形结构】—— 多对多的关系。

逻辑结构

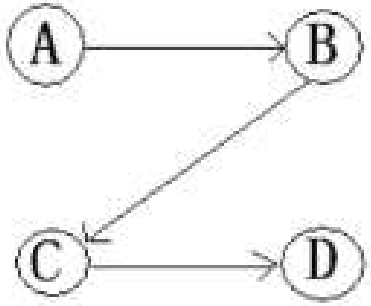
存储结构

数据的逻辑结构在计算机中的存储形式称为数据的存储结构。

就是研究如何把数据元素存储到计算机的存储器中。数据存储结构中不仅存放各数据元素信息，还存放数据关系的信息。

通常存储结构形式有两种：顺序存储和链式存储

（1）第一种存储方式：



头指针 | 103

101	*C*	
102		
103	*A*	
104		
105	*B*	
106	*D*	

11

（2）第二种存储方式：

把数据元素依次存放在地址连续的存储单元里，逻辑上相邻的元素则其在存储单元中的物理位置也相邻。

{at，bat，cat，dat，eat，fat，gat，hat}

101	102	103	104	105	106	107	108

12

（1）顺序存储结构：

把数据元素依次存放在地址连续的存储单元里，逻辑上相邻的元素则其在存储单元中的物理位置也相邻。

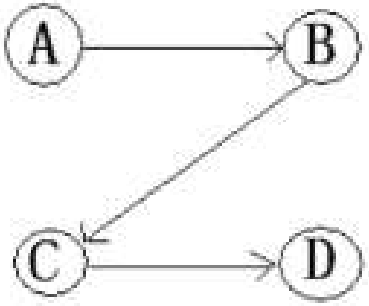
{at，bat，cat，dat，eat，fat，gat，hat}

101	*102*	*103*	*104*	*105*	*106*	*107*	*108*

14

（2）链式存储方式：

数据元素存放在任意单元里，这组存储单元物理地址可以连续，也可以不连续。



头指针 103

101	*C*	
102		
103	*A*	
104		
105	*B*	
106	*D*	

15

适当放松一下，咱们的课程即将正式开启！

16

数据类型(Data Type):

用于刻划数据对象的类型。

一个值的集合以及定义在该值的集合上的一组操作的总称.

一个数学模型以及定义在该模型上的一组操作。

（Abstract Data Type）

ADT 抽象数据类型名

{ Data： 构成该抽象类型所必需的基本数据元素

Relation： 数据元素间的关系

Operation：

操作1

初始条件:

操作结果:

操作2

初始条件:

操作结果:

.....

} ADT 抽象数据类型名

抽象数据类型

❖例 抽象数据类型复数的定义

抽象数据类型

ADT Complex {

数据对象: D={C_1,C_2| C_1,C_2 R}

数据关系: S={< C_1,C_2 >| C_1,C_2 D}

基本操作:

Create(x,y,&z)

初始条件: 已知两个实数x,y

操作结果: 生成一个复数z=x+iy

Add(z_1,z_2,&sum)

初始条件: 已知两个复数$z_1=x_1+iy_1$, $z_2=x_2+iy_2$

操作结果: 得到z_1和 z_2两个复数的和

sum=$(x_1+x_2)+i(y_1+y_2)$

Subtract (z_1,z_2,&dif)

初始条件:已知两个复数$z_1=x_1+iy_1$, $z_2=x_2+iy_2$

操作结果: 得到z_1和 z_2两个复数的差 $dif=(x_1-x_2)+i(y_1-y_2)$

Multiply(z_1,z_2,&pro)

初始条件:已知两个复数$z_1=x_1+iy_1$, $z_2=x_2+iy_2$

操作结果: 得到z_1和 z_2两个复数的积

$pro=(x_1.x_2-y_1.y_2)+i(x_1.y_2+x_2.y_1)$

Get_Realpart(z)

初始条件:已知复数z=x+iy

操作结果: 得到复数z的实部x

Get_Imagpart(z)

初始条件:已知复数z=x+iy

操作结果: 得到复数z的虚部y

}ADT Complex

◆

抽象数据类型

什么是算法？

什么是“好”的算法？

思考：求1+2+3+.....+100的程序？

算法1：

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

算法2：

```
int i, sum = 0,n = 100;
sum =  （1 + n）  * n / 2;
printf ( " %d " , sum);
```

算法

解决特定问题求解步骤的描述，在计算机中表现为指令的有限序列，并且每条指令表示一个或多个操作。

特性 有穷性、确定性、可行性、输入性、输出性

设计要求 正确性、可读性、健壮性、时间效率高、存储量低

时间复杂度

算法1：（n+2次）

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

算法2：（3次）

```
int i, sum = 0,n = 100;
sum = （1 + n） * n / 2;
printf ( " %d " , sum);
```

对于同样问题的输入规模n：

算法1：执行次数f(n)= n+2

算法2：无论n为多少，f(n)= 3

分析一个算法的运行时间时，需把算法执行次数与输入规模关联起来，即生成执行次数与问题输入规模的函数。