



# 数据结构

(C语言版) (第2版)

## 绪论

# 数据结构基本概念及研究内容

主讲教师：汪红松



## 课程目的



能够分析研究计算机加工的对象特性，获得其逻辑结构，根据需求，选择合适存贮结构及其相应的算法；



学习一些常用的算法；



程序设计的训练过程，要求编写的程序结构清楚和正确易读；



了解算法的时间分析和空间分析技术。



## 教学目标

01

OPTION

熟悉数据结构的一些基本概念

02

OPTION

了解抽象数据类型的表示和实现

03

OPTION

理解算法的概念和特性

04

OPTION

了解算法时间复杂度和空间复杂度的分析



# 教学内容 Contents

- 一、 数据结构基本概念和术语
- 二、 算法的内涵及效率评价

老师

## ▶▶▶ 一、数据结构基本概念和术语

### 1.数据 ( data )

所有能输入到计算机中去的描述客观事物的符号。

◆ 数值性数据

◆ 非数值性数据 ( 多媒体信息处理 )

### 2.数据元素 ( data element )

数据的基本单位，也称结点 ( node ) 或记录 ( record )。

### 3.数据项 ( data item )

有独立含义的数据最小单位，也称域(field)

### 4.数据对象(Data Object) :

相同特性数据元素的集合，是数据的一个子集。

- ◆ 整数数据对象

$$N = \{ 0, 1, 2, \dots \}$$

- ◆ 学生数据对象

学生记录的集合

“结构”就是指数据元素之间存在的关系。

### 5.数据结构 ( Data Structure )

相互之间存在一种或多种特定关系的数据元素的集合。

## ▶▶▶ 一、数据结构基本概念和术语

数据结构的两个层次：

逻辑结构



数据元素间抽象化的相互关系，与数据的存储无关，独立于计算机。

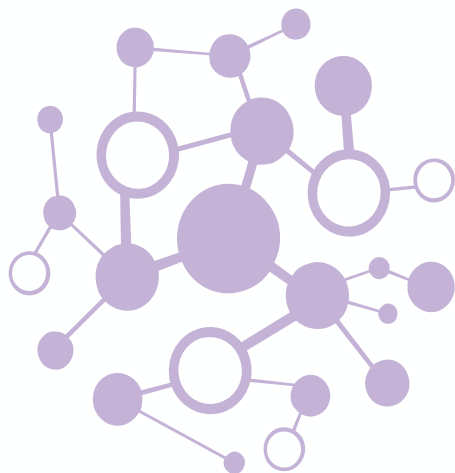
存储结构（物理结构）



数据元素及其关系在计算机存储器中的存储方式。

## 6. 逻辑结构

---



### (1) 线性结构——

OPTION

有且仅有一个开始和一个终端结点，并且所有结点都最多只有一个直接前趋和一个后继。

例如：线性表、栈、队列、串

### (2) 非线性结构——

OPTION

一个结点可能有多个直接前趋和直接后继。

例如：树、图



## 7. 存储结构

---

### (1) 顺序存储结构

借助元素在存储器中的相对位置来表示数据元素间的逻辑关系。

### (2) 链式存储结构

借助指示元素存储地址的指针表示数据元素间的逻辑关系。

## 8.数据类型

定义：在一种程序设计语言中，变量所具有的数据种类

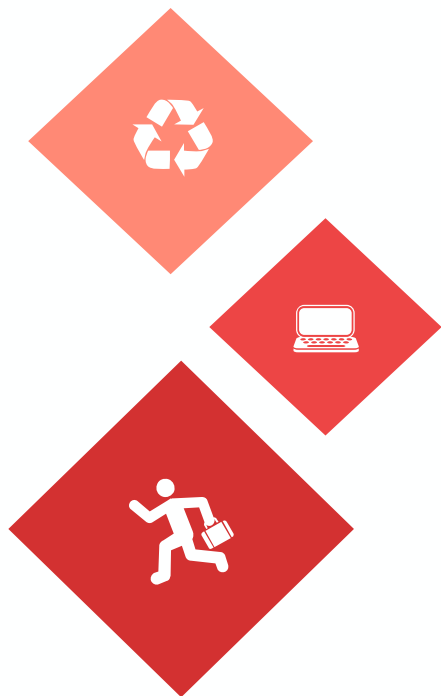
C语言：基本数据类型：char int float double void

构造数据类型：数组、结构体、共用体、文件

数据类型是一组性质相同的值的集合, 以及定义于这个集合上的一组运算的总称。

## 9.抽象数据类型

(ADTs: Abstract Data Types)



更高层次的数据抽象。

---

由用户定义，用以表示应用问题的数据模型。

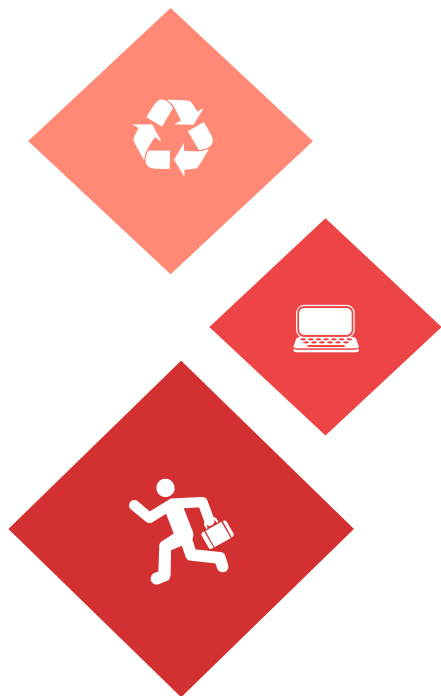
---

由基本的数据类型组成, 并包括一组相关的操作。

---

## 9.抽象数据类型

(ADTs: Abstract Data Types)



更高层次的数据抽象。

---

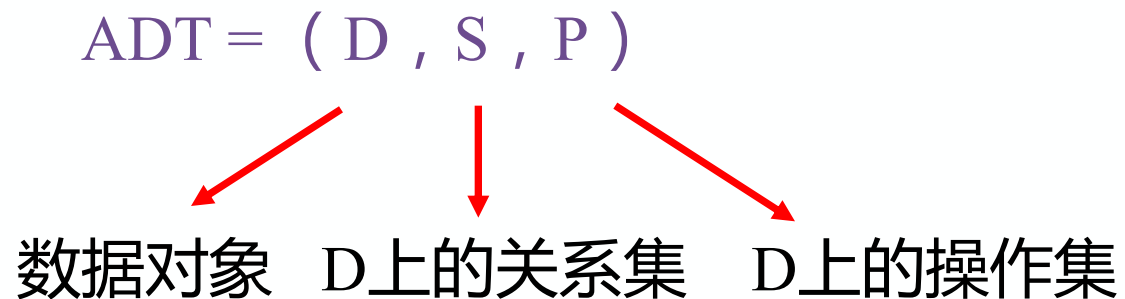
由用户定义，用以表示应用问题的数据模型。

---

由基本的数据类型组成, 并包括一组相关的操作。

---

## 9.抽象数据类型



### ADT常用定义格式

ADT抽象数据类型名 {

    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>

} ADT抽象数据类型名

## 9.抽象数据类型

抽象数据类型的  
表示与实现



可以通过**固有的**数据类型（如整型、实型、字符型等）来表示和实现。



有些类似C语言中的**结构（struct）类型**，但增加了相关的**操作**。



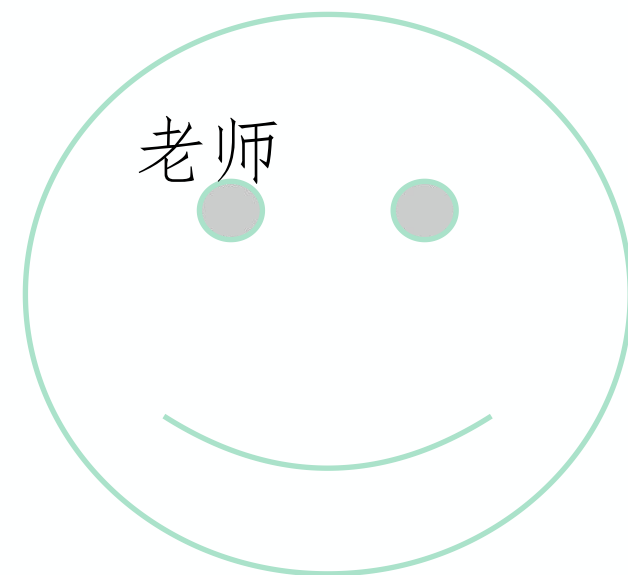
**具体实现**，如C或C++语言等。

# 教学内容 Contents

一、 数据结构基本概念和术语

二、 算法的内涵及效率评价

老师



## 二、算法的内涵及效率评价

### 1. 算法和算法分析

#### 算法定义

一个有穷的指令集，  
这些指令为解决某一  
特定任务规定了一个  
运算序列。



#### 算法的描述

自然语言  
流程图  
程序设计语言  
伪码





## 二、算法的内涵及效率评价

### 1.算法和算法分析

#### (1) 算法效率

用依据该算法编制的程序在计算机上执行所消耗的时间来度量

- **事后统计法**需要先将算法实现，然后测算其时间和空间开销。  
同一个算法用不同的语言、不同的编译程序、在不同的计算机上运行，效率均不同→使用**绝对时间单位**衡量算法效率不合适。
- **事前分析估算法**，通过计算算法的渐进复杂度来衡量算法的效率。
  -

### 2.时间复杂度的渐进表示法

算法中基本语句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 $n$ 的增大，算法执行的时间的增长率和 $f(n)$ 的增长率相同，称渐近时间复杂度。

## 二、算法的内涵及效率评价

### 2.时间复杂度的渐进表示法

#### $n * n$ 阶矩阵加法

```
for( i = 0; i < n; i++)  
    for( j = 0; j < n; j++)  
        c[i][j] = a[i][j] + b[i][j];
```

**语句的频度** ( Frequency Count ): 重复执行的次数 :  $n*n$ ;

$$T(n) = O(n^2)$$

矩阵加法的运算量和问题规模的平方是同一个量级

### 3.分析算法时间复杂度的基本方法

- 找出语句频度最大的那条语句作为基本语句
- 计算基本语句的频度得到问题规模 $n$ 的某个函数 $f(n)$
- 取其数量级用符号“O”表示

```
x = 0; y = 0;  
for ( int k = 0; k < n; k ++ )  
    x ++;  
for ( int i = 0; i < n; i ++ )  
    for ( int j = 0; j < n; j ++ )  
        y ++;
```


$$T(n) = O(n^2)$$

$$f(n) = n^2$$

### 4. 渐进空间复杂度

空间复杂度: 算法所需存储空间的度量,  
记作:  $S(n) = O(f(n))$   
其中  $n$  为问题的规模(或大小)。

算法要占  
据的空间

算法本身要占据的空间  
，输入/输出，指令，  
常数，变量等

算法要使用的  
辅助空间

### 4. 渐进空间复杂度

例：将一维数组a中的 $n$ 个数逆序存放到原数组中。

$S(n) = O(1)$   
原地工作

【算法1】

```
for(i=0; i<n/2; i++)  
{  
    t=a[i];  
    a[i]=a[n-i-1];  
    a[n-i-1]=t;  
}
```

$S(n) = O(n)$

【算法2】

```
for(i=0; i<n; i++)  
    b[i]=a[n-i-1];  
for(i=0; i<n; i++)  
    a[i]=b[i];
```

- 1.数据、数据元素、数据项、数据结构等基本概念
- 2.对数据结构的两个层次的理解
  - 逻辑结构
  - 存储结构
- 3.抽象数据类型的表示方法
- 4.算法、算法的时间复杂度及其分析方法

