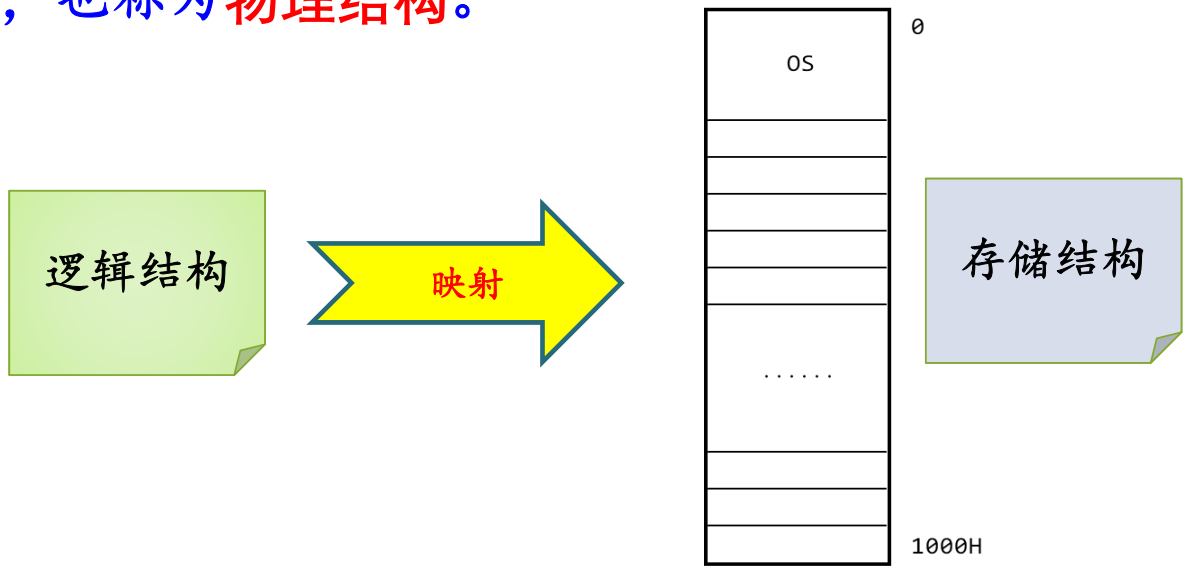
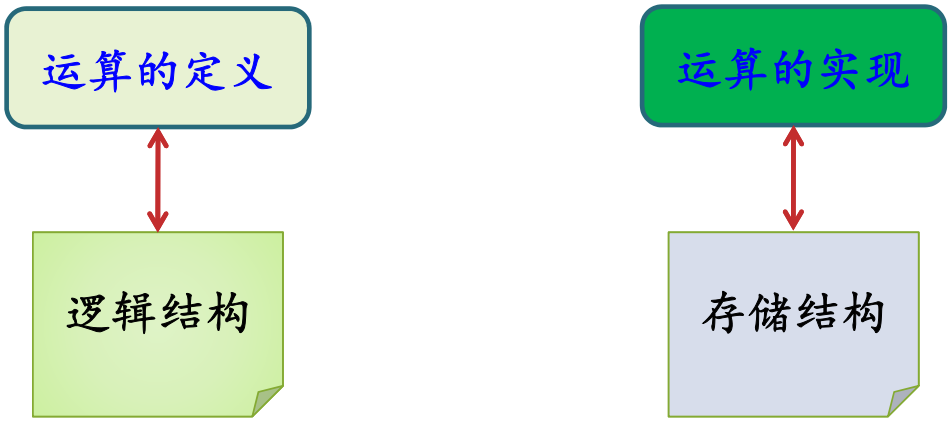
存储结构

- 数据的逻辑结构在计算机内存中的存储方式称为数据的存储结构，也称为物理结构。



- 实际上，同一种逻辑结构可以有多种存储结构，在不同的存储结构中，实现同一种运算的算法可能不同。

■ 逻辑结构、存储结构和运算三者之间的关系：



□ 将逻辑结构映射为存储结构时，需要存储逻辑结构中的：

- 所有元素。
- 元素之间的关系。

■ 主要的几种存储结构

1. 顺序存储结构

- 顺序存储结构采用一组连续的存储单元存放所有的数据元素。
- 逻辑上相邻的元素，其存储单元也相邻。也就是说，元素之间的逻辑关系由存储单元地址间的关系隐含表示，即顺序存储结构将数据的逻辑结构直接映射到存储结构。

◆ 顺序存储结构可实现对各数据元素的随机存取。

即顺序存储结构具有随机存取特性。

➢ 随机存取是指给定某元素的逻辑序号i，能在常量时间内查找到对应的元素值。

- 假设每个数据元素占用30个字节，且从100号单元开始由低地址向高地址方向存储。

学号	姓名	分数
201340701067	张茜	82
201340701001	刘雯月	85
201340701047	孟凡辉	89
201340701028	赵慧	98
201340701011	任豪	95

则Score对应的顺序存储结构如下：

地址	学号	姓名	分数
100	201340701067	张茜	82
130	201340701001	刘雯月	85
160	201340701047	孟凡辉	89
190	201340701028	赵慧	98
220	201340701011	任豪	95

特点

- 所有元素的存储地址是连续的；
- 通过存储关系直接反映逻辑关系。

2．链式存储结构

- 链式存储结构中每个结点单独存储，无需占用一整块存储空间。
- 但为了表示结点之间的关系，给每个结点附加指针字段，用于存放相邻结点在内存中的存储地址。

Score对应的链式存储结构

- 每个存储单元（结点）存放一个学生成绩记录；
- 每个存储单元（结点）还需要存储“下一个元素的存储地址”，即后继元素指针域。
- 使用head标记第一个数据元素的存储位置（地址）来标识整个链式存储结构。

特点

- ✓ 所有结点的地址不一定连续；
- ✓ 一个数据元素（结点）的所有数据项占用一片连续空间；
- ✓ 增加指针域来表示元素之间的逻辑关系。

数据运算

在数据结构中，数据运算就是指插入、删除、查询等操作。

- **运算定义**（或运算描述）：确定运算的功能，是抽象的。
- **运算实现**：在存储结构上确定对应运算实现算法，是具体的。

数据类型和抽象数据类型

1、数据类型

数据类型是高级程序设计语言中的一个基本概念，用于描述变量的存储方式。

- **数据类型**是一组性质相同的**值的集合**和定义在此集合上的**一组操作**的总称。

C/C++语言中常用的数据类型

- int型：可以有3个修饰符：short（短整数）、long（长整数）和unsigned（无符号整数）
- float型
- double型
- char型
- 指针类型
- 结构体类型
- ……

C语言中的自定义类型

- C/C++语言中允许使用typedef关键字为一个数据类型指定一个别名，例如：

```
typedef char ElemType;
```

该语句将char类型与ElemType等同起来。这样做有两个好处：

- 方便程序调试，例如，将上述语句改为typedef int ElemType，则程序中所有ElemType都改为int类型了。
- 可以简化代码。

```
typedef struct student        //student结构体类型
{   int no;
    char name[10];
    char sex;
    int class;
} StudType;        //用StudType别名表示student结构体类型
```

```
StudType s1,s2;
```

```
struct student s1,s2;
```

等价

2、抽象数据类型

- 抽象数据类型（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。
- 定义一个抽象数据类型时，需要给出其名称和各运算名称及其功能描述。
- 抽象数据类型需要通过固有数据类型（高级编程语言中已实现的数据类型）来实现。

- 从数据结构的角度看，一个求解问题可以通过抽象数据类型来描述。

  - 也就是说，抽象数据类型（ADT）对一个求解问题从逻辑上进行了准确的定义，所以抽象数据类型由数据逻辑结构和运算定义两部分组成。

抽象数据类型（ADT）＝数据逻辑结构＋运算定义

【示例-3】定义单个集合的抽象数据类型ASet，其中所有元素为正整数，包含创建一个集合、输出一个集合和判断一个元素是否为集合中元素的基本运算。

在此基础上再定义两个集合运算的抽象数据类型BSet，包含集合的并集、差集和交集运算。

解：（1）抽象数据类型ASet的定义如下：

ADT ASet

{

数据对象：**D={d_i | 0≤*i*≤*n*，*n*为一个正整数}**

运算的定义：

void cset(&s,a,n)：由含*n*个元素的数组*a*创建一个集合*s*。

void dispset(s)：输出集合*s*。

int inset(s,e)：判断元素*e*是否为集合*s*中的元素。

}

（2）抽象数据类型BSet的定义如下：

ADT BSet

{

数据对象：**D={ $s_i \in$ ASet | $0 \le i \le n$，n为一个正整数}**

运算的定义：

void add(s_1,s_2,s_3)：$s_3=s_1 \cup s_2$ //求集合的并集

void sub(s_1,s_2,s_3)：$s_3=s_1-s_2$ //求集合的差集

void intersection(s_1,s_2,s_3)：$s_3=s_1 \cap s_2$ //求集合的交集

}

END