用计算机解决一个具体问题大致需要3个步骤：

1. 分析问题，确定数据模型；
2. 设计相应的算法；
3. 编写程序，运行并调试程序，直至得到正确结果。

* **基本概念**

**数据**是描述客观事物的数字和字符的组合。

**数据元素**通常作为数据的基本单位。例如，每个班级中的每个学生记录都是一个数据元素。在有些情况下，数据元素也称为元素、结点、顶点或者记录等。一个数据元素可以由若干个数据项组成。

**数据项**是具有独立含义的数据最小单位，也称为字段或域。

**数据对象**是指性质相同的数据元素的集合，它是数据的一个子集。在数据结构课程中讨论的数据通常指的是数据对象。

**数据结构**是指所有数据元素及数据元素之间的关系，可以看作是相互之间存在看某种特定关系的数据元素的集合。因此，我们可以把数据结构看成是带结构的数据元素的集合。

**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，是某种程序设计语言中已实现的数据结构。比如说Java中：int型是整数集合，并限制了其上的数据操作只能是加减乘除、取余及比较、移位等；String类型数据只能做字符串比较。

**抽象数据类型**

* **逻辑结构**

1. **集合**是指数据元素之间除了“同属于一个集合”的关系以外别无关系。
2. **线性结构**是指该结构中的数据元素之间存在一对一的关系。其特点是开始元素和终端元素都是唯一的，除了开始元素和终端元素以外，其余元素都有且仅有一个前驱元素和一个后继元素。线性表就是一种典型的线性结构。
3. **树形结构**是指该结构中的数据元素之间存在一对多的关系。其特点是除了开始元素以外，每个元素有且仅有一个前驱元素，除了终端元素以外，每个元素有一个或多个后继元素。二叉树就是一种典型的树形结构。
4. **图形结构**

* **存储结构**

1. **顺序存储结构**

是采用一组连续的存储单元存放所有的数据元素，也就是说，所有数据元素在存储器中占有一整块存储空间，而且两个逻辑上相邻的元素在存储器中存储位置也相邻。因此，数据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。

主要优点：

①存储效率高。因为分配给数据的存储单元全用于存放数据元素，元素之间的逻辑关系没有占用额外的存储空间；

②可实现对元素的随机存取。即每个元素对应一个逻辑序号，由该序号可直接计算出对应元素的存储地址，从而获取元素值。

主要缺点：

不便于数据修改，对元素的插入或删除操作可能需要移动一些列的元素。

例如，采用C/C++中的结构体数组来存储学生表：

|  |
| --- |
| struct{  int no;  char name[8];  char sex[2];  char class[4]  }Stud[7]={{1,”张斌”,”男”,”9901”},...{5,”刘丽”,”女”,”9901”}};  其中，数组名称Stud作为数组的起始地址，用于唯一标识该存储结构。 |

1. **链式存储结构**

在链式存储结构中，每个逻辑元素用一个内存结点存储，每个结点是单独分配的，所有的节点地址不一定是连续的，所以无需占用一整块存储空间。为了表示数据元素之间的逻辑关系，给每个结点附加指针域，用于存放相邻接点的存储地址，也就是通过指针域将所有结点链接起来。

主要优点：便于数据修改，在对元素进行插入或删除操作时仅需要修改相应结点的指针域，不必移动结点。

主要缺点：

①与顺序存储结构相比，存储空间的利用率较低，因为分配给元素的存储单元有一部分被用来存储节点之间的逻辑关系；

②由于逻辑上相邻的元素在存储空间中不一定相邻，所以不能对元素进行随机存取。

|  |
| --- |
| typedef struct Studnode{  int no;  char name[8];  char sex[2];  char class[4];  struct Studnode \*next; //存储指向下一个学生结点的指针  }StudType； //结点类型  学生表中的每个学生记录采用一个StudType类型的结点单独存储，一个学生结点的next域指向逻辑结构中他的后继学生记录对应的结点，从而构成一个链表。 |

1. **索引存储结构**

是指在存储数据元素信息的同时还建立附加的索引表。存储所有数据元素信息的表称为主数据表，其中，诶个数据元素有一个关键字和对应的存储地址。

索引表中的每一项称为索引项，索引项的一般形式为“关键字，地址”。其中，关键字唯一标识一个元素，地址对应该关键字的元素在主数据表中的存储地址。通常，索引表中的所有索引项是按关键字有序排列的。

在按关键字查找时，首先在索引表中利用关键字的有序性快速查找到该关键字的地址，然后通过改地址在主要数据表中找到对应的元素。

优点：查找效率高；

缺点：需要建立索引表，从而增加了空间开销。

1. **哈希（或散列）存储结构**

基本思想是根据元素的关键字通过哈希（或散列）函数直接计算出一个值，并将这个值作为该元素的存储地址。

优点：查找速度快。只要给出待查元素的关键字就可立即计算出该元素的存储地址。

缺点：与前3种存储方法不同的是，哈希存储方法之存储元素的数据，不存储元素之间的逻辑关系，所以哈希存储结构一般只适合要求对数据能够进行快速查找和插入的场合。

上述4种方法既可以单独使用，也可以组合使用。同一种逻辑结构采用不同的存储方法可以得到不同的存储结构。选择何种存储结构来表示相应的逻辑结构视具体要求而定，主要考虑的是运算方便及算法的时空要求。

**算法**是对特定问题求解步骤的一种描述，它是指令的有限序列。

**算法的特点**：有穷性；确定性；可行性；有输入；有输出；

注：算法区别于程序。程序是指实用某种计算机语言对一个算法的具体实现，即要怎么做；而算法侧重于对解决问题的方法描述，即要做什么。算法必须满足有穷性，而程序不一定满足有穷性，如Windows操作系统在用户没有退出、硬件不出现故障以及不断电的情况下理论上可以无限时运行，所以严格地讲算法和程序是两个不同的概念。然，算法也可以用计算机程序来描述，这样算法和程序就是一回事了。

**算法设计的目标**：①正确性；②可使用性；③可读性；④健壮性；⑤高效率与地存储量需求。

* **算法时间复杂度分析**

**事后统计法**就是编写算法对应程序，统计其执行时间。一个算法用计算机语言实现后，在计算机上执行所消耗的时间与很多因素有关，如计算机的运行速度、编写程序采用的计算机语言、编译产生的机器语言代码质量和问题的规模等。这种方法存在的缺点有二：意识必须执行程序，而是存在很多因素掩盖了算法本质。

**事前估算法**是撇开这些与计算机硬件、软件有关的因素，仅考虑算法本身的效率高低，可以认为一个特定算法的“运行工作量”的大小只依赖于问题的规模。或者说算法的执行时间是问题规模的函数。

1. **计算算法的频度**
2. **渐进时间复杂度**

* **算法空间复杂度分析**