数据（data）是对客观事物的符号表示，在计算机科学中是指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。

数据元素（data element）是数据的基本单位，在计算机程序中通常作为一个整体进行考虑和处理。

数据项：有时，一个数据元素可由若干个数据项（data item）组成。数据项是数据的不可分割的最小单位。

数据对象（data object）是性质相同的数据元素的集合，是数据的一个子集。

|  |  |  |
| --- | --- | --- |
| 数据结构（data structure）是相互之间存在一种或多种特定关系的数据元素的集合。数据元素相互之间的关系称为结构（structure）。  形式定义为：数据结构是一个二元组Data\_Structure。其中，是数据元素的有限集，是上关系的有限集。 | | |
| ┣ | 逻辑结构：结构定义中的“关系”描述的是数据元素之间的逻辑关系，因此又称为数据的逻辑结构。 | |
| ┃ | ┣ | 集合：结构中的数据元素之间除了“同属于一个集合”的关系外，别无其他关系。 |
| ┃ | ┃ | 线性结构：结构中的数据元素之间存在一个对一个的关系。 |
| ┃ | ┃ | 树形结构（树型结构）：结构中的数据元素之间存在一个对多个的关系。 |
| ┃ | ┗ | 图状结构（网状结构）：结构中的数据元素之间存在多个对多个的关系。 |
| ┗ | 物理结构（存储结构）：数据结构在计算机中的表示（又称映像）称为数据的物理结构，又称存储结构。  位：在计算机中表示信息的最小单位是二进制数的一位，叫做位（bit）。  元素（结点）：在计算机中，我们可以用一个由若干位组合起来形成的一个位串表示一个数据元素，通常称这个位串为元素（element）或结点（node）。  数据域：当数据元素由若干数据项组成时，位串中对应于各个数据项的子位串称为数据域（data field）。 | |
|  | ┣ | 顺序映像的特点是借助元素在存储器中的相对位置来表示数据元素之间的逻辑关系。  顺序存储结构 |
|  | ┗ | 非顺序映像的特点是借助元素存储地址的指针（pointer）表示数据元素之间的逻辑关系。  链式存储结构 |

任何一个算法的设计取决于选定的数据（逻辑）结构，而算法的实现依赖于采用的存储结构。

|  |  |  |
| --- | --- | --- |
| 数据类型（data type）是和数据结构密切相关的一个概念，它最早出现在高级程序语言中，用以刻画（程序）操作对象的特性。数据类型是一个值的集合和定义在这个集合上的一组操作的总称。 | | |
| ┗ | 按值的不同特性（！） | |
|  | ┣ | （非结构的）原子类型：原子类型的值是不可分解的。 |
|  | ┗ | 结构类型：结构类型的值是由若干成分按某种结构组成的，因而是可以分解的，并且它的成分可以是非结构的，也可以是结构的。 |

固有数据类型：处理器中已经定义并实现的数据类型（原子类型、结构类型）。

|  |  |  |
| --- | --- | --- |
| 抽象数据类型（Abstract Data Type,ADT）是指一个数学模型以及定义在该模型上的一组操作。  抽象数据类型的定义仅取决于它的一组逻辑特性。  抽象数据类型和数据类型实质上是一个概念。  和数据结构的形式定义相对应，抽象数据类型可用以下三元组表示。其中，是数据对象，是上的关系集，是对的基本操作集。 | | |
| ┗ | 按值的不同特性（！） | |
|  | ┣ | 原子类型（atomic data type）：属原子类型的变量的值是不可分解的。 |
|  | ┣ | 固定聚合类型（fixed-aggregate data type）：属该类型的变量，其值由确定数目的成分按某种结构组成。 |
|  | ┗ | 可变聚合类型（variable-aggregate data type）：和固定集合类型的值相比较，构成可变聚合类型“值”的成分的数目不确定。 |

固定聚合类型和可变聚合类型统称结构类型。

多形数据类型（polymorphic data type）是指其值的成分不确定的数据类型。

|  |  |  |  |
| --- | --- | --- | --- |
| 算法（algorithm）是对特定问题求解步骤的一种描述，它是指令的有限序列，其中每一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 | |
| ┃ | ┗ | 输出 | |
| ┣ | 描述 | | |
| ┃ | ┣ | 文字 | |
| ┃ | ┣ | 流程图 | |
| ┃ | ┗ | 代码 | |
| ┗ | 要求 | | |
|  | ┣ | 正确性 | |
|  | ┣ | 可读性 | |
|  | ┃ | ┣ | 标识符名称 |
|  | ┃ | ┣ | 空白字符 |
|  | ┃ | ┗ | 注释 |
|  | ┣ | 健壮性 | |
|  | ┗ | 效率与低存储量需求 | |

线性表：个数据元素的有限序列。

一个数据元素可由若干个数据项（item）组成。常把数据元素称为记录（record），含有大量记录的线性表又称文件（file）。

线性表的长度：线性表中元素的个数（）。

空表：。

线性表的顺序表示：用一组地址连续的存储单元依次存储线性表的数据元素。

线性表的起始位置（基地址）：第一个数据元素的存储位置。

线性表的顺序存储结构或顺序映像（sequential mapping）：...。通常，称这种存储结构的线性表为顺序表。

~~线性表的存储结构~~顺序表是一种随机存取的存储结构。

线性表的链式存储结构的特点是用一组任意的存储单元存储线性表的数据元素（这组存储单元可以是连续的，也可以是不连续的）。

结点（node）：数据元素本身的信息和指示其直接后继的信息（即直接后继的存储位置）组成数据元素的存储映像。

数据域：存储数据元素信息的域。

指针域：存储直接后继存储位置的域。

指针（链）：指针域中存储的信息。

链表（线性表的链式存储结构、线性链表、单链表）：个结点（（）的存储映像）链结。

单链表是非随机存取的存储结构。

静态链表：用数组描述的链表。

循环链表（circular linked list）

双向链表（double linked list）

栈（stack、后进先出（last in first out）的线性表|LIFO结构）：限定仅在表尾进行插入或删除操作的线性表。

栈顶（top）：表尾端。

栈底（bottom）：表头端。

空栈：不含元素的空表。

假设，则称为栈底元素，为栈顶元素。

顺序栈，即栈的顺序存储结构是利用一组地址连续的存储单元依次存放自栈底到栈顶的数据元素，同时附设指针top指示栈顶元素在顺序栈中的位置。

队列（queue）是一种先进先出（first in first out，缩写为FIFO）的线性表。它只允许在表的一端进行插入，而在另一端删除元素。

队尾（rear）：允许插入的一端。

队头（front）：允许删除的一端。

假设队列为，那么，就是队头元素，则是队尾元素。

双端队列（deque）：限定插入和删除操作在表的两端进行的线性表。这两端分别称做端点1和端点2。

链队列：用链表表示的队列。一个链队列需要两个分别指示队头和队尾的指针（分别称为头指针和尾指针）才能唯一确定。

循环队列：将顺序队列臆造为一个环状的空间。

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |