CHAPTER 1

# 第1章 绪 论

## 1.1 学习要点

(1) 领会数据、数据类型等基本定义和基本概念。

(2) 理解数据之间的关系和数据结构的概念及内容。对于数据的逻辑结构和存储结构之间的关系,必须分清哪些是逻辑结构的性质,哪些是存储结构的性质。

(3) 理解抽象数据类型的定义、表示和实现方法,能结合实际问题列举抽象数据类型的各种应用,并能自定义满足实际需要的抽象数据类型。

(4) 理解算法的定义、算法特征与算法的描述方法。

(5) 能用 C 语言描述数据结构、数据类型、抽象数据类型及简单算法。掌握计算语句频度和估算算法时间复杂度的方法,对简单算法能通过比较算法的时间与空间复杂性评价性能的优劣。

## 1.2 主要内容及知识点

### 1. 数据结构的基本概念和术语

本节内容包括数据、数据元素、数据对象、存储结构和数据类型等概念术语的含义。数据结构指数据及数据之间的相互关系。一般的,数据结构包括以下 3 个方面的内容:数据的逻辑结构、数据的存储结构(又称数据的物理结构)和数据的运算及实现。数据的逻辑结构通常有集合、线性结构、树型结构和图状或网状结构。数据的存储结构表示数据元素及其关系如何存放在计算机内存的问题,有顺序存储方法、链状存储方法、索引存储方法和散列存储方法 4 种基本存储方法。数据的运算指定义在数据的逻辑结构上的一组操作的集合。每种逻辑结构都允许有某些运算,这些运算实际上是在逻辑结构上对抽象数据所施加的一系列“抽象”的操作。

### 2. 数据类型和抽象数据类型

数据类型是具有相同性质的计算机数据的集合及在这个数据上的一组运算,是与数据结构密切相关的概念。抽象数据类型(ADT)是指一个数据模型及其定义在该数据模型上的一组操作。抽象数据模型的定义仅取决于它的一组逻辑特性,与其在计算机内部如何表示和实现无关。抽象数据类型是算法的一个数据模型连同定义在该模型上的作为

该算法构件的一组运算。

### 3. 算法描述与算法评价

算法是对特定问题求解方法和步骤的一种描述。算法必须具备 5 个重要特性：输入、输出、有穷性、确定性和可行性。算法可以用自然语言、数学语言或约定的符号来描述，也可用计算机高级程序语言来描述。设计一个好的算法可以从正确性、可读性、健壮性和高效率几个方面考虑。评价算法的优劣程度可从时间效率、空间效率和简单性进行评价。算法的时间效率一般可以通过分析程序的语句频度和时间复杂性来进行。算法的空间效率可用空间复杂度来度量。

## 1.3 基础知识题

### 1.3.1 填空题

1. 数据结构即数据的逻辑结构，包括 ① 、② 、③ 和 ④ 4 种类型。树型结构和图型结构合称为 ⑤ 。数据的存储结构即物理结构，包括 ⑥ 、⑦ 、⑧ 和 ⑨ 4 种基本类型。

2. 数据结构是研究数据的 ① 和 ② ，以及它们之间的相互关系，并对这种结构定义相应的 ③ ，设计相应的 ④ ，而确保经过这些运算后得到的新结构是 ⑤ 结构类型。

3. 一个数据结构用二元组表示时，它包括 ① 集合 K 和 K 上 ② 的集合 R。

4. 一个算法应具有 ① 、② 、③ 、④ 和 ⑤ 5 个特性。

5. 一个算法的时间复杂性是该算法包含的 ① 的多少，它是一个算法运行时间的 ② 。一个算法的空间复杂性是指该算法在运行过程中临时占用的 ③ 的大小。

6. 一个算法的时间复杂性通常用它相对于问题的规模的 ① 形式表示。当一个算法的时间复杂性与问题的规模 $n$ 大小无关时，则表示为 ② ；成正比时，则表示为 ③ ；成对数关系时，则表示为 ④ ；成平方时，则表示为 ⑤ 。

7. ① 是描述客观事物的数、字符，以及所有能输入到计算机且被计算机程序加工处理的符号集合。 ② 是数据的基本单位，有时一个 ② 由若干个 ③ 组成，在这种情况下，称 ② 为记录， ③ 是数据的最小单位，而由记录组成的线性表为 ④ 。被计算机加工的 ② 不是孤立无关的，它们彼此之间一般存在某种联系，通常将 ② 间的这种联系称为 ⑤ 。算法的计算量的大小称为计算的 ⑥ 。

**解：**

1. ① 线性结构　② 树型结构　③ 图型结构　④ 集合　⑤ 非线性结构
   ⑥ 顺序　⑦ 链接　⑧ 索引　⑨ 散列
2. ① 物理结构　② 逻辑结构　③ 运算　④ 算法　⑤ 原来的
3. ① 数据元素　② 二元关系
4. ① 有穷性　② 确定性　③ 可行性　④ 0 或多个输入　⑤ 1 或多个输出

5. ① 简单操作次数　② 相对量度　③ 存储空间

6. ① 数量级　② $O(1)$　③ $O(n)$　④ $O(\log_2 n)$　⑤ $O(n^2)$

7. ① 数据　② 数据元素　③ 数据项　④ 文件　⑤ 结构　⑥ 复杂性

## 1.3.2 选择题

1. 下面程序的时间复杂性为_____。

```
for(i=0;i<m;i++)
    for(j=0;j<n;j++)
        A[i][j]=i*j;
```

(A) $O(m^2)$　(B) $O(n^2)$　(C) $O(m\times n)$　(D) $O(m+n)$

2. 执行下面程序段,语句 3 的执行次数为_____。

```
for(i=0;i<n-1;i++)
    for(j=n;j>i;j--)
        state;
```

(A) $n(n+2)/2$　(B) $(n-1)(n+2)/2$　(C) $n(n+1)/2$　(D) $(n-1)(n+2)$

3. 下列程序的时间复杂性为_____。

```
i=0;s=0;
while(s<n)
{i++;
 s=s+i;
}
```

(A) $O(\sqrt{n})$　(B) $O(1)$　(C) $O(n)$　(D) $O(n^2)$

4. 下列程序的时间复杂性为_____。

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

(A) $O(m\times n\times t)$　(B) $O(m+n+t)$　(C) $O(m+n\times t)$　(D) $O(m\times t+n)$

5. 下列程序段的时间复杂性_____。

```
i=1;k=0;n=100;
do{k=k+10*i;
    i=i++;
  }while(i!=n);
```

(A) $O(1)$　(B) $O(n)$　(C) $O(i)$　(D) $O(i\times n)$

6. 下列程序段的时间复杂性为_____。

```
x=n;          /*n>1*/
y=0;
while((x>=(y+1)*(y+1))
    y=y+1;
```

(A) $O(n)$　　(B) $O(\sqrt{n})$　　(C) $O(1)$　　(D) $O(n^2)$

7. 算法指的是_____。

(A) 计算机程序　　(B) 解决问题的计算方法

(C) 排序算法　　(D) 解决问题的有限运算序列

8. 某程序的时间复杂度为$(3n+n\log_2 n+n^2+8)$,其数量级表示为_____。

(A) $O(n)$　　(B) $O(n\log_2 n)$　　(C) $O(n^2)$　　(D) $O(\log_2 n)$

9. 若需要利用形参直接访问实参,则应把形参变量说明为_____参数。

(A) 指针　　(B) 引用　　(C) void　　(D) 值

**解**:

1.(C)　2.(B)　3.(A)　4.(A)　5.(A)　6.(B)　7.(D)

8.(C)　9.(B)

### 1.3.3 应用题

1. 选择解决某种问题的最佳数据结构的标准是什么?

**解**:一般有如下两条标准:

(1) 所需的存储空间量;

(2) 算法所需要的时间。

算法所需要的时间又包括以下几点:

(1) 程序运行时所需要的数据总量;

(2) 源程序进行编译所需要的时间;

(3) 计算机执行每条指令所需的时间;

(4) 程序中的指令重复执行的次数,而本条正是讨论算法中的重点内容。

2. 根据大 $O$ 表示法的意义,证明大 $O$ 表示法的加法法则。

**解**:首先给出如下加法法则:

若两个程序段的时间复杂性为 $T_1(n)=O(f_1(n))$和 $T_2(n)=O(f_2(n))$,那么这两个程序段依次执行时的时间复杂性如下:

$$T(n) = T_1(n) + T_2(n) = O(\max(f_1(n), f_2(n)))$$

上式可以证明如下。由于

$$T_1(n) = O(f_1(n)), \quad T_2(n) = O(f_2(n))$$

则存在常数 c 和 $n_0$,使得当 $n \geqslant n_0$时,有

$$T_1(n) + T_2(n) \leqslant c \times \max(f_1(n), f_2(n))$$

例如,　　$T_1(n)=O(n^2)$,　$T_2(n)=O(n^3)$

当 $n_0=1$ 时，对所有 $n \geqslant n_0$ 都有

$$n^3+n^2 \leqslant n^3+n^3=2n^3$$

当取 c=2 时，则有 $T(n) \leqslant c \times n^3$，取 $T(n)=O(n^3)$，所以

$$T(n)=T_1(n)+T_2(n)=O(\max(f_1(n),f_2(n))) \text{ 成立。}$$

3. 将数量级 $O(1)$，$O(n)$，$O(n^2)$，$O(n^3)$，$O(n\log_2 n)$，$O(\log_2 n)$，$O(2^n)$ 按增长率从小到大排列。

**解**：上面的几种类型的数量级中，$O(1)$ 为常量型，$O(n)$ 为线性型，$O(n^2)$ 为平方型，$O(n^3)$ 为立方型，$O(\log_2 n)$ 为对数型，$O(2^n)$ 为指数型，$O(n\log_2 n)$ 为线性对数型，它们按增长率从小到大的顺序为 $O(1)$，$O(\log_2 n)$，$O(n)$，$O(n\log_2 n)$，$O(n^2)$，$O(n^3)$，$O(2^n)$。

4. 猴子吃桃问题。猴子第 1 天摘下若干个桃子，当即吃了一半，还不过瘾，又多吃了一个；第 2 天早上又将剩下的桃子吃掉一半，又多吃了一个；以后每天早上都吃了前一天剩下的一半零一个，到第 10 天早上再吃时，见只剩下一个桃子了。求第 1 天共摘多少个桃子。

**解**：

```
main()
{ int day,x1,x2;
  day=9;
  x2=1;
  while(day>0)
  { x1=(x2+1)*2;
      x2=x1;
      day--;
  }
 printf("桃子总数=%d\n",x1);
}/*main*/
```

结果：桃子总数为 1534。

5. 求两个自然数，其和是 667，其最小公倍数和最大公约数之比是 120∶1。

**解**：设两个自然数分别为 $m$ 和 $667-m$（$2 \leqslant m \leqslant 333$），然后只要求这两个自然数的最小公倍数与最大公约数，判断是否满足条件即可。其程序如下：

```
main()
{ int gcd();
  int gbs();
  int m,n,k;
  for(m=2;m<=333;m++)
  { n=gbs(m,667-m);
    k=gcd(m,667-m);
    if ((n==120*k) && (n%k==0))
      printf("\n%3d %3d",m,667-m);
  }
}/*main*/
```

```
int gcd(a,b)
int a,b;
{ int i;
  for(i=a;i>=1;i--)
  if (! ((a%i)||(b%i)))
    return(i);
  }/* gcd */
  int gbs(a,b)
  int a,b;
  { int i;
    i=b;
    while((i%a!=0))  i+=b;
    return(i);
}/* lcm */
```

6. 哥德巴赫猜想的命题之一：大于 6 的偶数由两个素数组成。编写程序，验证 4～100 之间偶数由哪些素数组成。

**解**：基本思想是输入一个偶数 $t$，将 $t$ 分成两部分 $t1$ 和 $t2$，而且 $t1$ 初值为 1，然后 $t1$ 增加 1，直到 $t1$ 为素数为止；此时检查 $t2=t-t1$ 是否为素数，若不是，重复以上过程，直到 $t2$ 也为素数为止，$t$ 的值在 4～100 之间变化。其程序如下：

```
#include "math.h"
#define N 100
main()
{ int i=0,t,t1,t2;
  int prime();/* 判断素数函数 */
  t=4;
  while(t<=N)
  { t1=1;
    do
      { do
        { if (t1<3) t1++;
          else
            t1=t1+2;
        }while(! prime(t1));
       t2=t-t1;
       }while(! prime(t2));
     printf("%3d=%3d +%3d",t,t1,t2);
     i++;
     if (i % 4 ==0)  printf("\n") ;
     t=t+2;
     }
}/* main */
int prime(n) /* 判断素数函数 */
```

```
int n;
{ int flag=1,k,i;
  k=(int) (sqrt(n));
  for (i<=k;i=2;i++)
    if( n % i ==0)
      { flag=0;
        break;
        }
  if (i>=k+1) flag=1;
      else flag=0;
  return(flag);
}/* prime */
```

7. 分析下列程序段，求用大 $O$ 记号表示执行时间为 $n$ 的函数。

```
① i=1;k=0;
   while(i<=n-1)
   { k=k*10*i;
     i++;
   }
```

**解**：$T(n)=O(n)$。

```
② i=1;j=0;
   while((i+j)<=n)
   if (i>j) j++;
   else i++;
```

**解**：$T(n)=O(n)$。

```
③ x=91; y=100;
   while(y>0)
   if(x>0)
     {x=x-10;
      y=y-1;
       }
   else
       x=x+1;
```

**解**：本程序段是著名的 McCarthy 函数

$$M(x)=\begin{cases}x-10, & x>100\\ M(M(x+1)), & x\leqslant 100\end{cases}$$

对任何的 $x\leqslant 100$，$M(x)=91$，所以此程序段实质上是一个二重循环，对每个 $y(y>0)$ 值，if 语句执行 11 次，其中 10 次是执行 $x++$ 语句，但它们与 $n$ 无关，所以 $T(n)=O(1)$。

```
④ for(i=0;i<n;i++)
```

```
for(j=0;j<i;j++)
  for(k=0;k<j;k++)
    x=x+1;
```

**解**：$T(n)=O(n^3)$。

⑤
```
i=1;
do
  {j=1;
    do
    {printf("%d\n",i*j);
      j++;
      }while(j>n);
    i++;
    }while(i>n);
```

**解**：$T(n)=n(n+1)/2=O(n^2)$。

⑥
```
x=n;          /*n>1*/
y=0;
while((x>=(y+1)*(y+1))
      y=y+1;
```

**解**：$T(n)=O(\sqrt{n})$。

8. 按增长率由小到大的顺序排列下列各函数：$2^{100}$，$(3/2)^n$，$(2/3)^n$，$(4/3)^n$，$(n)^n$，$(n)^{2/3}$，$(n)^{3/2}$，$\sqrt{n}$，$n!$，$n$，$\log_2 n$，$n/\log_2 n$，$\log_2^2 n$，$\log_2(\log_2 n)$，$n\log_2 n$，$n^{\log_2 n}$。

**解**：各函数的排列次序如下，即$(2/3)^n$，$2^{100}$，$\log_2(\log_2 n)$，$\log_2 n$，$\log_2^2 n$，$\sqrt{n}$，$(n)^{2/3}$，$n$，$n\log_2 n$，$(n)^{3/2}$，$n/\log_2 n$，$(4/3)^n$，$(3/2)^n$，$n^{\log_2 n}$，$n!$，$(n)^n$。

9. 背包问题。有不同价值、不同重量的物品 $n$ 件，求从这 $n$ 件物品中选取一部分物品的选择方案，使选中物品的总重量不超过指定的限制重量，但选中物品的价值之和为最大。

**解**：可以按照以下算法编写递归和非递归程序。

```
【算法】背包问题，找最佳方案。
try(物品 i,当前选择已达到的重量和 tw,本方案可能达到的总价值 tv)
{ /*考虑物品 i 包含在当前方案的可能性*/
    if(包含物品 i 是可接受的)
     { 将物品 i 包含在当前方案中;
       if (i<n-1)
          try(i+1,tw+物品 i 的重量,tv);
       else /*又一个完整方案,因它比前面的方案好,以它作为最佳方案*/
     以当前方案作为临时最佳方案保存;
     恢复物品 i 不包含状态;
  }
```

```
/*考虑物品 i 不包含在当前方案的可能性*/
  if(不包含物品 i 仅是可考虑的)
    if (i<n-1)
       try(i+1,tw,tv-物品 i 的重量);
    else /*又一个完整方案,因它比前面的方案好,以它作为最佳方案*/
      以当前方案作为临时最佳方案保存;
}/* try */
```

也可以采用非递归的程序求解。可以通过依次考察每个物品形成所有可能的方案，即生成所有的候选解。对物品 i 的考察有如下几种情况：当该物品包括在候选解中依旧满足解的总重量的限制，该物品包括在候选解中是应该继续考虑的；反之，该物品不应该包括在当前正在形成的候选解中。同样，仅当物品不包括在候选解中，还有可能找到比目前临时最佳解更好的候选解时，才去考虑该物品不被包括在候选解中；反之，该物品不包括在当前候选解中的方案也不应继续考虑。对于任一值得继续考虑的方案，程序就去进一步考虑下一个物品。按照这样的思想可以编写出非递归程序。

CHAPTER 2

# 第2章 线 性 表

## 2.1 学习要点

(1) 理解线性表的逻辑结构特性,即数据元素之间存在某种线性关系。

(2) 掌握线性表的顺序存储结构和链式存储结构的描述方法。

(3) 掌握在线性表的顺序存储结构和链式存储结构上的基本运算及其实现。

(4) 理解循环链表结构的概念和遍历循环链表、循环链表的合并等运算。

(5) 理解双向链表结构的概念、双向链表中元素的插入、删除和定位方法。

(6) 了解线性表的顺序存储结构和链式存储结构的不同点以及应用,能对线性表的基本运算,如插入、删除等运算的时间复杂度进行估计。

## 2.2 主要内容及知识点

### 1. 线性表的逻辑结构定义、特点及抽象数据类型定义

线性表是由若干个数据元素组成的有限序列。线性表的特点是具有惟一的起始结点和惟一的终端结点,除起始结点外的每个结点都有惟一的直接前驱,除终端结点外的每个结点都有惟一的直接后继。

线性表的抽象数据类型定义(ADT List):数据对象、关系的定义以及与线性表相关操作的定义。线性表的数据对象是具有相同性质的数据元素集合。与线性表相关的基本运算常见的有置空表,求表长,取元素,定位,求前驱、后继、检索、插入、删除以及遍历线性表等。

### 2. 存储结构的描述方法及操作实现

顺序表是在一组地址连续的存储单元中依次存放线性表中的数据元素,通常用数组来描述顺序表。链表是用一组任意存储单元存储线性表的数据元素(存储单元间可以是连续的,也可以是不连续的),在实现时结点通常由 2 个域构成,一个存放数据,另一个存放后继结点的地址。循环链表是一种特殊的链表,它的特点是表中最后一个结点的指针域指向头结点,整个链表形成环状结构。双向链表中结点的指针域有 2 个,一个指针域指向后继,另一个指针域指向前趋。与循环链表类似,依照双向链表建立起来的环形结构称为双向循环链表。在线性表的顺序与链接两种存储结构下有取表长、判表空、取前驱、取