第3章 chapter 3

简单数据结构

本章介绍顺序表、链表、栈、队列和广义表等一些简单的数据结构的基本概念、存储结构及其运算，同时对章后习题做了较详细的解答。

3.1 基本知识点

1. 顺序表

线性表(Linear List)是一种最简单最常用的数据结构。一个线性表是由 $n(n\geqslant 0)$ 个相同类型数据元素(结点)组成的有限序列。表中有且仅有一个第一个结点，它没有前趋而只有一个后继；有且仅有一个最后一个结点，它没有后继而只有一个前趋；其余结点都只有一个前趋和一个后继。

线性表的形式化定义如下：

$$\text{Linear List}=(D,R)$$

其中：$D=\{a_i \mid a_i\in D_o, i=1,2,\cdots,n, n\geqslant 0\}$；$R=\{<a_{i-1},a_i> \mid a_{i-1},a_i\in D_o, i=2,3,\cdots,n\}$；$D_o$ 为某个数据对象。

线性表的基本运算：置空表 setnull(L)、求长度 length(L)、取元素 get(L,i)、取前趋 prior(L,x)、取后继 next(L,x)、定位序 locate(L,x)、插入 insert(L,x,i)和删除 delete(L,i)运算。利用这些基本运算，可以实现线性表的其他较复杂运算，如线性表的分拆、合并和逆置等。

顺序表(Sequential List)是用一组地址连续的存储单元依次存放线性表中的各个数据元素，是一种最简单最自然的线性表存储方法。换句话说，顺序表以数据元素在计算机内的物理位置相邻来表示数据元素在线性表中的逻辑相邻关系。

顺序表有两个优点：

(1) 无须为表示数据元素之间的关系而额外增加存储空间。

(2) 可以随机存取表中任一数据元素，元素存储位置可以用一个简单、直观的公式表示。

顺序表也有两个缺点：

(1) 插入和删除运算必须移动大量(几乎一半)数据元素，效率较低。

(2) 必须预先分配存储空间,造成空间利用率低,且表的容量难以扩充。

2. 链表

线性表的链式存储结构,是用一组任意的存储单元(这组存储单元的地址可以连续,也可以不连续)来存放线性表中的各个数据元素的。存放数据元素信息的域 data 称作数据域,存放后继元素地址信息的域称作指针域或链域。一个线性表的 n 个元素通过每个结点的指针域拉成一条"链子",所以称为链表(Linked List)。由于这种链表中每个结点只含有一个指向后继的指针,所以称作线性链表或单链表(Single Linked List)。

单链表上的基本运算有建立单链表、求表长、元素的查找、插入和删除等。

循环链表(Circular Linked List)是链式存储结构的另一种形式,其特点是表中最后一个结点的指针域指向头结点,使整个链表构成一个环,可以从表中任一结点出发访问遍所有结点(即遍历)。在单循环链表中,虽然可以从任一已知结点出发找到其前趋结点,但需耗时 O(n),原因在于每个结点只有一个指向后继的指针域。如果希望能够快速确定任一结点的前趋,就必须付出空间代价,即增加一个指针域存储其前趋信息,这样每个结点就有两个方向不同的链,称之为双向链表(Double Linked List)。如果让每条链都构成一个环,则称为双向循环链表(Double Circular Linked List)。

双向链表是一种对称结构,每个结点都既有指向其前趋的指针,也有指向其后继的指针;每个结点的地址既放在其后继结点的前趋域中,也放在其前趋结点的后继域中。

链式存储结构(链表)克服了顺序表的两个缺点,但是它不具备顺序表的两个优点。

3. 栈

栈(Stack)是操作受限的线性表,限定对元素的插入和删除运算只能在表的一端进行。通常把进行插入和删除操作的这一端称作栈顶(Top),另一端称作栈底(Bottom);把栈的插入元素操作称作进栈、入栈或压入,栈的删除元素操作称作退栈、出栈或弹出;当栈中没有元素时称作空栈。由栈的定义可知,每一次入栈的元素都在原栈顶元素之上成为新的栈顶元素,每一次出栈的元素总是当前栈顶元素使次栈顶元素成为新的栈顶元素,即最后进栈的元素总是最先出栈。所以栈也称作后进先出(Last In First Out)表,简称 LIFO 表。

栈的基本运算有 5 种:置空栈 setnull(s)、判栈空 empty(s)、进栈 push(s,x)、出栈 pop(s)和读栈顶元素 gettop(s)运算。

利用顺序存储结构实现的栈称作顺序栈(Sequential Stack)。在栈满时做入栈操作会产生空间不足的情况,简称上溢(Overflow);而在栈空时做出栈操作会产生无元素可出的情况,简称下溢(Underflow)。

利用链式存储结构实现的栈称作链栈(Link Stack)。链栈中的每个数据元素用一个结点表示,其结构形式与单链表完全相同。链栈从本质上讲就是单链表,无非是限制了插入和删除运算只能在链头进行,所以可以说链栈就是限制插入和删除运算只能在链头进行的单链表。

栈有着相当广泛的应用,重要的典型应用是实现函数和过程的递归调用。

4. 队列

队列(Queue)也是一种操作受限的线性表。和栈不同的是,队列是限定所有的插入操作只能在表的一端进行,而所有的删除操作都只能在表的另一端进行。允许插入的一端叫做队尾(Rear),允许删除的一端叫做队头(Front)。新插入的元素只能添加到队尾,被删除的只能是排在队头的元素。先进入队列的元素总是先离开队列,或者说队列的操作是按先进先出的原则进行的。因此,队列也称作先进先出(First In First Out,FIFO)的线性表。

通常对队列可施行的基本运算有 5 种:初始化队列 iniqueue(q)、入队列 addqueue(q,x)、出队列 outqueue(q)、读队头元素 gethead(q)和判队列空 empty(q)运算。

队列的顺序存储结构实现称作顺序队列。与顺序表一样,顺序队列也是用一维数组来存储其数据元素的。

队列的链式存储结构实现称作链队列。与链栈类似,链队列实质上也是单链表。

队列广泛应用于操作系统中的 I/O 缓冲区管理、CPU 分时管理、优先队列及双端队列等方面。

5. 广义表

广义表(generalized list)又称作列表,是线性表的一种推广。它打破了线性表中的数据元素只能是一个数或一个记录的限制,容许元素本身又是结构,其定义如下:一个广义表是 $n(n\geqslant 0)$个元素 $d_1,d_2,\cdots,d_n$ 的有限序列,其中 d_i 或者是一个数据元素,或者是一个广义表。通常记作 LS=$(d_1,d_2,\cdots,d_n)$,LS 是广义表的名字,n 是广义表的长度。d_i 为单个数据元素时称为广义表 LS 的单元素,其结构上不可再分割,是某一确定类型的原子项;d_i 为广义表时称为广义表 LS 的子表,它又是由若干元素构成的一个广义表。习惯上常用小写字母表示单元素,用大写字母表示广义表的名字。

由广义表的定义可得到广义表如下 3 条重要性质:

(1) 广义表是一种多层次数据结构。

(2) 广义表具有递归性质。

(3) 广义表具有共享性质。

广义表的基本运算包括求长度 len(LS)、求深度 depth(LS)、求表头 head(LS)和求表尾 tail(LS)等运算。

广义表的结点结构可以模仿单链表的结点结构。由于广义表中元素有单元素和子表之分,其结点结构也应相应区分。单元素结点应包括值域和指向其后继的指针域,子表结点应包括指向表头结点的指针域和指向其后继的指针域,可通过设标志域来区分单元素结点时的值和子表结点时的指向表头结点的指针。除了这种存储方式外,广义表还有其他一些形式的链式存储结构。如双链表示法,其中一个链指向第一个元素(表头)结点;另一个链指向表尾。

3.2 习题解答

1. 线性表可用顺序表和单链表作为存储结构。试问：

(1) 两种存储表示各有哪些主要优缺点？

(2) 如果有 n 个表同时并存，且处理过程中各表的长度会动态发生变化，表的总数也可能自动改变，在此情况下应选用哪种存储表示？为什么？

(3) 若表的总数基本稳定，且很少进行插入和删除，但要求以最快速度存取表中元素，这时应采用哪种存储表示？为什么？

解答：

(1) 线性表具有两种存储结构，即顺序存储结构和链式存储结构。虽然线性表的顺序存储结构可以直接存取数据元素，方便灵活，效率高，但插入、删除操作时将会引起元素的大量移动，因而降低效率；在链接存储结构中内存采用动态分配，利用率高，但需增设指示结点之间关系的指针域，存取数据元素不如顺序存储方便，但结点的插入、删除操作较简单。

(2) 应选用链接存储结构，因为链式存储结构是用一组任意的存储单元依次存储线性表中的各元素，这里存储单元可以是连续的，也可以是不连续的；这种存储结构对于元素的删除或插入运算是不需要移动元素的，只需修改指针即可，所以很容易实现线性表容量的扩充。

(3) 应选用顺序存储结构，因为顺序存储用一组地址连续的存储单元依次存放线性表中的各个数据元素，只要确定了其起始位置，线性表中的任意一个数据元素的位置就可以确定，就可以随机访问。因此，线性表的顺序存储结构是一种随机存取的存储结构，链表则是一种顺序存取的存储结构。

2. 试设计一个算法，它在带表头结点的单链表中寻找第 i 个结点，若找到则返回第 i 个结点的地址，否则返回空地址(NULL)。

解答：

```
#include "stdio.h"
typedef int elemtype;
typedef struct node
  { elemtype data;
    struct node *next;
    }linklist;
linklist *creatlinklist()                    /*建立带表头结点的单链表*/
 {int x;
  linklist *head,*r,*p;
  head=(linklist *)malloc(sizeof(linklist));
  head->next=NULL;
  r=head;
  scanf("%d",&x);
```

```
  while(x!=-1)
    {p=(linklist *)malloc(sizeof(linklist));
     p->data=x;p->next=NULL;
     r->next=p;
     r=r->next;
     scanf("%d",&x);}
     return head;
  }                                          /* End of creatlinklist */
/* 查找第 i 个结点,返回该结点的存储位置或者 NULL */
linklist *Get(linklist *h,int i)
 {int j;
  linklist *p;
  p=h;j=0;                                   /* 从头结点起扫描,计数器 j 的初值为 0 */
  while((p->next!=NULL)&&(i>j))
    {p=p->next;                              /* 指针右移,扫描下一个结点 */
    j++;                                     /* 统计扫描过的结点总数 */
    }
  if(j==i)
      return p;                              /* 返回找到的第 i 个结点的地址 */
  else
      return NULL;                           /* 未找到,返回 NULL */
 }                                           /* End of Get */
main()
 {linklist *head,*p;
  int i;
  printf("\nPlease input the data:");
  head=creatlinklist();
  printf("\nPlease input the i:");
  scanf("%d",&i);
  p=Get(head,i);
  if(p!=NULL)
     printf("The %dth data is %d.",i,p->data);
  else
     printf("No find!");
 }
```

3. 设 ha 和 hb 分别是两个带表头结点的升序单链表的表头指针。试设计一个算法将这两个链表合并成为一个降序单链表。要求结果链表仍使用原来两个链表的结点空间而不另开辟其他存储空间,表中允许出现重复数据。

解答:

```
#include "stdio.h"
typedef int elemtype;
typedef struct node
```

```
  { elemtype data;
    struct node * next;
    }linklist;
linklist * creatlinklist()                    /* 建立带表头的链表 */
{ int x;
  linklist * head, * r, * p;
  head= (linklist * )malloc(sizeof(linklist));
  head->next=NULL;
  r=head;
  scanf("%d",&x);
  while(x!=-1)                                /* -1 为结束标志 */
    {p= (linklist * )malloc(sizeof(linklist));
     p->data=x;p->next=NULL;
     r->next=p;
     r=r->next;
     scanf("%d",&x);
     }
  return head;
  }                                           /* End of creatlinklist */
/* 将这两个升序链表合并成为一个降序单链表 */
linklist * connect(linklist * ha,linklist * hb)
{ linklist * pa=ha->next;                     /* 把 ha 第 1 个结点赋给 pa */
  linklist * pb=hb->next;                     /* 把 hb 第 1 个结点赋给 pb */
  linklist * hc, * p;
  hc=NULL;                                  /* pc 总是指向生成的新单链表的第一个结点 */
  while(pa!=NULL&&pb!=NULL)
    if(pb->data>pa->data)                     /* 若 pa 更小,链接到 hc */
       {p=pa;
        pa=pa->next;
        p->next=hc;
        hc=p;
        }
    else
      if(pb->data<pa->data)                   /* 若 pb 更小,链接到 hc */
        {p=pb;
         pb=pb->next;
         p->next=hc;
         hc=p;
         }
      else                    /* 在 pa->data==pb->data 时,这两个结点都链到 hc 中 */
        {p=pa;
         pa=pa->next;
         p->next=hc;
         hc=p;
```

```
        p=pb;
        pb=pb->next;
        p->next=hc;
        hc=p;
        }
  while(pa!=NULL)                          /* pa 单链表还有结点时 */
     {p=pa;
      pa=pa->next;
      p->next=hc;
      hc=p;
      }
  while(pb!=NULL)                          /* pb 单链表还有结点时 */
     {p=pb;
      pb=pb->next;
      p->next=hc;
      hc=p;
      }
  ha->next=hc;                            /* 把 ha 的头结点作为链表合并后的头结点 */
  return ha;
  }                                        /* End of connect */
print(linklist *h)                         /* 输出链表中的数据 */
  {h=h->next;
   while (h!=NULL)
      {printf(" %d  ",h->data);
       h=h->next;}
   printf("\n");
   }                                       /* End of print */
main()
{ linklist *ha,*hb,*head;
  printf("\nPlease input the data of ha:");
  ha=creatlinklist();
  printf("\nPlease input the data of hb:");
  hb=creatlinklist();
  head=connect(ha,hb);
  print(head);
}                                          /* End of main */
```

4. 设计一个算法,它通过一趟遍历在单链表中确定值最大的结点。

解答:

【解题思路】 以 p 遍历单链表,在遍历时用 q 指向 data 域值最大的结点,最后返回 q。

【程序实现】

```
#include "stdio.h"
```

```
typedef int elemtype;
typedef struct node
  {elemtype data;
   struct node * next;
  }linklist;
linklist * creatlinklist()                    /* 建立带表头的链表 */
 {int x;
  linklist * head, * r, * p;
  head=(linklist * )malloc(sizeof(linklist));
  head->next=NULL;
  r=head;
  scanf("%d",&x);
  while(x!=-1)                                /* -1 为结束标志 */
    {p=(linklist * )malloc(sizeof(linklist));
     p->data=x;p->next=NULL;
     r->next=p;
     r=r->next;
     scanf("%d",&x);
    }
  return head;
  }                                           /* End of creatlinklist */
linklist * maxnode(linklist * h)              /* 遍历在单链表中确定值最大的结点 */
  {linklist * q, * p;
   p=h->next;
   q=p;
   while(p!=NULL)
     {if(p->data>q->data)
         q=p;
      p=p->next;
     }
   return q;
  }                                           /* End of maxnode */
main()
 {linklist * head, * h;
  printf("\nPlease input the data:");
  head=creatlinklist();
  h=maxnode(head);
  printf("Max is %d",h->data);
 }                                            /* End of main */
```

5. 设计一个算法,它在非递减有序的单链表中删除值相同的多余结点。

解答:

【解题思路】 设单链表(类型为 linklist)的头指针 head 指向头结点,则可按下列步骤执行:

首先,用一个指针 p 指向单链表中第一个表结点;然后用另一个指针 q 检索链表中的结点元素,由于是单链表,故约束条件为 p==NULL,同时让指针 pre 指向 q 所指结点的前驱结点,当检索到结点具有 q->data==p->data 时删除 q 所指的结点;然后再修改 p,使 p 指针后移(即 p=p->next),重复进行,直到 p 为空时为止。

【程序实现】

```
#include "stdio.h"
typedef int elemtype;
typedef struct node
  {elemtype data;
   struct node * next;
   }linklist;
linklist * creatlinklist()                    /* 建立带表头的链表 */
 {int x;
  linklist * head, * r, * p;
  head=(linklist * )malloc(sizeof(linklist));
  head->next=NULL;
  r=head;
  scanf("%d",&x);
  while(x!=-1)                                /* -1 为结束标志 */
    {p=(linklist * )malloc(sizeof(linklist));
     p->data=x;p->next=NULL;
     r->next=p;
     r=r->next;
     scanf("%d",&x);
    }
  return head;
  }                                           /* End of creatlinklist */
delete(linklist * head)                       /* 删除单链表中值相同的多余结点 */
 {linklist * p, * pre, * q;
  p=head->next;
  q=p;
  while(p!=NULL)
    {pre=q;
     q=q->next;
     do
       {while((q!=NULL)&&(q->data!=p->data))   /* 检索值相同的结点 */
         {pre=q;
          q=q->next;
          }
        if(q!=NULL)                           /* 删除值相同的结点 */
          {pre->next=q->next;
           free(q);
           q=pre->next;
```

```
        }
      }while(q!=NULL);
    p=p->next;
    q=p;
    }
  }
main()
 {linklist *head,*p;
  printf("\nPlease input the data:");
  head=creatlinklist();
  delete(head);
  p=head->next;
  while(p!=NULL)
    {printf("%d ",p->data);
     p=p->next;
    }
  }
```

6. 设有一个线性表 $L=(a_1,a_2,\cdots,a_n)$，试分别在顺序表和单链表两种存储表示方式下，各设计一个将线性表 L 逆置的算法，要求不重新开辟存储空间。所谓逆置是指将线性表中的元素次序颠倒过来，即成为 $L'=(a_n,a_{n-1},\cdots,a_1)$。

解答：

(1) 单链表逆置。

【解题思路】 依次从原链表中取出每个结点，每次都把它作为第一个结点插入到新链表中去。为此要借用两个指针变量 p 和 q，p 用来指向原链表中的当前第一个结点，q 用来指向从原表取出的每一个结点并利用它插入到新链表中去，当 p 为空时完成逆置。

【程序实现】

```
#include "stdio.h"
typedef int elemtype;
typedef struct node
  {elemtype data;
   struct node *next;
   }linklist;
linklist *creatlinklist()                    /*建立带表头的链表*/
 {int x;
  linklist *head,*r,*p;
  head=(linklist *)malloc(sizeof(linklist));
  head->next=NULL;
  r=head;
  scanf("%d",&x);
  while(x!=-1)                               /* -1为结束标志*/
    {p=(linklist *)malloc(sizeof(linklist));
     p->data=x;p->next=NULL;
```