**第二章 线性表**

**§2.1 线性表的定义**

**§2.1.1 线性表概述**

**线性表顾名思义为线性结构，数据元素之间呈一种线性关系。**

**线性关系的特点是：**

**●所有数据元素的类型是相同的，元素一个接一个的排列；**

**●有唯一的“头元素”和唯一的“尾元素”，“头元素”没有前驱，而“尾元素”没有后继。其它元素均有一个直接的前驱和一个直接的后继。**

**简言之：所谓线性关系，元素的前驱和后继分别呈现0或1状态。**

**线性表中的数据元素的个数n称为表长。当n=0时，称为空表。**

**当n>0时，线性表通常表示为：L=，其中元素的值可以不同，但类型必须相同。例如：**

1. **数据元素是数字，表长为10的线性表：**

**(0,1,2,3,4,5,6,7,8,9)**

**（2）数据元素是大写英文字母，表长为26的线性表：（A,B,C,D,……,Z）**

1. **学生登记表，每个数据元素（通常称为记录，**

**将其中的学号、姓名、性别、年龄和专业称为数据项），共有5个学生，即表长为5。**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **学号** | **姓名** | **性别** | **年龄** | **专业** |
| **01** | **汤姆** | **男** | **18** | **计算机** |
| **02** | **杰克** | **女** | **19** | **英语** |
| **03** | **玛丽** | **女** | **20** | **法律** |
| **04** | **彼德** | **男** | **22** | **电子工程** |
| **05** | **大山** | **男** | **21** | **企业管理** |

**（4）一家有5个孩子，数据元素是各孩子，表长为5的线行表：（老大<女>，老二，老三，老四，老五<女>）**

**§2.1.2 线性表的抽象数据结构**

**数据结构的运算是建立在逻辑结构层次上的，而运算的具体实现是建立在存储结构上的。**

**线性表的抽象数据结构定义为：**

**ADT Linear\_List**

**{ 数据对象：任意数据元素的集合D={|任意数据元素}**

**数据关系：除头元素和尾元素外，其它元素均有一个直接的**

**前驱和一个直接的后继**

**基本操作：**

**ListInit(L) //初始化操作，构造一个空表**

**ListLength(L) //求表长**

**Listget(L,i) //取元素**

**ListLocate(L,x) //查找元素**

**ListPrior(L,e) //求元素的前驱**

**Listnext(L,e) //求元素的后继**

**ListInsert(L,i,e) //前插元素**

**ListDelete(L,i) //删除元素**

**ListEmpty(L) //判空表**

**ListClear(L) //清空表**

**}ADT Linear\_List**

**说明：以上操作只是表的基本操作，绝不是表的全部操作。**

**§2.2 线性表的顺序存储结构**

**数据结构在内存中的存储通常有两种：顺序存储（又称顺序表）和链式存储（又称链表）。**

**§2.2.1 顺序表**

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0011** | **0013** | **…**  **…** |  |  |  | **…** |  | **…** | **…** |  |

**特点：各元素用一块地址连续的存储空间，逻辑上相邻的元素物理存储上也相邻。**

**地址的计算公式（第i个元素的地址）：**

**Loc()=Loc()+(i-1)d**

**其中：1≤i≤n,d是每个元素所占的地址大小（通常为字节数）**

**如：设Loc(a1)的起始地址是0011，d=2（字节）**

**Loc(a2)=Loc(a1)+(2-1)d=Loc(a1)+d=0011+2=0013**

**在C语言中，对顺序表显然用数组存储比较适宜。**

**用C语言定义顺序存储结构如下：**

**#define MAXSIZE 100 //选择足够大的空间**

**typedef int ElemType;**

**//ElemType是int类型的一个别名**

**typedef struct node**

**{**

**ElemType data[MAXSIZE];//存储各表元素**

**int length; //表长**

**}SeqList; //顺序表的类型**

**SeqList L;//L为顺序表（结构体变量）**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **元素排序** | **1** | **2** | **……** | **i** | **……** |
| **元素存储** | **L.data[0]** | **L.data[1]** | **……** | **L.data[i-1]** | **……** |

**注意：元素次序为：1,2,3,……，n**

**C语言中数组元素的下标次序为：0,1,2,……,n-1**

**●若用L（结构体变量）表示：**

**第i个元素自身：****L.data[i-1]**

**前驱是：L.data[i-2]**

**后继是：L.data[i]**

**表长：L.length**

**§2.2.2 顺序表的基本操作**

**1．顺序表的初始化**

**void SeqListInit(SeqList L)**

**{**

**L.length=0; //将顺序表长度置0**

**}**

**2．顺序表求长度**

**int SeqListLength(SeqList L)**

**{**

**return(L.length); //返回顺序表的长度**

**}**

**时间复杂度：O(1)**

**3．顺序表取元素**

**ElemType SeqListGet(SeqList L,int i)**

**{**

**if(i>=1&&i<=L.length) //i值是否合法**

**return(L.data[i-1]); //返回该元素**

**else {printf(“i值不合法”);exit(0);}**

**}**

**时间复杂度：O(1)**

**4．顺序表元素的定位操作**

**int SeqListLocate(SeqList L,ElemType e)**

**{**

**i=1;**

**while(i<=L.length&&e!=L.data[i-1])i++;**

**if(i<=L.length)return(i);**

**else {printf(“此元素不在表中”);exit(0);}**

**}**

**时间复杂度：O(n)**

**5. 顺序表求前驱操作**

**ElemType SeqListPrior(SeqList L,ElemType e)**

**{**

**i=SeqListLocate(L,e);**

**if(i==1){printf(“第一个元素没有前驱”);**

**exit(0);}**

**else return(L.data[i-2****]);//返回该元素的前驱**

**}**

**时间复杂度：O(1)**

**6. 顺序表求后继操作**

**ElemType SeqListPrior(SeqList L,ElemType e)**

**{**

**i=SeqListLocate(L,e);**

**if(i==L.length)**

**{printf(“最后一个元素没有后继”);**

**exit(0);}**

**else return(L.data[i]);//返回该元素的后继**

**}**

**时间复杂度：O(1)**

**7.顺序表的前插操作**

**在表的第i的位置之前插入一个值为b的新元素，使表长变为L.length+1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **下标** | **0** | **1** | **…** | **i-2** | **i-1** | **i** | **i+1** | **…** |
| **插入前** |  |  | **…** |  |  |  | **…** | **…** |
| **准备** |  |  | **…** |  |  |  |  | **…** |
| **插入后** |  |  | **…** |  | **b** |  |  | **…** |

**操作步骤：**

**(1)检查插入要求的合理性**

**(2)将至顺序向后挫动，为新元素b让出位置**

**(3)将b插入空出的第i的位置**

**(4)修改表长的值L.length+1**

**void SeqListInsert(SequList L,int i,ElemType b)**

**{**

**int j;**

**if(L.length==MAXSIZE)**

**{printf("表满，无法插入");exit(0);}**

**if(i<1||i>L.length)**

**{printf("插入位置i非法");exit(0);}**

**for(j=L.length-1;j>=i-1;j--)**

**L.data[j+1]=L.data[j]; // 元素后移**

**L.data[i-1]=b; // 在第i个位置上插入b**

**L.length++; // 表长加1**

**}**

**●时间复杂度分析：在第i个位置上插入，需要向后移动元素，共n-i+1个元素，设插入的概率，依等概率计算，平均移动次数E:**

****

**时间复杂度为O(n)。**

**8.顺序表的删除**

**将表的第i个元素从表中删除，使原表长L.length-1**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **下标** | **0** | **1** | **…** | **i-2** | **i-1** | **i** | **i+1** | **…** |
| **删除前** |  |  | **…** |  |  |  | **…** | **…** |
| **准备** |  |  | **…** |  |  |  | **…** | **…** |
| **删除后** |  |  | **…** |  |  |  | **…** | **…** |

**删除步骤如下：**

**(1) 检查删除要求的合理性**

**(2)将至顺序向前移动挤掉（删除）**

**(3) 修改表长的值L.length-1。**

**void SeqListDelete(SeqList L,int i)**

**{**

**int j;**

**if(i<1||i>L.length)**

**{printf("删除位置i非法");exit(0);}**

**for(j=i;j<=L.length-1;j++)**

**L.data[j-1]=L.data[j]; // 元素前移**

**L.length--; // 表长减1**

**}**

**时间复复杂度分析：在第i个位置上删除，需要向前移动元素，共n-i个元素，设删除的概率，依等概率计算，平均移动次数E:**

****

**时间复杂度为O(n)。**

**9.顺序表判空操作**

**int SeqListEmpty(SeqList L)**

**{**

**return(!L.length);//L.length==0**

**}**

**10．顺序表的遍历（visit,访问，周游，走）**

**对结构体中数组的元素遍历并输出。**

**void SeqListTraverse(SeqList L)//遍历**

**{**

**int i;**

**if(SeqListEmpty(L))printf(“空表”);**

**// SeqListEmpty(L)==1**

**else for(i=1;i<=L.length;i++)**

**printf("%5d",L.data[i-1]);**

**}**

**11．顺序表的创建**

**void SeqListcreat(SeqList L) // 创建**

**{**

**int i=0;**

**ElemType x;**

**printf("创建顺序表,输入若干整数,-1作为结束: ");**

**scanf("%d",&x);**

**while(x!=-1)**

**{**

**L.data[i]=x;**

**scanf("%d",&x);**

**i++;**

**}**

**L.length=i; // 记录数据个数（即表长）**

**}**

**P2-1.c顺序表的创建、遍历插入和删除**

**#define MAXSIZE 100**

**#include<stdio.h>**

**typedef int ElemType;**

**typedef struct**

**{**

**ElemType data[MAXSIZE];**

**int length;**

**}SeqList;**

**SeqList L;**

**int empty(void) //判空表**

**{**

**return(!L.length);**

**}**

**void creat(void) // 创建**

**{**

**int i=0;**

**ElemType x;**

**printf("创建顺序表,输入若干整数,-1作为结束: ");**

**scanf("%d",&x);**

**while(x!=-1)**

**{**

**L.data[i]=x;**

**scanf("%d",&x);**

**i++;**

**}**

**L.length=i; // 记录数据个数（即表长）**

**}**

**void visit(void)//遍历**

**{**

**int i;**

**if(empty())printf("空表");**

**else for(i=1;i<=L.length;i++)printf("%5d",L.data[i-1]);**

**printf("\n表长是：%d\n\n",L.length);**

**}**

**void insert(int i,ElemType b)//插入**

**{**

**int j;**

**if(L.length==MAXSIZE)printf("表满，无法插入");**

**if(i<1||i>L.length)**

**{printf("插入位置i非法\n");exit(0);}**

**for(j=L.length-1;j>=i-1;j--)L.data[j+1]=L.data[j];**

**// 元素后移**

**L.data[i-1]=b; // 在第i个位置上插入b**

**L.length++; // 表长加1**

**}**

**void deletes(int i)//删除**

**{**

**int j;**

**if(i<1||i>L.length)**

**{printf("删除位置i非法\n");exit(0);}**

**for(j=i;j<=L.length-1;j++)L.data[j-1]=L.data[j];**

**// 元素前移**

**L.length--; // 表长减1**

**}**

**void main()**

**{**

**int i;ElemType x;**

**creat();//创建**

**printf("创建后的顺序表L: ");visit();//创建后的遍历**

**printf("输入插入位置int i和数据int x: ");**

**scanf("%d%d",&i,&x);**

**insert(i,x);//插入**

**printf("插入后的顺序表L: ");visit();//插入后的遍历**

**printf("输入删除位置int i: ");scanf("%d",&i);**

**deletes(i);//删除**

**printf("删除后的顺序表L: ");visit();//删除后的遍历**

**}**

**p2-2.c已知顺序表的数据元素递增有序，在表中插入一个数据后仍保持顺序表有序。**

**// 输入若干数据，先变为递增有序顺序表，再在表中插入一个元素，并仍保持递增有序。**

**#define MAXSIZE 100**

**#include<stdio.h>**

**typedef int DataType;**

**typedef struct node**

**{**

**DataType data[MAXSIZE+1];**

**int last;**

**}SequList;**

**void Creat(SequList \*a) //创建顺序表**

**{**

**DataType x;**

**int i=0;**

**printf("创建顺序表,输入若干整数(int),**

**-1作为结束: ");**

**scanf("%d",&x); // 输入第一个数据**

**while(x!=-1)**

**{**

**i++;**

**a->data[i]=x;**

**scanf("%d",&x); // 输入下一个数据**

**}**

**a->last=i; // 记录数据个数（即表长）**

**}**

**void Sort(SequList \*a) //顺序表排序**

**{**

**int i,j;**

**DataType temp;**

**for(i=1;i<a->last;i++)//选择排序**

**for(j=i+1;j<=a->last;j++)**

**if(a->data[i]>a->data[j])**

**temp=a->data[i],**

**a->data[i]=a->data[j],**

**a->data[j]=temp;**

**}**

**void OrdInsert(SequList \*a,DataType value)//插入一个元素**

**{**

**int i,pos=1;//从第一个元素开始**

**a->data[a->last+1]=value;//设置监视哨**

**while(value>a->data[pos])pos++;**

**//查找插入位置**

**for(i=a->last;i>=pos;i--)**

**a->data[i+1]=a->data[i];//数据元素移动**

**a->data[pos]=value;//插入数据**

**a->last++;//修改表长**

**}**

**void Visit(SequList L)//遍历顺序表**

**{**

**int i;**

**for(i=1;i<=L.last;i++)**

**printf("%5d",L.data[i]);**

**printf("\n\n");**

**}**

**void main()**

**{**

**DataType value;**

**SequList L;**

**Creat(&L);//创建顺序表**

**printf("创建后的顺序表:");Visit(L);**

**Sort(&L);**

**printf("排序后的顺序表:");Visit(L);**

**printf("输入要插入的数据int value: ");**

**scanf("%d",&value);**

**OrdInsert(&L,value);//插入**

**printf("插入后的链表:");**

**Visit(L);//插入后的遍历**

**}**

**●对线性表的顺序存储的评价：**

**优点：元素的位置容易确定（依据数组的下标）。所以生成和遍历可以顺序（或倒序）和随机进行。**

**缺点：由于逻辑相邻的元素物理存储上也相邻，所以需要一片连续的存储单元；插入和删除会引起前后大量元素的移动，又由于通常是用数组来存放，而数组事先要确定长度（静态）。如果数组长度确定过小，插入时容易造成溢出；如果数组长度确定过大，删除过多的元素造成存储浪费。**

**这种机制有点像我们以前走过的“计划经济”过程。**

**下面介绍的链式存储可以解决这种不足。从“计划经济”转变为“市场经济”。**