**第二章作业**

**第一题（2.1）若将顺序表中记录其长度的分量 listlen 改为指向最后一个元素的位置 last，在实现 各基本运算时需要做那些修改？**

在插入元素后的时候把last++;

判空的时候last为-1;

判满时候last为Maxlen-1;

删除元素时候last--;

**第二题（2.7）假设顺序表L中的元素按从小到大的次序排列，设计算法以删除表中重复的元素, 并 要求时间尽可能少。要求：**

**算法描述：在循环中每次都判断当前元素与下一个元素是否相等，若不相等，则用一个新的顺序表存储当前元素。**

**（1）对顺序表（1,1,2,2,2,3,4,5,5,5,6,6,7,7,8,8,8,9）模拟执行本算法，并统计移动元素的 次数。**

**#include<iostream>**

**using namespace std;**

**int main(){**

**int a[100]={1,1,2,2,2,3,4,5,5,5,6,6,7,7,8,8,8,9};**

**int list[100];**

**int j=0;**

**for(int i=0;a[i]!=0;i++)**

**if(a[i]!=a[i+1]){**

**list[j]=a[i];**

**j++;**

**}**

**cout<<"删除重复元素后,顺序表元素为："<<endl;**

**for(int i=0;list[i]!=0;i++)**

**cout<<list[i]<<" ";**

**cout<<endl;**

**}**

**移动元素的次数为：17**

**（2）分析算法的时间性能。**

**算法的时间性能为：O(n) n为数组的长度**

**第二题（2.21）设计算法将链表 L 就地逆置，即利用原表各结点的空间实现逆置。**

**算法描述：对面链表的每一个节点，都采用头插法**

**#include<iostream>**

**using namespace std;**

**struct node{**

**int data;**

**node \*next;**

**};**

**int main(){**

**node \*head=new node;**

**node \*p=head;**

**for(int i=0;i<10;i++){**

**node \*new\_node=new node;**

**new\_node->data=i;**

**new\_node->next=NULL;**

**p->next=new\_node;**

**p=p->next;**

**}**

**p=head->next->next;**

**head->next->next=NULL;**

**node \*p1;**

**while(p){**

**p1=p->next;**

**p->next=head->next;**

**head->next=p;**

**p=p1;**

**}**

**p=head->next;**

**while(p){**

**cout<<p->data<<" ";**

**p=p->next;**

**}**

**}**

**第二题（2.23）设计算法以判断带头结点的双循环链表 L 是否是对称的，即从前往后和从后往前的 输出序列是相同的。若对称，返回 1，否则返回 0。**

**算法描述：用后向指针和前向指针每次都输出当前节点的元素，判断是否相等，若不相等，则不对称，若全部相等，则堆成**

**#include<iostream>**

**using namespace std;**

**struct node{**

**int data;**

**node \*next;**

**node \*pre;**

**};**

**int main(){**

**node \*head=new node;**

**node \*p=head;**

**int a[100]={1,2,3,4,5,4,3,2,1};**

**for(int i=0;a[i]!=0;i++){**

**node \*new\_node=new node;**

**new\_node->data=a[i];**

**new\_node->pre=p;**

**p->next=new\_node;**

**p=p->next;**

**}**

**p->next=head->next;**

**node\* p1=head->next;**

**while(p!=head){**

**if(p->data!=p1->data){**

**cout<<"不对称"<<endl;**

**return 0;**

**}**

**p=p->pre;**

**p1=p1->next;**

**}**

**cout<<"对称"<<endl;**

**}**