表结构习题

19335089 蒋雨芮

1. **线性表**：线性逻辑结构，有顺序表和链表两种存储结构。
   顺序表：用顺序结构存储数据，数据在内存中是连续的。
   链表：用链式结构存储数据，数据在内存中是不连续的。

2. **顺序表**：
   优点：结构简单、存储效率高，是随机存储结构。
   缺点：插入删除需移动数据元素效率底、需要预先分配空间。

   链表：
   优点：灵活，易扩容、插入删除效率高。
   缺点：访问效率低、存储效率偏低。

   故当只需要对数据进行读取操作时选用顺序表，频繁使用插入删除操作并涉及扩容时选取链表。

3. 平均移动 2/n 个节点，具体次数与长度和位置有关。

4. 有序，体现在存储单元通过保存存储的地址（指针）来关联。不一定，数据元素之间以某种方式有序关联。

5.

```
void insert(int a[],int length,int value){
    int left=0,right=length-1,mid=(left+right)/2;
    while(left<right){                    //用二分法找到该元素应该插入的位置
        mid=(left+right)/2;
        if(value==a[mid])
            break;
        if(value<a[mid])
            right=mid;
        else
            left=mid;
    }

    for(int i=length;i>mid;i--)           //将插入位置之后的元素右移
        a[i]=a[i-1];

    a[mid]=value;
}
```

6.

```
int ListLength(linkList L){
    linkList p=L;
    int length=0;

    while(p){
        p=p->next;
        length++;
    }

    return length;
}
```

7.

```
void removeDuplicates(linkList L){
    linkList p1=L;
    int flag;

    while(p1){                              //遍历链表
        flag=p1->value;
        linkList p2=p1->next;

        while(p2){
            linkList temp=p2->next;         //从 p1 的下一个开始比较去重
            if(p2->value==flag)
                deleteNode(p2);
            p2=temp;
        }

        p1=p1->next;
    }
}
```

8.

```
template <typename T>
void deleteValue(vector<T> A,T x,T y){
    for(int i=0;i<A.size();i++){     //遍历并删除
        if(A[i]>x&&A[i]<y)
            A.erase(i);
    }
}
```

9.

```
linkList mergeLists(linkList A,linkList B){
    linkList newList=new Node;  //新链表头节点
    linkList p1=A,p2=B;
    linkList temp;

    while(p1&&p2){
        if(p1->value<p2->value){       //两个指针分别指向 A,B 俩链表，
            temp=p1->next;             //将指针当前所指值小的插到头节点后，
            p1->next=newList->next;  //且指针后移

            newList=p1;
            p1=temp;
        }
        else{
            temp=p2->next;
            p2->next=newList->next;
            newList=p2;
            p2=temp;
        }
    }

    while(p1){                          //处理一条链表添加完的情况
        temp=p1->next;
        p1->next=newList->next;
        newList=p1;
        p1=temp;
    }

    while(p2){
        temp=p2->next;
        p2->next=newList->next;
        newList=p2;
        p2=temp;
    }

    temp=newList;                     //删除头节点
    newList=newList->next;
    delete temp;

    return newList;

}
```