编程题目

01-删除链表元素

问题描述分析

默认链表的开始元素需要为 `1`

1、已知单链表的存储结构如下所示，编写程序在L中将线性表L中第i个数据元素删除。

数据结构

```
typedef  int   ElemType;
typedef struct LNode{
     ElemType   data;       //数据域
     struct LNode  *next;   //指针域
}LNode,*LinkList;           // *LinkList为Lnode类型的指针
```

编写函数

```
Status ListDelete_L(LinkList &L,int i,ElemType &e){

}
```

- L 单链表
- i 要删除的元素序号
- e 返回的元素删除值

题解

```
Status ListDelete_L(LinkList L,int i,ElemType* e){
    LinkList p = L; int f = 0;
    while(p&&f<(i-1)){//将工作节点移动到待删除节点的前一个节点
       p = p->nodeNext;
       f++;
    }
    if(!p->nodeData||f>i-1){//元素删除序号大于总长度或小于1
       printf("null element");
       return false;
    }
    // 如果元素存在执行删除逻辑-
    //将待删除的前一个元素的指针域指向删除元素的指针域指向的元素,然后释放元素
    LinkList delNode = p->nodeNext;

 *e = delNode->nodeData;//指向待删除元素的数据域
    p->nodeNext = delNode->nodeNext;
```

```
    free(delNode);
    return true;
}
```

02-链表二分查找

问题描述分析

默认元素开始位置为1

2、已知顺序表ST的存储结构如下所示，编写程序在ST用折半查找方式中返回关键字为key变量的位置，如表中不存在则返回0。

数据结构

```
typedef struct {
    KeyType key;
} ElemType;
typedef struct {
    ElemType     *R; //表基址
    int  length;     //表长
}SSTable;
```

`SSTable` 是一个顺序表里面存储了 `R` 数组 和 `length` 数组长度

ElemType 是数组元素类型存储了 `key` 为元素的值

编写函数

```
int Search_Bin(SSTable ST,KeyType key){
}
```

折半查找

```
int main(){
  int anums[] = {1,2,3,4,5,6,7,8,9,10};
  int mid,high,low;
  low = 0; high = 9;
  int key = 4;
  while(low<=high){
    mid = (low+high)/2;
    if(anums[mid]==key){
      printf("find key: %d",mid);
      break;
    }else if(anums[mid]<key){
      low = mid+1;
    }else if(anums[mid]>key){
      high = mid-1;
```

```
    }
  }
}
```

题解

```
int Search_Bin(SSTable ST,KeyType key){
  int low = 1,high = ST.length,mid;
  while(low<=high){
    mid = (low+high)/2;
    if(ST.R[mid].key==key){
      return mid;//返回key值所在索引
    }else if(ST.R[mid].key<key){
      low = mid+1;
    }else if(ST.R[mid].key>key){
      high = mid-1;
    }
  }
  return 0;//未找到key索引
}
```

03-链表冒泡排序

问题描述分析

默认初始索引值为1

3、已知顺序表L的存储结构如下所示，编写程序用冒泡排序对顺序表L从小到大的排序 。

数据结构

```
# define MAXSIZE 20             //设记录不超过20个
typedef  int  KeyType ;         //设关键字为整型量（int型）
Typedef  struct {              //定义每个记录（数据元素）的结构
    KeyType      key ;         //关键字
}RedType ;
Typedef  struct {               //定义顺序表的结构
  RedType  r [ MAXSIZE +1 ];    //存储顺序表的向量，r[0]作哨兵或缓冲区
  int length ;                  //顺序表的长度
}SqList ;
```

编写函数

```
void bubble_sort(SqList &L) {}
```

冒泡排序

需要注意数组边界值 ， 每道题不一样

```
int main(){
  int anums[] = {2,5,6,7,4,9,3,1,8,0};
  for(int i=0;i<9;i++){
    for(int j=0;j<(9-i+1);j++){
      if(anums[j]>anums[j+1]){
        int temp = anums[j];
        anums[j] = anums[j+1];
        anums[j+1] = temp;
      }
    }
  }
  for(int i=0;i<10;i++){
    printf("%d ",anums[i]);
  }
}
```

题解

```
void bubble_sort(SqList* L){
  for(int i = 1;i < L->length;i++){
    for(int j = 1; j < (L->length-i+1);j++){
      if(L->r[j].key>L->r[j+1].key){
        int temp = L->r[j].key;
        L->r[j].key = L->r[j+1].key;
        L->r[j+1].key = temp;
      }
    }
  }
  return;
}
```

04-构造哈夫曼树

4、已知某哈夫曼树n个叶子节点，权值由输入值来确定个数大小，编写程序来构造哈夫曼树。

```
// HT是指向HTNode的二级指针 w为权值数组 n为原始节点个数
void CrateHuffmanTree(HuffmanTree* HT,int n){
  if(n<=1)return ;//单节点不构成
  int m = 2*n - 1; //生成二叉树的总节点个数

 *HT = (HuffmanTree) malloc((m+1)*sizeof(HTNode));//数组索引从1开始
  HuffmanTree p = *HT;
  for(int i=1;i<=m;i++){//初始化原始节点和生成二叉树节点
    (p+i)->weight = 0;
    (p+i)->right = 0;
    (p+i)->left = 0;
```

```
    (p+i)->parent = 0;
  }
  for(int i=1;i<=n;++i){//输入原始节点权值
    int  temp;scanf("%d",&temp);
    (p+i)->weight = temp;
  }
  //题解
  for(int i=n+1;i<=m;i++){
    int s1,s2;
    Select(*HT,i-1,&s1,&s2);//查找权重值最小的两个结点索引
    (*HT)[s1].parent = (*HT)[s2].parent = i;//给s1,s2设置父节点索引
    (*HT)[i].left = s1;
    (*HT)[i].right = s2;
    (*HT)[i].weight = (*HT)[s1].weight+(*HT)[s2].weight;
  }
}
```

参考：哈夫曼树（赫夫曼树、最优树）详解 (biancheng.net)

05-链表插入排序

问题描述分析

5、已知顺序表L的存储结构如下所示，编写程序用直接插入排序对顺序表L从小到大的排序。

数据结构

```
# define MAXSIZE 20             //设记录不超过20个
typedef  int  KeyType ;         //设关键字为整型量（int型）
Typedef  struct {              //定义每个记录（数据元素）的结构
    KeyType      key ;         //关键字
 }RedType ;
Typedef  struct {               //定义顺序表的结构
  RedType  r [ MAXSIZE +1 ];    //存储顺序表的向量，r[0]作哨兵或缓冲区
  int length ;                  //顺序表的长度
}SqList ;
```

编写函数

```
void InsertSort(SqList &L){}
```

直接插入排序

需要注意边界值，不同题目不一样可能索引值默认从1开始 0 索引做辅助哨兵作用

```
int main(){
  int anums[] = {2,5,6,7,4,9,3,1,8,0};
```

```
  for(int i=1;i<10;i++){
    if(anums[i]<anums[i-1]){
      int temp = anums[i];
      int j;
      for(j = i-1;temp<anums[j];j--){//升序排序
        anums[j+1] = anums[j];//后移元素
      }
      anums[j+1]=temp;
    }
  }
  for(int i=0;i<10;i++){
    printf("%d ",anums[i]);
  }
}
```

题解

```
void InsertSort(SqList* L){
  for(int i=2;i<=L->length;i++){
    if(L->r[i].key<L->r[i-1].key){
      L->r[0].key = L->r[i].key;
      int j;
      for(j=i-1;L->r[0].key<L->r[j].key;j--){
        L->r[j+1].key = L->r[j].key;
      }
      L->r[j+1].key = L->r[0].key;
    }
  }
}
```