**要求代码和实验报告规范，在算法思想中：对实验涉及的数据结构进行有效设计和分析；对算法进行分析并给出时间、空间复杂度的结论；清晰表达实验思路、出现的问题及解决方法。**

一、调试成功程序及说明

题目：

建立元素值为整型的顺序表，编程，用插入排序算法实现顺序表排序。

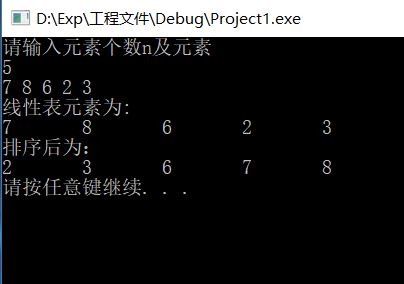
算法思想：

插入法排序，从第二个元素开始循环，将元素逐个插入有序数列。

时间复杂度：O(n^2)

空间复杂度: O(1)

运行结果：



结果分析：

输入的序列排序成功。

附源程序:

//建立元素值为整型的顺序表，插入排序算法实现顺序表排序。

//C++结构体实现

#include "List\_Sq.h"

void ListInsertionSort\_Sq(SqList &L)

{

// a 为顺序表数组头指针, t 临时存放数据

int \*a = L.elem, t;

int i, j;

//简单插入法排序

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

{

t = a[i];

for ( j = i - 1;j >= 0 && a[j] > t;j--)

a[j + 1] = a[j];

a[j + 1] = t;

}

return;

}

int main()

{

//排序算法测试

SqList L;

int n;

cout << "请输入元素个数n及元素" << endl;

cin >> n;

CreatList\_Sq(L, n);

cout << "线性表元素为:" << endl;

ListTraverse\_Sq(L);

//排序函数调用

ListInsertionSort\_Sq(L);

cout << "排序后为：" << endl;

ListTraverse\_Sq(L);

//顺序表销毁

DestroyList\_Sq(L);

system("pause");

return 0;

}

题目：

建立元素值为整型的单链表，编程，用插入排序算法实现单链表排序。

算法思想：

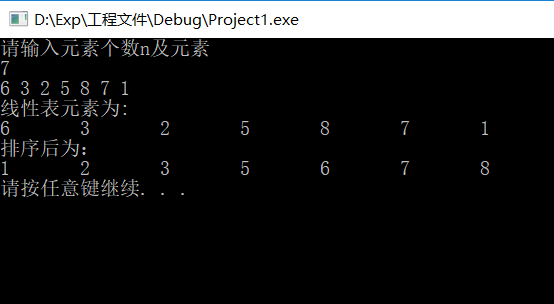
插入法排序，从第二个元素开始循环，将元素逐个插入有序数列。

在链表中，先将链表置空，再将结点元素逐个按序插入即可。

时间复杂度：O(n^2)

空间复杂度: O(1)

运行结果：



结果分析：

输入的序列排序成功。

附源程序：

//建立元素值为整型的单链表，插入排序算法实现单链表排序。

//C++结构体实现

#include "List\_Link.h"

//该函数实现将结点s插入非递减链表L后仍保持L为非递减链表

void AscendingInsert\_Link(LinkList &L, Link s)

{

Link p, q;

p = L.head;

q = p->next;

//循环结束后 q 指向第一个数据 >= s->data 的结点

while (q && q->data < s->data)

{

p = q;

q = q->next;

}

p->next = s;

s->next = q;

//若插入位置为链表尾部，则需要对尾指针做调整

if (!q)

L.tail = p;

return;

}

//插入法排序链式实现

void ListInsertionSort\_Link(LinkList &L)

{

// p 指针遍历链表元素结点， t 暂存 p 结点的下一个位置

Link p, t;

p = L.head->next;

//将链表置空，再将未释放的结点非递减插入

L.head->next = NULL;

L.tail = L.head;

while (p)

{

t = p->next;

AscendingInsert\_Link(L, p);

p = t;

}

return;

}

//测试函数功能

int main()

{

//排序算法测试

LinkList L;

int n;

cout << "请输入元素个数n及元素" << endl;

cin >> n;

CreatList\_Link(L, n);

cout << "线性表元素为:" << endl;

ListTraverse\_Link(L);

//排序函数调用

ListInsertionSort\_Link(L);

cout << "排序后为：" << endl;

ListTraverse\_Link(L);

//链表销毁

DestroyList\_Link(L);

system("pause");

return 0;

}

题目：

用顺序结构编程实现下列功能：假设以两个元素值为整型依值递增有序排列的线性表A和B 分别表示两个集合

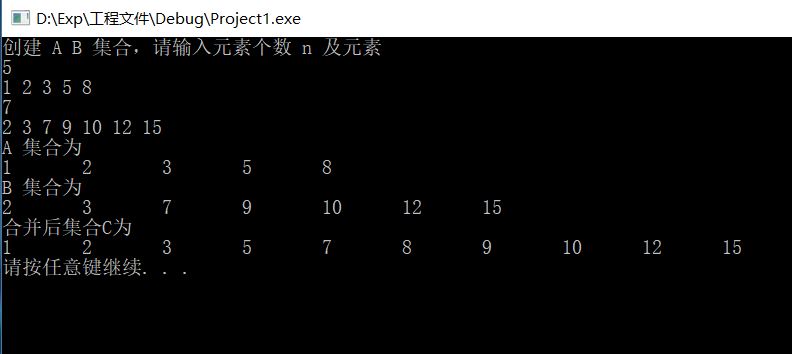
算法思想：

遍历 A B 集合，将 B 集合中与 A 集合中不重复的元素插入 A 集合，将 C 先ing表指针指向 A 线性表。

时间复杂度：O(max(La.length,Lb.length)\*La.length)

空间复杂度：O(1)

运行结果：



结果分析：

合并成功，新集合C仍是依值递增。

附源程序:

//线性表顺序结构实现 C = A ∪ B；

//A,B,C 均递增，且C在A的空间上建立

#include "List\_Sq.h"

void ListUnion\_Sq(SqList &La, SqList Lb,SqList &Lc)

{

//a,b记录两线性表中的数组首地址

int \*a, \*b;

a = La.elem;

b = Lb.elem;

//i,j记录 a b 数组中元素位置

int i = 0, j = 0;

//La Lb 均未遍历完时

while (i < La.length && j < Lb.length)

{

if (a[i] < b[j])

i++;

else if (a[i] == b[j])

{

i++;

j++;

}

else

{

ListInsert\_Sq(La, i + 1, b[j]);

i++;

j++;

}

}

//当A未遍历完，B遍历完成时，此时并集已完成

//当 Lb 未遍历完时，将 Lb 中剩余元素全部插入 La 中

while (j < Lb.length)

{

ListInsert\_Sq(La, i + 1, b[j]);

i++;

j++;

}

//线性表 Lc 指向线性表 La 的空间

Lc.elem = La.elem;

Lc.length = La.length;

Lc.listsize = La.listsize;

return;

}

//函数测试

int main()

{

SqList La, Lb, Lc;

int m, n;

cout << "创建 A B 集合，请输入元素个数 n 及元素" << endl;

cin >> m;

CreatList\_Sq(La, m);

cin >> n;

CreatList\_Sq(Lb, n);

cout << "A 集合为" << endl;

ListTraverse\_Sq(La);

cout << "B 集合为" << endl;

ListTraverse\_Sq(Lb);

ListUnion\_Sq(La, Lb, Lc);

cout << "合并后集合C为" << endl;

ListTraverse\_Sq(Lc);

//线性表销毁

DestroyList\_Sq(La);

DestroyList\_Sq(Lb);

system("pause");

return 0;

}

题目：

实现同上一题的功能，用链表实现。

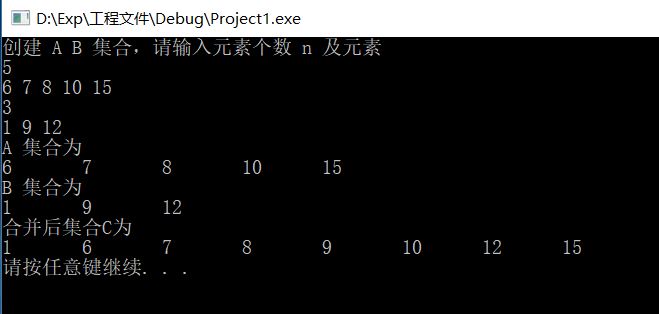
算法思想：

遍历 A B 集合组成的链表的结点，C 链表使用A B 链表的空间，在其上组成 一个有序的新表。

时间复杂度：O(La.length+Lb.length)

空间复杂度: O(1)

运行结果：



结果分析：

运行结果符合题目要求。

附源程序：

//线性表链式结构实现 C = A ∪ B；

//A,B,C 均递增，且C在A的空间上建立；

#include "List\_Link.h"

void ListUnion\_Link(LinkList &La, LinkList Lb,LinkList &Lc)

{

// p q 指向 La 中的结点 , s 指向 Lb 中的结点

Link p, q, s;

// t 记录 s 的前一个位置

Link t;

int e;

p = La.head;

q = p->next;

t = Lb.head;

s = t->next;

while (q && s)

{

if (q->data == s->data)

{

p = q;

q = q->next;

t = s;

s = s->next;

}

else if (q->data < s->data)

{

p = q;

q = q->next;

}

else

{

t->next = s->next;

p->next = s;

s->next = q;

p = s;

s = t->next;

}

}

//当A未遍历完，B遍历完成时，此时并集已完成

//如果 Lb 未遍历完，将 Lb 中剩余元素一次全部插入 La 中

if (s)

{

p->next = s;

// La 的尾部有添加元素，则此时需要修改尾指针

La.tail = Lb.tail;

}

//线性表 Lc 指向线性表 La 的空间

Lc.head = La.head;

Lc.len = La.len;

Lc.tail = La.tail;

return;

}

//函数测试

int main()

{

LinkList La, Lb, Lc;

int m, n;

cout << "创建 A B 集合，请输入元素个数 n 及元素" << endl;

cin >> m;

CreatList\_Link(La, m);

cin >> n;

CreatList\_Link(Lb, n);

cout << "A 集合为" << endl;

ListTraverse\_Link(La);

cout << "B 集合为" << endl;

ListTraverse\_Link(Lb);

ListUnion\_Link(La, Lb, Lc);

cout << "合并后集合C为" << endl;

ListTraverse\_Link(Lc);

//链表销毁

DestroyList\_Link(La);

DestroyList\_Link(Lb);

system("pause");

return 0;

}

题目：

首先输入正整数n，接着输入n个正整数，如果其中存在一个数，比该数大的个数等于比该数小的个数，则输出该数，如果不存在则输出-1。

算法思想：

顺序结构实现输入数据的存储；

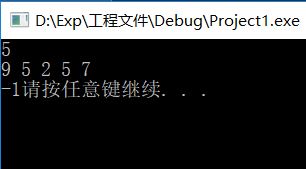
对线性表排序，找出中间的数，只有中间的一个或两个数可能是中间数，判断它左右比它小或大的数的个数是否相同；

若相同，则为中间数，反之，中间数不存在。

时间复杂度：约为排序算法的时间复杂度，使用简单选择排序法则为O(n^2)

空间复杂度: O(1)

运行结果：



结果分析：

运行结果符合题目要求。

附源程序：

//CSP题目

//问题描述：首先输入正整数n，接着输入n个正整数，如果其中存在一个数:

//比该数大的个数等于比该数小的个数，则输出该数，如果不存在则输出 - 1。

#include <iostream>

using namespace std;

void ListSort(int a[], int n)

{

int i, j, t;

for (i = 1;i < n;i++)

{

t = a[i];

for (j = i - 1;j >= 0 && a[j] > t;j--)

a[j + 1] = a[j];

a[j + 1] = t;

}

return;

}

bool MiddleExit(int a[], int n, int position)

{

int i;

int count1 = 0, count2 = 0;

int middle = a[position - 1];

for (i = 0;i < position - 1;i++)

if (a[i] != middle)

count1++;

for (i = position;i < n;i++)

if (a[i] != middle)

count2++;

if (count1 == count2)

return true;

else

return false;

}

int main()

{

int n;

int \*a;

cin >> n;

a = new int[n];

for (int i = 0;i < n;i++)

cin >> a[i];

//对输入数据排序

ListSort(a, n);

// n 为偶数时

if (n%2==0)

{

int position1, position2;

position1 = n / 2;

position2 = position1 + 1;

if (MiddleExit(a, n, position1))

cout << a[position1 - 1];

else if (MiddleExit(a, n, position2))

cout << a[position2 - 1];

else

cout << -1;

}

//n 为奇数时

if (n % 2 != 0)

{

int position;

position = (n + 1) / 2;

if (MiddleExit(a, n, position))

cout << a[position-1];

else

cout << -1;

}

//释放空间

delete[]a;

system("pause");

return 0;

}

二、未调试成功程序及说明

此次程序调试成功。

三、小结

内容：

简单插入法排序

线性表合并的空间利用，及在已知链表上构造新的链表。

注意：

未封装的线性表链表要释放空间，动态申请的空间要释放。