



电子科技大学
University of Electronic Science and Technology of China

《软件技术基础》

上机报告总结

学生姓名：李宇航

学号：2014010906024

指导教师：廖丹

编译环境：OSX 10.11.1

编译工具：Xcode7.1.1 & IntelliJ Idea14

源码地址：<https://github.com/Toxni/experiment>

上机实验一：顺序表

1. 首先创建一个顺序表：从键盘读入一组整数（长度小于等于20），按输入顺序放入顺序表，输入以-1结束（注意-1不放到顺序表内）；将创建好的顺序表元素依次输出到屏幕上。
2. 在已创建好的顺序表中插入一个元素：从键盘读入需插入的元素值和插入位置，调用插入函数完成插入操作；然后将顺序表元素依次输出到屏幕上。
3. 在已创建好的顺序表中删除一个元素：从键盘读入欲删除的元素位置（序号），调用删除函数完成删除操作；然后将顺序表元素依次输出到屏幕上。
4. 设线性表存放在向量 A[1..MAXNUM] 的前elenum个分量中，且递增有序，写一算法，将x插入到线性表适当位置，以保持线性表带有序性。

// 实现代码过于冗长 见附页

// 要求 1-3 实现结果如下

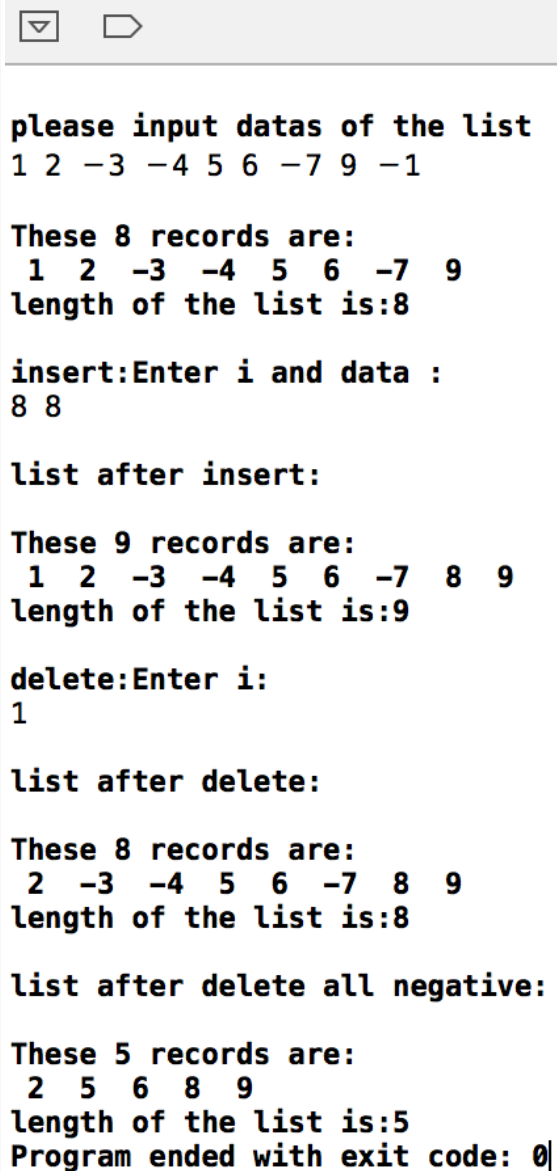
键入 1 2 -3 -4 5 6 -7 9 -1 enter

键入 8 8 enter
// 插入的位置和值

// 在第八个位置插入数字8

键入 1
// 删除的位置

// 删除负值之后的顺序表



```
please input datas of the list
1 2 -3 -4 5 6 -7 9 -1

These 8 records are:
1 2 -3 -4 5 6 -7 9
length of the list is:8

insert:Enter i and data :
8 8

list after insert:

These 9 records are:
1 2 -3 -4 5 6 -7 8 9
length of the list is:9

delete:Enter i:
1

list after delete:

These 8 records are:
2 -3 -4 5 6 -7 8 9
length of the list is:8

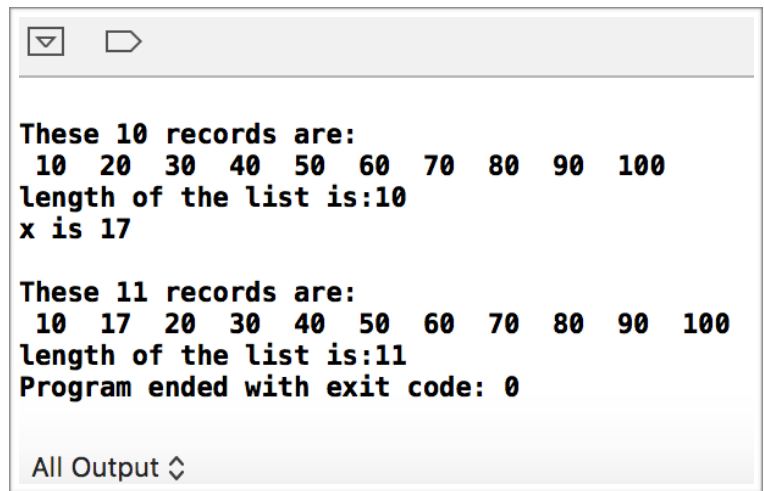
list after delete all negative:

These 5 records are:
2 5 6 8 9
length of the list is:5
Program ended with exit code: 0
```

All Output ↕

// 要求 4 实现结果如下

```
// 生成10 - 100的数字  
  
// 随机生成一个0 - 100的数字  
  
// 将随机生成数插入队列 [*]
```



The screenshot shows a terminal window with the following output:

```
These 10 records are:  
10 20 30 40 50 60 70 80 90 100  
length of the list is:10  
x is 17  
  
These 11 records are:  
10 17 20 30 40 50 60 70 80 90 100  
length of the list is:11  
Program ended with exit code: 0  
  
All Output ^
```

[*] 此处随机生成数字的函数creatx为

```
int creatx() {  
    int x;  
    srand((unsigned)time(NULL));  
    x = rand()%100;  
    printf("x is %d\n", x);  
    X = x;  
    return x;  
}
```

由于rand() 函数产生的是伪随机数，为了避免每次实验结果一致，在此给rand() 函数以当前时间数字为seed，如此来模拟“真随机”。

// 实验心得与体会

实验一由于老师PPT里面给出了大部分代码，自己只编写了一小部分。诚然，我自己编写代码的能力远不及老师。不过这里有几个无伤大雅的代码规范方面的小细节老师编写的时候可能没有注意到

- 逗号、分号在其后加空格；
- 比较操作符, 赋值操作符"="、 "+="，算术操作符"+"、"%", 逻辑操作符"&&"、"&", 位域操作符 "<<"、">" 等双目操作符的前后都应加空格；
- 变量名，函数名一般采用小驼峰法命名，比如createlist应该写为createList，即第二个单词开始起首字母大写。

Paul Graham在他的书中说过：代码是写给人看的，顺便能在机器上运行。个人认为代码规范还是很重要啦。

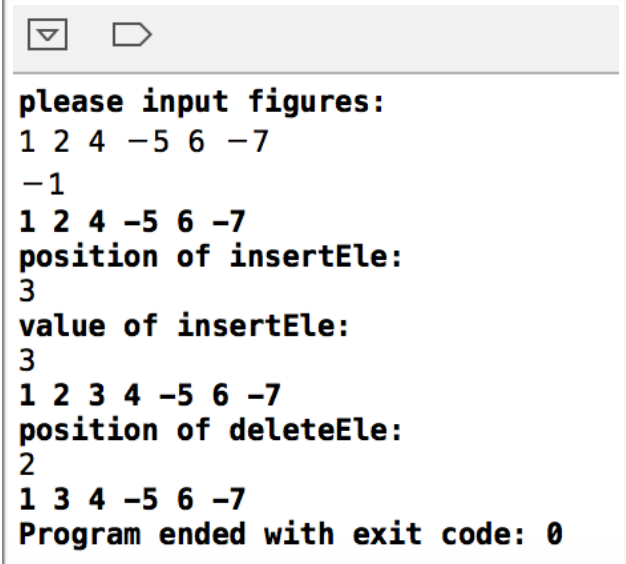
上机实验二：链表

1. 首先创建一个单链表：从键盘读入五个整数，按输入顺序形成单链表。将创建好的链表元素依次输出到屏幕上。
2. 在已创建好的链表中插入一个元素：从键盘读入元素值和插入位置，调用插入函数完成插入操作。然后将链表元素依次输出到屏幕上。
3. 在已创建好的链表中删除一个元素：从键盘读入欲删除的元素位置（序号），调用删除函数完成删除操作。然后将链表元素依次输出到屏幕上。

// 实现代码过于冗长 见附页

// 要求 1-3 实现结果如下

```
键入 1 2 4 -5 6 -7 enter
键入 -1 enter // -1作为结束符
键入 3 enter // 插入位置
键入 3 enter // 插入值
键入 2 enter // 删除的位置
// 输出结果
```



```
please input figures:
1 2 4 -5 6 -7
-1
1 2 4 -5 6 -7
position of insertEle:
3
value of insertEle:
3
1 2 3 4 -5 6 -7
position of deleteEle:
2
1 3 4 -5 6 -7
Program ended with exit code: 0
```

// 实验心得与体会

- 合理使用typedef。编写好的链表插入，删除等函数可以封装起来使用。此时在函数顶部规定typedef int ElemType，在需要修改链表内部数据类型的时候直接修改typedef后面的类型即可。由此我想到了类似思想的less和sass，两者都是css的超集，为css增加了动态语言特性。比如说less就可以在顶部规定@color = #ff0000，那么后面的color变量都表示红色，如果需要修改整体颜色，直接修改@color变量即可，而不是去css内部一个一个修改；
- 单链表的开链操作很容易，删除、插入操作比顺序表来得高效。

上机实验三：栈和队列

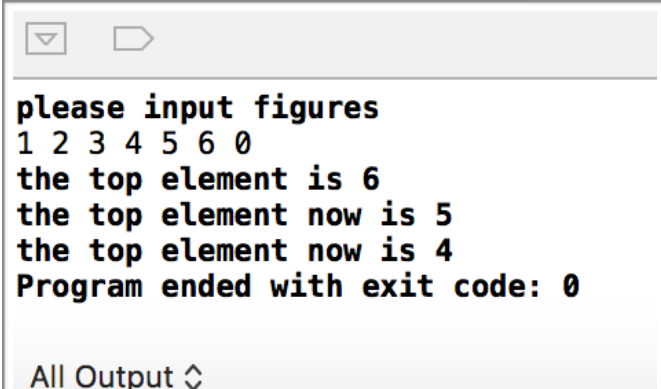
1. 编写链栈的pop和push函数。编写main函数，首先建立一空链栈；调用进栈函数，将从键盘输入的数据元素逐个进栈，输入0结束；显示进栈后的数据元素；调用两次出栈函数，显示出栈后的数据元素。
2. 编写循环队列的出列，入列函数。编写函数：void aa(queue_type *q)；调用出对函数把队列q中的元素一一出对列。编写main函数，首先建立一个队列，其中的数据元素为：{2, 3, -4, 6, -5, 8, -9, 7, -10, 20}；然后调用aa函数，并将aa函数调用前后队列的数据元素分别输出到屏幕上。

// 实现代码过于冗长 见附页

// 要求 1 实现结果如下

键入 1 2 3 4 5 6 0 enter

```
// pop once  
// pop twice
```

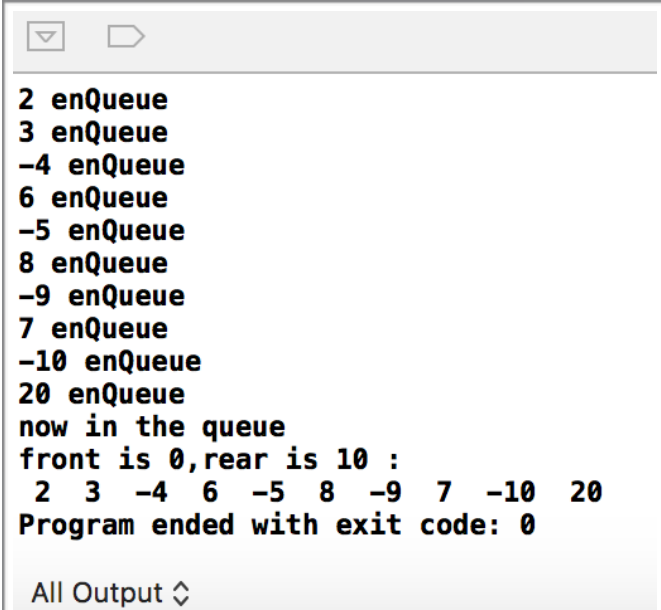


```
please input figures  
1 2 3 4 5 6 0  
the top element is 6  
the top element now is 5  
the top element now is 4  
Program ended with exit code: 0  
All Output ↕
```

// 要求 2 实现结果如下

键入 2 enter // 下同

// 调用 aa() 函数，输出结果



```
2 enqueue  
3 enqueue  
-4 enqueue  
6 enqueue  
-5 enqueue  
8 enqueue  
-9 enqueue  
7 enqueue  
-10 enqueue  
20 enqueue  
now in the queue  
front is 0, rear is 10 :  
2 3 -4 6 -5 8 -9 7 -10 20  
Program ended with exit code: 0  
All Output ↕
```

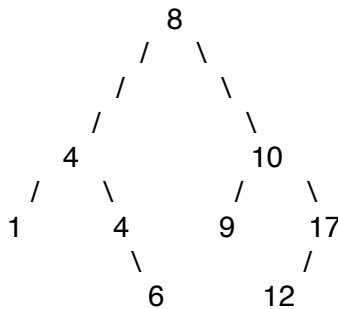
上机实验四：二叉树和哈夫曼树

1. 编写二叉树的创建函数，可以是排序二叉树的创建思路（见教材），或者以先序遍历为框架。编写中序、后序、先序遍历函数编写main()函数，先调用create函数，建立一颗二叉排序树；然后分别调用中序、后序、先序遍历函数，将二叉树的先序、中序和后序遍历序列输出到屏幕上。
2. 输入一组数（权值），编写算法建立哈夫曼树，输出每个权值对应的二进制编码。

// 实现代码过于冗长 见附页

// 要求 1 实现结果如下

// 测试树形如下



// 键入各节点值，-1 代表空：

// 先序遍历结果 —>

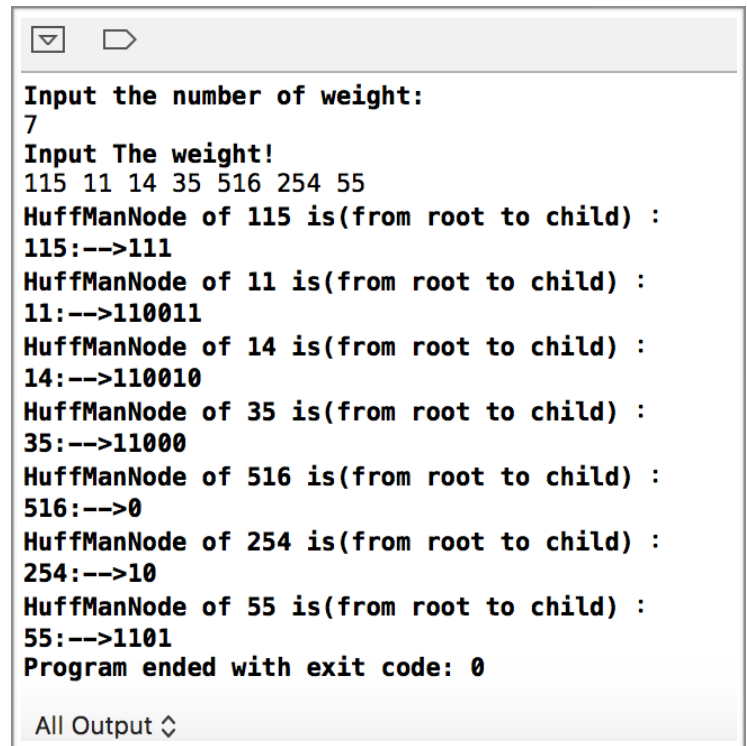
// 中序遍历结果 —>

// 后序遍历结果 —>

```
please input the value of first node, -1 means no node here:
8
input leftChild of 8: 4
input leftChild of 4: 1
input leftChild of 1: -1
input rightChild of 1: -1
input rightChild of 4: 4
input leftChild of 4: -1
input rightChild of 4: 6
input leftChild of 6: -1
input rightChild of 6: -1
input rightChild of 8: 10
input leftChild of 10: 9
input leftChild of 9: -1
input rightChild of 9: -1
input rightChild of 10: 17
input leftChild of 17: 12
input leftChild of 12: -1
input rightChild of 12: -1
input rightChild of 17: -1
preorder traversal:
8 4 1 4 6 10 9 17 12
inorder traversal:
1 4 4 6 8 9 10 12 17
preorder traversal:
1 6 4 4 9 12 17 10 8
Program ended with exit code: 0
```

// 要求 2 实现结果如下

```
// 键入路径个数7  
  
// 键入路径权重值  
  
  
// 输出各权值对应的二进制编码
```



```
Input the number of weight:  
7  
Input The weight!  
115 11 14 35 516 254 55  
HuffManNode of 115 is(from root to child) :  
115:-->111  
HuffManNode of 11 is(from root to child) :  
11:-->110011  
HuffManNode of 14 is(from root to child) :  
14:-->110010  
HuffManNode of 35 is(from root to child) :  
35:-->11000  
HuffManNode of 516 is(from root to child) :  
516:-->0  
HuffManNode of 254 is(from root to child) :  
254:-->10  
HuffManNode of 55 is(from root to child) :  
55:-->1101  
Program ended with exit code: 0  
  
All Output
```

// 疑问

- 如上输入权重为115 11 14 35 516 254 55，得到二进制编码结果与教科书 p47 上不符？

// 尝试解答

- 哈夫曼树本来就不是唯一的，它是权路径长度之和最小的二叉树的统称，左右设置方式按使用者习惯，原则上建议一棵树的排列方式唯一，即左小右大或左大右小。同时，考虑到哈夫曼树的哈夫曼编码应用，为获得较小的码方差，一般将合并所得的权值（哈夫曼编码中的概率）放在较上的位置。下面左为课本上构造树，右为我的代码的构造树。显然，教科书上的树形更好。

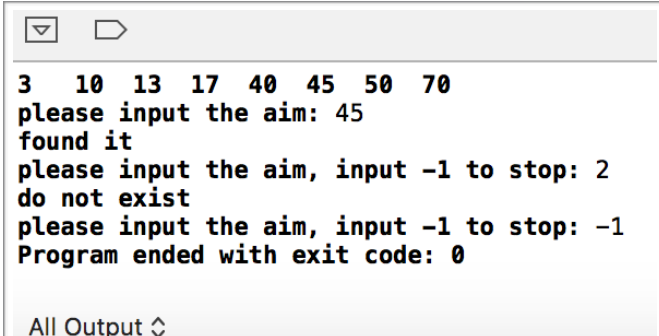
上机实验五：查找和排序

1. 编设有序序列的数据元素为 (3, 10, 13, 17, 40, 43, 50, 70), 分别编写顺序、二分法查找函数, 并且查找函数并使用其寻找元素的位置。
2. 编写简单选择法排序、直接插入法、冒泡法排序, 并且在主程序中输入一组数据元素(513, 87, 512, 61, 908, 170, 897, 275, 653, 462), 分别调用三种排序函数, 输出每趟排序结果。

// 实现代码过于冗长 见附页

// 要求 1 顺序法寻找实现结果如下

```
输出待寻找数组
键入45 enter
// 找到了found it
键入2 enter
// 不存在 do not exist
键入-1 enter
// 退出
```

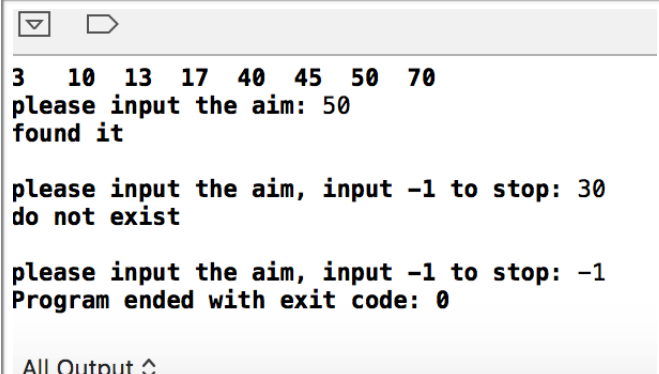


```
3  10  13  17  40  45  50  70
please input the aim: 45
found it
please input the aim, input -1 to stop: 2
do not exist
please input the aim, input -1 to stop: -1
Program ended with exit code: 0

All Output ↕
```

// 要求 1 二分法寻找实现结果如下

```
输出待寻找数组
键入50 enter
// 找到了found it
键入30 enter
// 不存在 do not exist
键入-1 enter
// 退出
```



```
3  10  13  17  40  45  50  70
please input the aim: 50
found it

please input the aim, input -1 to stop: 30
do not exist

please input the aim, input -1 to stop: -1
Program ended with exit code: 0

All Output ↕
```



// 实验心得

- 顺序法是一种简单的寻找算法, 其实现方法是从序列的起始元素开始, 逐个将序列中的元素与所要查找的元素进行比较。
- 二分法优点是查找速度快, 缺点是要求所要找的数据必须是有序序列。

// 要求 2 简单选择法排序实现结果如下

// 排序前数组

// 排序后数组





```
arr before sort is:
513 87 512 61 908 170 897 275 653 462
arr after sort is:
61 87 170 275 462 512 513 653 897 908
Program ended with exit code: 0
```

// 要求 2 插入法排序实现结果如下

// 排序前数组

// 排序后数组





```
arr before sort is:
513 87 512 61 908 170 897 275 653 462
arr after sort is:
61 87 170 275 462 512 513 653 897 908
Program ended with exit code: 0
```

// 要求 2 冒泡法排序实现结果如下

// 排序前数组

// 排序后数组



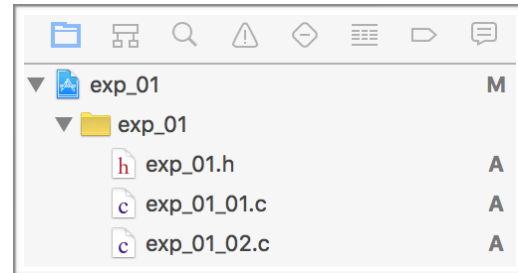
```
arr before sort is:
513 87 512 61 908 170 897 275 653 462
arr after sort is:
61 87 170 275 462 512 513 653 897 908
Program ended with exit code: 0
```

// 实验心得

- 选择排序，时间复杂度为 $O(n^2)$ ，不稳定排序，适合规模比较小的。
- 简单插入排序，时间复杂度为 $O(n^2)$ ，稳定排序，适合已经排好序的。
- 冒泡排序，时间复杂度为 $O(n^2)$ ，稳定排序，适合规模比较小的。
- * 归并排序,时间复杂度为 $O(n\log n)$,稳定排序，适合规模比较大的排序。

一 顺序表 代码如下:

文件结构如右图, exp_01.h 存放插入、删除等主要功能函数; exp_01_01.c 存放实验要求 1-3 主程序代码; exp_01_02 存放实验要求 4 主程序代码。



exp_01.h

```
//
// exp_01.h
// exp_01
//
// Created by Toxni on 12/7/15.
// Copyright © 2015 Toxni. All rights reserved.
//

#ifndef exp_01_h
#define exp_01_h

#include <stdio.h>

#endif /* exp_01_h */

#define MAXNUM 20
#define true 1
#define false 0

typedef struct {
    int data[MAXNUM];
    int length;
}list_type;

void showlist(list_type *lp) {
    int i;
    printf("\nThese %d records are:\n",
lp->length);
    if (lp->length <= 0)
    {
        printf("No data!\n");
        return;
    }
    for (i = 0; i<lp->length; i++)
        printf(" %d ", lp->data[i]);
    printf("\nlength of the list is:%d\n",
lp->length);
}
```

```
//insert new element to the list

int insertlist(list_type *lp, int new_elem,
int i) {
    int j;
    if (lp->length >= MAXNUM) {
        printf("the list is full,can not
insert.");
        return(false);
    }
    if (i < 1 || i > lp->length + 1) {
        printf("\n%d is invalid value",
i);
        return(false);
    }
    for (j = lp->length; j >= (i - 1);
j--) {
        lp->data[j + 1] = lp->data[j];
    }
    lp->data[i - 1] = new_elem;
    lp->length++;
    return(true);
}

//delete a element in the list

int deletelist(list_type *lp, int i) {
    int j;
    if(i < 1 || i > lp->length)
    {
        printf("elem not exist");
        return(false);
    }
    for (j = i; j < lp->length + 1; j++) {
        lp->data[j - 1] = lp->data[j];
    }
    lp->length--;
    return(true);
}

//delete negative element

void delete_negative(list_type *lp) {
    int i;
    for (i = lp->length; i >= 1; i--) {
        if (lp->data[i - 1] < 0) {
            deletelist(&*lp, i);
        }
    }
}
```

exp_01_01.c

```

//
// exp_01.c
// exp_01
//
// Created by Toxni on 12/7/15.
// Copyright © 2015 Toxni. All
rights reserved.
//

#include "exp_01.h"

/*create a list:input data from
keyboard,end by -1*/

void createlist(list_type *lp) {
    int i, elem;
    lp->length = 0;
    printf("\nplease input datas of
the list\n");
    for (i = 0; i < MAXNUM; i++)
    {
        scanf(" %d", &elem);
        if(elem == -1) break;
        lp->data[i] = elem;
        lp->length++;
    }
}

int main() {
    list_type list;
    int i, data;
    createlist(&list);
    showlist(&list);
    printf("\ninsert:Enter i and data
:\n");
    scanf("%d" "%d", &i, &data);
    insertlist(&list, data, i);
    printf("\nlist after insert:\n");
    showlist(&list);
    printf("\ndelete:Enter i:\n");
    scanf("%d", &i);
    deletelist(&list, i);
    printf("\nlist after delete:\n");
    showlist(&list);
    delete_negative(&list);
    printf("\nlist after delete all
negative:\n");
    showlist(&list);
}

```

exp_01_02.c

```

//
// exp_01_02.c
// exp_01
//
// Created by Toxni on 12/8/15.
// Copyright © 2015 Toxni. All rights
reserved.
//

#include <stdlib.h>
#include <time.h>
#include "exp_01.h"

int X;
int NEWMAX = 10;

void createlist(list_type *lp) {
    int i;
    for (i = 0; i < NEWMAX; i++) {
        lp->data[i] = (i + 1)*10;
    }
    lp->length = NEWMAX;
}

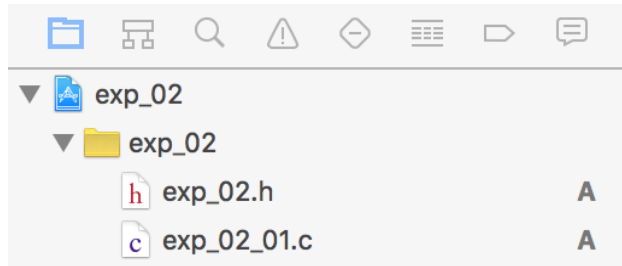
int creatx() {
    int x;
    srand((unsigned)time(NULL));
    x = rand()%100;
    printf("x is %d\n", x);
    X = x;
    return x;
}

int main() {
    int i;
    list_type list;
    createlist(&list);
    showlist(&list);
    creatx();
    for (i = 0; i < NEWMAX; i++) {
        if (list.data[i] >= X) {
            insertlist(&list, X, i + 1);
            break;
        }
    }
    showlist(&list);
}

```

二 链表 代码如下:

文件结构如右图, exp_02.h 存放插入、删除等主要功能函数; exp_02_01.c 存放实验要求主函数代码。



exp_02.h

```
//
// exp_02.h
// exp_02
//
// Created by Toxni on 12/22/15.
// Copyright © 2015 Toxni. All rights
// reserved.
//

#ifndef exp_02_h
#define exp_02_h

#include <stdio.h>
#include <stdlib.h>

#endif /* exp_02_h */

typedef int ElemType;

typedef struct Node {
    ElemType data;
    struct Node *next;
}Node, *LinkedList;

//init the chain
LinkedList LinkedListInit() {
    Node *L;
    L = (Node *)malloc(sizeof(Node));
    if(L == NULL)
        printf("malloc failed\n");
    L->next = NULL;
    return 0;
}
```

```
LinkedList LinkedListCreate() {
    Node *L;
    L = (Node *)malloc(sizeof(Node));
    L->next = NULL;
    Node *r;
    r = L;
    ElemType x;
    while(scanf("%d",&x) != EOF)
    {
        if (x == -1) {
            break;
        }
        Node *p;
        p = (Node *)malloc(sizeof(Node));
        p->data = x;
        r->next = p;
        r = p;
    }
    return L;
}

LinkedList LinkedListInsert(LinkedList
L, int i, ElemType x) {
    Node *pre;
    pre = L;
    int tempi = 0;
    for (tempi = 1; tempi < i; tempi++)
        pre = pre->next;
    Node *p;
    p = (Node *)malloc(sizeof(Node));
    p->data = x;
    p->next = pre->next;
    pre->next = p;

    return L;
}

LinkedList LinkedListDelete(LinkedList L,
int i) {
    Node *pre;
    Node *p = NULL;
    pre = L;
    int tempi = 0;
    for (tempi = 1; tempi <= i; tempi++) {
        p = pre;
        pre = pre->next;
    }
    p->next = pre->next;

    free(p);
    return L;
}
```

exp_02_01.c

```
//
// exp_02.c
// exp_02
//
// Created by Toxni on 12/7/15.
// Copyright © 2015 Toxni. All rights reserved.
//

#include "exp_02.h"

int main() {
    LinkedList list, start;
    printf("please input figures:\n");
    list = LinkedListCreate();
    for(start = list->next; start != NULL; start = start->next)
        printf("%d ", start->data);
    printf("\n");

    int i;
    ElemType x;
    printf("position of insertEle:\n");
    scanf("%d",&i);
    printf("value of insertEle:\n");
    scanf("%d",&x);
    LinkedListInsert(list, i, x);

    for(start = list->next; start != NULL; start = start->next)
        printf("%d ", start->data);

    printf("\n");
    printf("position of deleteEle:\n");
    scanf("%d",&i);
    LinkedListDelete(list,i);

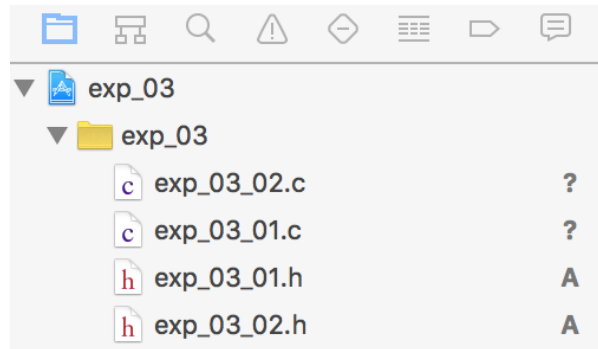
    for(start = list->next; start != NULL; start = start->next)
        printf("%d ", start->data);

    printf("\n");

    return 0;
}
```

三 栈与队列 代码如下:

文件结构如右图, exp_03_01.h 存放链栈插入、删除等主要功能函数; exp_03_01.c 存放实验要求 1 主程序代码; exp_03_02.h 存放循环队列插入、删除等主要功能函数; exp_03_02.c 存放实验要求 2 主程序代码。



exp_03_01.h

```
//
// exp_03.h
// exp_03
//
// Created by Toxni on 12/20/15.
// Copyright © 2015 Toxni. All rights
// reserved.
//

#ifndef exp_03_h
#define exp_03_h

#include <stdio.h>
#include <stdlib.h>

#endif /* exp_03_h */

typedef int elementType;
typedef struct node {
    elementType data;
    struct node *next;
}stackNode, *linkStack;

void initStack(linkStack top) {
    top->next = NULL;
}

int push(linkStack top, elementType
element) {
    stackNode *temp;
    temp = (stackNode
*)malloc(sizeof(stackNode));
    if(temp == NULL) return 0;
    temp->data = element;
    temp->next = top->next;
    top->next = temp;
    return 1;
}

int pop(linkStack top, elementType
*element) {
    stackNode *temp = top->next;
    *element = temp->data;
    top->next = temp->next;
    free(temp);
    return 1;
}
```

exp_03_01.c

```
//
// exp_03.c
// exp_03
//
// Created by Toxni on 12/20/15.
// Copyright © 2015 Toxni. All rights
// reserved.
//

#include "exp_03_01.h"

int main() {
    linkStack s;
    elementType x;
    elementType t = 0;
    printf("please input figures\n");
    s =
(linkStack)malloc(sizeof(stackNode));

    while(scanf("%d",&x) != EOF) {
        if (x == 0) {
            break;
        }
        else {
            push(s, x);
        }
    }
    int topEle;
    pop(s, &topEle);

    // pop element 0 out
    t = topEle;
    printf("the top element is %d\n", t);

    // pop once
    pop(s, &topEle);
    t = topEle;
    printf("the top element now is %d\n",
t);

    // pop twice
    pop(s, &topEle);
    t = topEle;
    printf("the top element now is %d\n",
t);

}
```

exp_03_02.h

```
//
// exp_03_02.h
// exp_03
//
// Created by Toxni on 12/28/15.
// Copyright © 2015 Toxni. All rights
// reserved.
//

#ifndef exp_03_02_h
#define exp_03_02_h

#endif /* exp_03_02_h */

//
// exp_04_01.h
// exp_04
//
// Created by Toxni on 12/28/15.
// Copyright © 2015 Toxni. All rights
// reserved.
//

#ifndef exp_04_01_h
#define exp_04_01_h

#include <stdio.h>
#include <stdlib.h>

#endif /* exp_04_01_h */

typedef int ElemType;
typedef int Status;

typedef struct BiTNode{
    ElemType data;
    struct BiTNode* lChild, *rChild;
}BiTNode, *BiTree;

//create with preorder

Status CreateBiTree(BiTree *T)
{
    ElemType ch;
    ElemType temp;

    scanf("%d", &ch);
    temp = getchar();

    if (-1 == ch)
        *T = NULL;
    else
    {
        *T =
        (BiTree)malloc(sizeof(BiTNode));
        if (!(*T))
            exit(-1);
    }
}
```

exp_03_02.c

```
//
// exp_03.c
// exp_03
//
// Created by Toxni on 12/20/15.
// Copyright © 2015 Toxni. All rights
// reserved.
//

#include "exp_03_02.h"

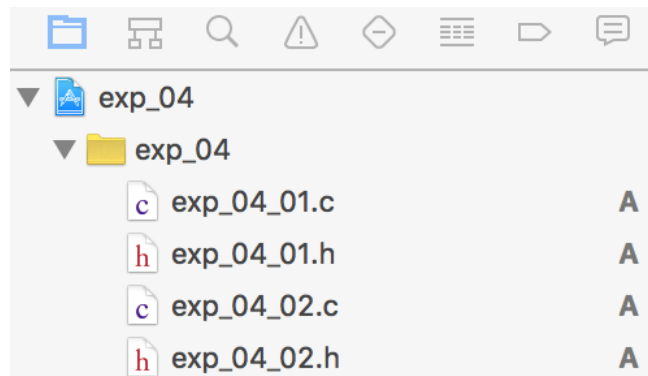
int main() {
    QUEUE queue;

    initQueue(&queue);
    enqueue(&queue, 2);
    enqueue(&queue, 3);
    enqueue(&queue, -4);
    enqueue(&queue, 6);
    enqueue(&queue, -5);
    enqueue(&queue, 8);
    enqueue(&queue, -9);
    enqueue(&queue, 7);
    enqueue(&queue, -10);
    enqueue(&queue, 20);

    aa(&queue);
}
```

四 二叉树与哈夫曼树 代码如下:

文件结构如右图, exp_04_01.h 存放二叉树构造等主要功能函数; exp_04_01.c 存放实验要求 1 主程序代码; exp_04_02.h 存放哈夫曼树构造等主要功能函数; exp_04_02.c 存放实验要求 2 主程序代码。



exp_04_01.h

```
//  
// exp_04_01.h  
// exp_04  
//  
// Created by Toxni on 12/28/15.  
// Copyright © 2015 Toxni. All rights  
// reserved.  
//
```

```
#ifndef exp_04_01_h  
#define exp_04_01_h
```

```
#include <stdio.h>  
#include <stdlib.h>
```

```
#endif /* exp_04_01_h */
```

```
typedef int ElemType;  
typedef int Status;
```

```
typedef struct BiTNode{  
    ElemType data;  
    struct BiTNode* lChild, *rChild;  
}BiTNode, *BiTree;
```

```
//create with preorder
```

```
Status CreateBiTree(BiTree *T)  
{  
    ElemType ch;  
    ElemType temp;  
    scanf("%d", &ch);  
    temp = getchar();
```

```
    if (-1 == ch)  
        *T = NULL;  
    else  
    {  
        *T =  
(BiTree)malloc(sizeof(BiTNode));  
        if (!(*T))  
            exit(-1);
```

```
        (*T)->data = ch;  
        printf("input leftChild of  
%d: ", ch);  
        CreateBiTree(&(*T)->lChild);  
        printf("input rightChild of  
%d: ", ch);  
        CreateBiTree(&(*T)->rChild);  
    }  
}
```

```
    return 1;  
}
```

```
//preorder traversal  
void TraverseBiTree(BiTree T)  
{
```

```
    if (NULL == T)  
        return ;  
    printf("%d ", T->data);  
    TraverseBiTree(T->lChild);  
    TraverseBiTree(T->rChild);  
}
```

```
//inorder traversal  
void InOrderBiTree(BiTree T)  
{
```

```
    if (NULL == T)  
        return ;  
    InOrderBiTree(T->lChild);  
    printf("%d ", T->data);  
    InOrderBiTree(T->rChild);  
}
```

```
//postorder traversal  
void PostOrderBiTree(BiTree T)  
{
```

```
    if (NULL == T)  
        return ;  
    PostOrderBiTree(T->lChild);  
    PostOrderBiTree(T->rChild);  
    printf("%d ", T->data);  
}
```


exp_04_02.h

// 此处部分代码参考自 <http://blog.csdn.net/m6830098/article/details/8713066>

```
//
// exp_04_02.h
// exp_04
//
// Created by Toxni on 12/28/15.
// Copyright © 2015 Toxni. All rights
// reserved.
//

#ifndef exp_04_02_h
#define exp_04_02_h

#include <stdio.h>
#include <stdlib.h>
#include <string.h>

#define OK 1
#define ERROR 0
#define OVERFLOW -1

#define Status int

#endif /* exp_04_02_h */

//哈夫曼树节点类型定义
typedef struct HTNodew
{
    unsigned int weight;//权重
    unsigned int parent,lchild,rchild;//
    父节点, 左孩子, 右孩子(无符号类型)
}HTNode,*HuffmanTree;

typedef char ** HuffmanCode;

HuffmanTree HT;

//选取最小的二个序列
Status Select(int n,int *s1,int *s2,int
*foot);

//构造哈夫曼树
Status HuffmanCoding(HuffmanCode *HC,
int n,unsigned int *w)
{
    /*
    **w存放n个字符的权值(均大于0), 构造哈夫曼
    树HT, 并且求出n个字符的哈夫曼编码HC
    */
    if(n<=0)
        return ERROR;
    int m=2*n-1; //m是总共需要的节
    点个数

    HT =
    (HuffmanTree)malloc(sizeof(HTNode)*(m
    +1)); //0号单元不使用, 下标从1开始到m结束
    int i=1;
    HuffmanTree p;
    for(p = HT,p++;i<=n;i++,p++,w++)//依次
    给
    }
    //设置一个m长度的数组记录节点是否被访问过
```

```
int *foot=(int
*)malloc(sizeof(int)*(m+1));// (下标也是从1
开始, 为了和上面的正好对应)
int *s=foot;
s++;
int k;
for( k=1;k<=m;k++)
{
    *(s++)=0;//初始化为0, 当有访问过, 则
    赋值1, 表示已经访问过了
}
/*
**构建哈夫曼树
*/
for(i = n+1;i<=m;i++) //依次给后面的n-
1个节点赋值 (从n+1到m)
{
    /*
    **在HT[1...i-1]选择parent为0 且
    weight最小的二个节点, 其序号为s1和s2
    */
    int *s1 = (int
*)malloc(sizeof(int ));
    int *s2 = (int
*)malloc(sizeof(int ));
    if(!s1||!s2)
    {
        printf("分配内存空间错误.\n");
        return ERROR;
    }

    Select(i-1,s1,s2,foot);
    HT[*s1].parent = i;
    HT[*s2].parent = i;
    HT[i].lchild = *s1;
    HT[i].rchild = *s2;
    HT[i].weight = HT[*s1].weight +
    HT[*s2].weight;
}

//-----从叶子到根节点逆向求每个字符的哈夫曼
编码-----//

*HC=(HuffmanCode)malloc(sizeof(char
*)*(n+1));//从1开始, 抛弃下标1

char *cd = (char
*)malloc(sizeof(char)*n); //分配求编码的
工作区间 (n个字符最长的编码需要n-1个空间, 加上结
束符号)
cd[n-1]='\0';//编码结束符

int start;//用来跟踪位置
for(i = 1;i<=n;i++)//逐个字符求哈夫曼编
码
{
    start = n-1;//编码结束符位置
```

上接exp_04_02.h

```
int c, f;
start--;
for(c = i, f = HT[i].parent; f != 0; c = f, f = HT[f].parent)
{
    if(HT[f].lchild == c)
        cd[start] = '0';
    else
        cd[start] = '1';
    start--;
}
start+=1; //这里一定要加1 否则多分配了内存空间
(*HC)[i] = (char *)malloc((n-start)*sizeof(char)); //分配n-start个char空间即可
strcpy((*HC)[i], &cd[start]);
char *p = (*HC)[i];
printf("HuffmanNode of %d is (from root to child) : \n", HT[i].weight);
printf("%d:-->%s ", HT[i].weight, p);

printf("\n");
}
free(cd);
return OK;
}

Status Select(int n, int *s1, int *s2, int *foot)
{
    int i, j;
    int min = 65536;
    int temp = 0;
    for(i = 1; i <= n; i++) //找到第一个最小的值的下标
    {
        if(HT[i].weight < min && foot[i] == 0)
        {
            min = HT[i].weight;
            temp = i; //记录最终的最小值的下标
        }
    }
    foot[temp] = 1;
    *s1 = temp;

    min = 65536;
    for(j = 1; j <= n; j++) //寻找最小节点下标
    {
        if(HT[j].weight < min && j != (*s1) && foot[j] == 0)
        {
            min = HT[j].weight;
            temp = j;
        }
    }
    foot[temp] = 1;
    *s2 = temp;
    //交换*s1和*s2, 让*s1指向小的坐标 *s2指向大的坐标
    temp = *s1;
    *s1 = *s2;
    *s2 = temp;
    return OK;
}
```

exp_04_01.c

```
//
// exp_04_01.c
// exp_04
//
// Created by Toxni on 12/28/15.
// Copyright © 2015 Toxni. All rights reserved.
//

#include "exp_04_01.h"

int main()
{
    BiTree T;

    printf("please input the value of first node, -1 means no node here:\n");
    CreateBiTree(&T);
    printf("preorder traversal:\n");
    TraverseBiTree(T);
    printf("\n");
    printf("inorder traversal:\n");
    InOrderBiTree(T);
    printf("\n");
    printf("preorder traversal:\n");
    PostOrderBiTree(T);
    printf("\n");
    return 0;
}
```

exp_04_02.c

```
//
// exp_04_02.c
// exp_04
//
// Created by Toxni on 12/28/15.
// Copyright © 2015 Toxni. All rights reserved.
//

#include "exp_04_02.h"

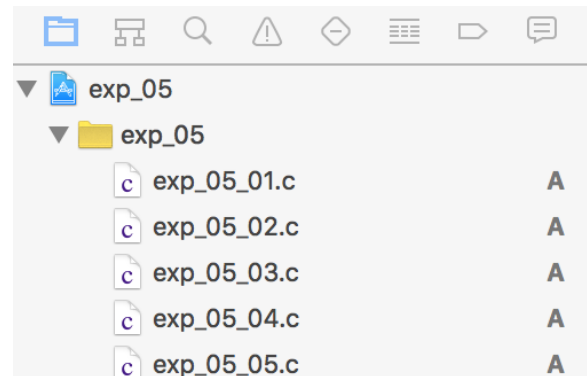
int main() {
    int n;
    printf("Input the number of weight:\n");
    scanf("%d",&n);
    unsigned int *w=(unsigned int *)malloc(sizeof(unsigned int)*n);
    printf("Input The weight!\n");

    unsigned s;
    unsigned *p=w;
    int i = 1;
    // unsigned int p[8]={5,29,7,8,14,23,3,11};
    while(i<=n)
    {
        scanf("%ud",&s);
        *p=s;
        i++;
        p++;
    }
    p=w;
    HuffmanCode *HC=(HuffmanCode *)malloc(sizeof(HuffmanCode));

    HuffmanCoding(HC,n,p);
    return 0;
}
```

五 查找与排序 代码如下：

文件结构如右图，exp_05_01.c 存放顺序查找函数；exp_05_02.c 存放二分法查找函数；exp_05_03.c 存放顺序排序函数；exp_05_04.c 存放简单插入排序函数；exp_05_05.c 存放冒泡发排序函数。



exp_05_01.c
// 顺序查找

```
//
// main.c
// exp_05
//
// Created by Toxni on 12/28/15.
// Copyright © 2015 Toxni. All rights reserved.
//

#include <stdio.h>
#include <stdlib.h>
#include <memory.h>

int ordersearch(int a[], int n, int des){
    int i;
    for(i=0; i<n; i++){
        if(des==a[i])
            return 1;
    }
    return 0;
}

int main(){
    int i, val;
    int a[] = {3, 10, 13, 17, 40, 45, 50, 70};
    int ret;

    for (i=0; i<8; i++){
        printf("%d\t", a[i]);
    }

    printf("\nplease input the aim: ");
    while (1){
        scanf("%d", &val);

        if (val == -1) {
            break;
        }
        ret = ordersearch(a, 8, val);

        if(1 == ret)
            printf ("found it");
        else
            printf ("do not exist");

        printf("\nplease input the aim, input -1 to stop: ");
    }
    return 0;
}
```

exp_05_02.c

// 二分法查找

```
//
// exp_05_02.c
// exp_05
//
// Created by Toxni on 12/28/15.
// Copyright © 2015 Toxni. All rights
// reserved.
//

#include <stdio.h>

int binarySearch(int a[], int n, int key)
{
    int low = 0;
    int high = n - 1;
    while(low <= high){
        int mid = (low + high)/2;
        int midVal = a[mid];
        if(midVal < key)
            low = mid + 1;
        else if(midVal > key)
            high = mid - 1;
        else
            return mid;
    }
    return -1;
}

int main(){
    int i, val, ret;
    int a[8]={3, 10, 13, 17, 40, 45, 50,
70};
    for(i=0; i<8; i++)
        printf("%d\t", a[i]);

    printf("\nplease input the aim: ");

    while (1){
        scanf("%d",&val);
        if (val == -1) {
            break;
        }

        ret = binarySearch(a,8,val);

        if (-1 == ret)
            printf("do not exist\n");
        else
            printf("found it\n");

        printf("\nplease input the aim,
input -1 to stop: ");
    }

    return 0;
}
```

exp_05_03.c

// 选择法排序

```
//
// exp_05_03.c
// exp_05
//
// Created by Toxni on 12/28/15.
// Copyright © 2015 Toxni. All rights
// reserved.
//

#include <stdio.h>

void selectSort(int arr[], int len) {
    int i,j,temp;
    for(i = 0; i < len; i++)
    {
        for(j = i + 1; j < len; j++)
        {
            if(arr[i]>arr[j])
            {
                temp=arr[i];
                arr[i]=arr[j];
                arr[j]=temp;
            }
        }
    }
}

int main() {
    int i;
    int a[10]={513, 87, 512, 61, 908,
170, 897, 275, 653, 462};
    printf("arr before sort is: \n");
    for (i = 0; i < 10; i++) {
        printf("%d ",a[i]);
    }

    selectSort(a, 10);

    printf("\narr after sort is: \n");
    for (i = 0; i < 10; i++) {
        printf("%d ",a[i]);
    }
    printf("\n");
}
```

exp_05_04.c

// 插入法排序

```
//
// exp_05_04.c
// exp_05
//
// Created by Toxni on 12/28/15.
// Copyright © 2015 Toxni. All rights
// reserved.
//

#include <stdio.h>

void insertSort(int arr[],int len)
{
    int i,j;
    int key;
    for(i=1;i<len;i++)
    {
        key=arr[i];
        for(j=i-1;j>=0;j--)
        {
            if(arr[j]>key)
                arr[j+1]=arr[j];
            else
                break;
        }
        arr[j+1]=key;
    }
}

int main() {
    int i;
    int a[10]={513, 87, 512, 61, 908,
170, 897, 275, 653, 462};
    printf("arr before sort is: \n");
    for (i = 0;i < 10;i++) {
        printf("%d ",a[i]);
    }

    insertSort(a, 10);

    printf("\narr after sort is: \n");
    for (i = 0;i < 10;i++) {
        printf("%d ",a[i]);
    }
    printf("\n");
}
```

exp_05_05.c

// 冒泡法排序

```
//
// exp_05_05.c
// exp_05
//
// Created by Toxni on 12/28/15.
// Copyright © 2015 Toxni. All rights
// reserved.
//

#include <stdio.h>

void bubbleSort(int arr[],int len)
{
    int i,j,temp;
    for(i=0;i<len;i++)
    {
        for(j=0;j<len-i-1;j++)
        {
            if(arr[j]>arr[j+1])
            {
                temp=arr[j];
                arr[j]=arr[j+1];
                arr[j+1]=temp;
            }
        }
    }
}

int main() {
    int i;
    int a[10]={513, 87, 512, 61, 908, 170,
897, 275, 653, 462};
    printf("arr before sort is: \n");
    for (i = 0;i < 10;i++) {
        printf("%d ",a[i]);
    }

    bubbleSort(a, 10);

    printf("\narr after sort is: \n");
    for (i = 0;i < 10;i++) {
        printf("%d ",a[i]);
    }
    printf("\n");
}
```