**武汉纺织大学**

**《数据结构》课程设计报告**

**模拟竞价系统**

**成 绩：**

**学 号：1804240417 1804240127**

**1804240224 1809440426**

**1804240317 1813220629**

**1804240231**

**姓 名：陈宇航 赵子航 梁洪源 尹逸凡**

**田祺 杭子奇 曾时铸**

**班 级： 物联网11801**

**指导教师： 叶 鹏**

**报告日期：2019 年 12月31日**

目录

[1. 题目与要求 3](#_Toc24554759)

[1.1. 问题描述 3](#_Toc24554760)

[1.2. 本系统涉及的知识点 3](#_Toc24554761)

[1.3. 功能要求 3](#_Toc24554762)

[2. 功能设计 3](#_Toc24554763)

[2.1. 数据结构定义 3](#_Toc24554764)

[2.2. 模块图 4](#_Toc24554765)

[3. 功能代码 4](#_Toc24554766)

[4. 调试与测试 1](#_Toc24554767)3

[4.1. 调试分析 1](#_Toc24554768)3

[4.2. 用户手册 1](#_Toc24554769)4

[4.3. 测试过程 1](#_Toc24554770)5

[5. 总结 1](#_Toc24554771)6

[6. 参考文献 1](#_Toc24554772)8

[7. 附录 1](#_Toc24554773)9

[8. 项目分工 32](#_Toc24554774)

# 1.题目与要求

## 问题描述

某公司欲新推一款产品，宣称因产能所限，首批货将采取客户通过手机号注册后竞拍的形式。当有5万名用户参与竞价后，再根据实际产量来确定中标者，价高者得（若报价相同，先出价者得）。注意：产品数量在竞拍期间未知，竞价完毕根据备货量确定！

为了激励高价，系统即时反馈当前的最高报价。为防止恶意虚报高价，破坏竞拍，管理员不定时地监控当前最高价，并与可疑虚报者确认，若确认虚报，则删除当前最高价。竞价结束后，备货数量可能是 1 台也许是 100 万台。

现为该公司设计这样一个系统，来模拟竞拍过程。随机生成测试数据（报价信息或删除命令）保存在文本文件中，系统将数据文件作为输入。并统计系统整个运行过程中的比较、移动（赋值） 操作的次数。

## 本系统涉及的知识点

快速排序、折半查找

## 功能要求

接受报价模块：添加一条报价信息，并显示当前最高价；

删除报价模块：删除当前最高报价信息；

输出竞拍结果：根据给定产品数量，列出中标客户的信息（手机号及报价）；

# **2.功能设计**

## 2.1.数据结构定义

typedef struct Bidder

{ // 表示每一条竞价信息

int phoneNumber; // 竞价信息的序列号(手机号)

int price; // 竞价信息的出价

} Bidder, \*bidNode;

typedef struct

{

Bidder \*bidderGroup;

int length; //当前长度

} SqList;

int ifSorted = 0; // 当前程序内的bidderGroup是否已排序,未排序为 0 ,已排序为 1 ;

int successfulBidders = 0;

int swapCount = 0; //交换次数 ;

int evalCount = 0; //移动与赋值次数 ;

int compCount = 0; //比较次数 ;

## 2.2.模块图

## qt_temp

# 3.功能代码

模块1: 创建和排序数据组

此模块将数据处理完毕后,导入至全局变量数组 bidderGroup 中

模块代码：

bool InitList\_Sq(SqList &L)

{

// 构造一个空的线性表L

L.bidderGroup = new Bidder[MAXSIZE];

if (!L.bidderGroup)

exit(OVERFLOW); // 分配空间失败

L.length = 0; //表示长度为 0

return OK;

// 由于函数运行至return就已经截止, 以下部分并不会执行, 仅作为后续函数编写时的参考, 便于诸位理解

L.bidderGroup[1].phoneNumber = 0000; // 表示当前读取的竞价表中第 1 条竞价信息的手机序列号赋值为 0000

L.bidderGroup[1].price = 1000; // 表示当前读取的竞价表中第 1 条竞价信息的出价为 1000

L.length = 100; // 表示当前竞价表中有 100 条竞价信息

// 示例:如何交换 1 号竞价者和 2 号竞价者在竞价表"bidderGroup"中的位置

Bidder T; // 新建一个临时竞价信息(临时变量)

T = L.bidderGroup[1];

L.bidderGroup[1] = L.bidderGroup[2];

L.bidderGroup[2] = T;

// 对比常见的交换变量 a 和 b 的值

int t, a = 1, b = 2;

t = a;

a = b;

b = t;

} // InitList\_Sq

// -----------------------------------

// Create/Initialize the BidderGroup

// -----------------------------------

bool createTheBidderGroup(SqList &L)

{

InitList\_Sq(L);

int i, n;

cout << "\nEnter the number of bidders: ";

cin >> n;

L.length = n;

srand((unsigned int)time(0)); // Initialize the 'randseed'

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

{

L.bidderGroup[i].phoneNumber = i + 1000;

L.bidderGroup[i].price = rand() % 9001 + 1000;

evalCount += 2;

}

ifSorted = 0;

return OK;

} // createTheBidderGroup

// -----------------------------------

// Print the BidderGroup

// -----------------------------------

void printTheAllBidderGroup(SqList L)

{

int i;

if (L.length == 0)

{

cout << "\nThere is no bid." << endl;

return;

}

cout << "\nThe BidderList:" << endl;

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

{

printf("Num %04d. ", i);

cout << "Phone:" << L.bidderGroup[i].phoneNumber << setw(10) << "Price:" << L.bidderGroup[i].price << endl;

}

cout << "\nFinished.\n"

<< endl;

} // printTheAllBidderGroup

// -----------------------------------

// SORTTING

// -----------------------------------

void ECore\_Swap(Bidder &A, Bidder &B)

{

Bidder T;

T = A;

A = B;

B = T;

++swapCount;

} // ECore\_Swap

void ECore\_Qsort(Bidder arr[], int low, int high){

if (high <= low) return;

int i = low;

int j = high + 1;

int key = arr[low].price;

while (true)

{

/\*从左向右找比key大的值\*/

while (arr[++i].price < key)

{

if (i == high){

break;

}

++compCount;

}

/\*从右向左找比key小的值\*/

while (arr[--j].price > key)

{

if (j == low){

break;

}

++compCount;

}

if (i >= j) break;

/\*交换i,j对应的值\*/

ECore\_Swap(arr[i], arr[j]);

}

/\*中枢值与j对应值交换\*/

ECore\_Swap(arr[low], arr[j]);

ECore\_Qsort(arr, low, j - 1);

ECore\_Qsort(arr, j + 1, high);

} // ECore\_Qsort

void sortTheAllBidderList(SqList &L)

{

if (L.length == 0)

{

cout << "\nThere is no bid.\n" << endl;

return;

}

ECore\_Qsort(L.bidderGroup, 0, L.length);

ifSorted = 1;

cout << "\nSort Finished.\n" << endl;

} // sortTheAllBidderList

模块2竞价信息组的数据动态管理

编写函数,实现全局变量数组 bidderGroup 的插入/删除操作

模块代码：

void insertBidder\_1(SqList &L, Bidder newBidder)

{

L.bidderGroup[L.length + 1] = newBidder;

++evalCount;

++L.length;

} // insertBidder\_1

void insertBidder\_2(SqList &L, Bidder newBidder)

{

int low = 1, high = L.length, m, j;

while (low <= high)

{

m = (low + high) / 2;

if (L.bidderGroup[m].price > newBidder.price)

high = m - 1;

else

low = m + 1;

++compCount;

} // while

for (j = L.length; j >= high; --j)

{

L.bidderGroup[j + 1] = L.bidderGroup[j];

++evalCount;

}

L.bidderGroup[low] = newBidder;

++evalCount;

++L.length;

} // insertBidder\_2

void insertBidder(SqList &L)

{

Bidder newBidder;

do

{

cout << "\nEnter the biider's price(1000~9999):";

cin >> newBidder.price;

cout << "\nEnter the bidder's phone(1000~9999):";

cin >> newBidder.phoneNumber;

} while (!(newBidder.phoneNumber >= 1000 && newBidder.phoneNumber <= 9999 && newBidder.price >= 1000 && newBidder.price <= 9999));

if (ifSorted == 0)

insertBidder\_1(L, newBidder);

if (ifSorted == 1)

insertBidder\_2(L, newBidder);

} // insertBidder

void deleteBidder(SqList &L)

{

int t, i = 1, j;

if (L.length == 0)

{

cout << "\nThere is no bid." << endl;

return;

}

cout << "Enter the bidder's phone:";

cin >> t;

while (TRUE)

{

if (L.bidderGroup[i].phoneNumber != t)

i++;

else

{

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

{

L.bidderGroup[i] = L.bidderGroup[i + 1];

++evalCount;

}

--L.length;

cout << "\nDelete Finished.\n"

<< endl;

return;

}

if (i > L.length)

break;

}

cout << "\nBidder not found.\n"

<< endl;

} // deleteBidder

// -----------------------------------

// Choose insert or delete

// -----------------------------------

void insertOrDelete(SqList &L)

{

int choose = 0;

while (!(choose == 1 || choose == 2))

{

cout << "\n1. Insert a bid\n"

<< "2. Delete a bid\n\n"

<< "Choose:";

cin >> choose;

}

if (choose == 1)

insertBidder(L);

if (choose == 2)

deleteBidder(L);

} // insertOrDelete

模块3：文件的载入和导出

编写函数, 实现:

1.将已有的外部数据文件载入全局变量数组 bidderGroup

2.将程序内的 bidderGroup数据导出至外部文件

模块代码：

// -----------------------------------

// Output File

// -----------------------------------

void outputTheAllbidderList(SqList L)

{

int i;

if (L.length == 0)

{

cout << "\nThere is no bid." << endl;

return;

}

FILE \*fp;

fp = fopen("BidderList", "w+");

fprintf(fp, "This is the List of bidders\nWARNING: DO NOT MODIFY THE FORMAT!\n\n");

fprintf(fp, "Length: %d\n\n", L.length);

fprintf(fp, "ifSorted: %d\n\n", ifSorted);

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

{

fprintf(fp, "Num %04d. Phone: %d Price: %d\n", i, L.bidderGroup[i].phoneNumber, L.bidderGroup[i].price);

}

fclose(fp);

cout << "\nOutput Finished.\n\n"

<< endl;

} // outputTheAllbidderList

// -----------------------------------

// Load File

// -----------------------------------

void loadTheAllbidderList(SqList &L)

{

InitList\_Sq(L);

int i, t;

FILE \*fp;

if ((fp = fopen("BidderList", "r")) == NULL)

{

cout << "\nFile not found!\n"

<< endl;

return;

}

fscanf(fp, "This is the List of bidders\nWARNING: DO NOT MODIFY THE FORMAT!\n\n");

fscanf(fp, "Length: %d\n\n", &L.length);

fscanf(fp, "ifSorted: %d\n\n", &ifSorted);

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

{

fscanf(fp, "Num %04d. Phone: %d Price: %d\n", &t, &L.bidderGroup[i].phoneNumber, &L.bidderGroup[i].price);

}

fclose(fp);

cout << "\nLoad Finished.\n\n"

<< endl;

} // loadTheAllbidderList

// -----------------------------------

// DESTROY THEM ALL!Wryyyyyyyyyyyyyy!!!

// -----------------------------------

void destroyTheTemp(SqList &L)

{

delete L.bidderGroup;

L.length = 0;

} // destroyTheTemp

void destroyTheFile(void)

{

FILE \*fp;

if ((fp = fopen("BidderList", "r")) == NULL)

{

cout << "\nFile not found!\n"

<< endl;

return;

}

fclose(fp);

system("del BidderList");

cout << "\nDelete Finished.\n"

<< endl;

} // destroyTheFile

// -----------------------------------

// SuccessfulBidders

// -----------------------------------

void setTheNumberOfSuccessfulBidders(SqList L)

{

if (L.length == 0)

{

cout << "\nThere is no bid\n"

<< endl;

return;

}

do

{

cout << "\nEnter the numbers of successful bidders(Now: " << L.length << " bidders):";

cin >> successfulBidders;

} while (successfulBidders < 0 || successfulBidders > L.length);

cout << "\nSet Finished.\n"

<< endl;

} // SetTheNumberOfSuccessfulBidders

bool outputTheListOfSuccessfulBidders(SqList L)

{

if (L.length == 0)

{

cout << "\nThere is no bid\n"

<< endl;

return ERROR;

}

if (ifSorted == 0)

{

cout << "\nThe List have not sorted.\n"

<< endl;

return ERROR;

}

if (successfulBidders <= 0 || successfulBidders > L.length)

setTheNumberOfSuccessfulBidders(L);

FILE \*fp;

int i;

fp = fopen("SucBidderList", "w+");

fprintf(fp, "This is the List of Successful bidders\n\n");

fprintf(fp, "Length: %d\n\n", successfulBidders);

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

{

fprintf(fp, "Num %04d. Phone: %d Price: %d\n", i, L.bidderGroup[L.length - i].phoneNumber, L.bidderGroup[L.length - i].price);

}

fclose(fp);

cout << "\nOutput Finished.\n\n"

<< endl;

return OK;

} // outputTheListOfSuccessfulBidders

# 4.调试与测试

## 4.1.调试分析

本程序在调试过程中较为顺利，主要算法为折半查找和快速排序，其时间复杂度分别为O(log2n)和O (nlogn)。

## 4.2.用户手册

## 本系统为模拟竞价系统，当程序运行后，用户可以在终端根据菜单输入相应数字从而获取自己想要功能。

## 输入1后，系统会导入已有竞价表。

## 输入2后，再输入你想生成的竞价信息的个数，便能随机生成指定数目的竞价信息,每一条信息包含一个手机号(序列号)和一个出价。

## 输入3后，将当前程序中的竞价表导入文件。

## 输入4后，系统会将当前程序中的竞价表从小到大排序，并输出当前最高的价格以及价格所属竞价者的电话号码。

## 输入5后，系统会输出所有的竞价信息，以便查看。

## 输入6后会出现两个选项，选项1为插入，选项2为删除。输入1后，再输入你想插入的竞价者的价格和号码即可将数据插入其中；输入2后，再输入你想删除的竞价者的号码，便可以删除相应的竞价信息。

## 输入7后，输入数字即可自定义中标的人数。

## 输入8后，系统会生成一个包含中标者的信息的文件。

## 输入9后，系统会销毁竞价表。

## 输入0后会退出该系统。

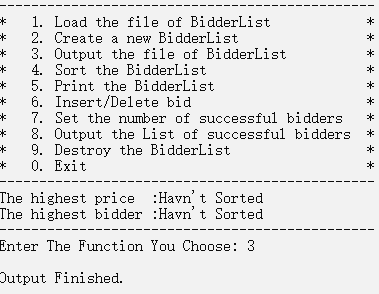
## 【注】每次操作完成后，系统都会输出当前价格最高的竞价者。

## 4.3.测试过程

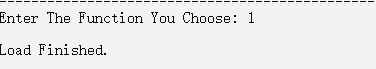
## 输入2，输15，让系统生成15组随机数据

## 

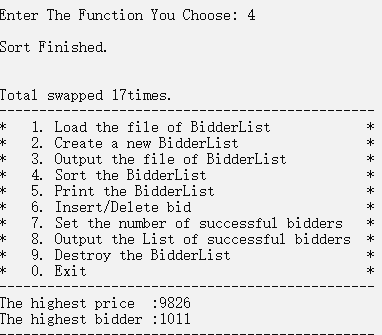
输入3，保存文件



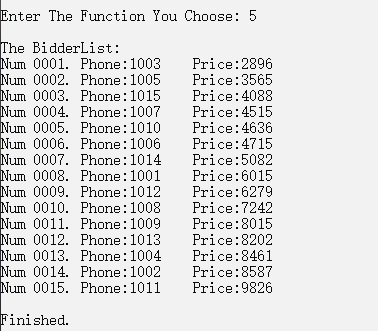
输入1，导入文件



输入4，将数据排序



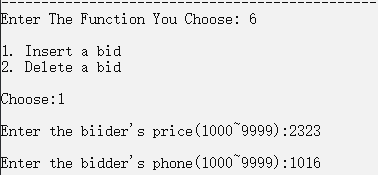
输入5，输出数据，15组数据已按照价格从小到大的顺序排好

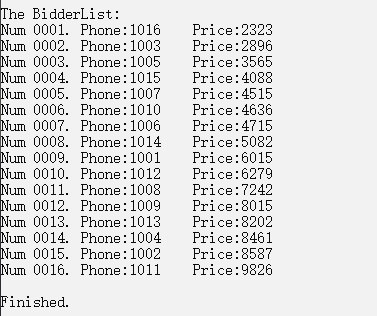


输入6，选择1，插入数据。

输入价格：2323；电话号码：1016。

并且再次输入5，输出数据，可以看到该数据已被插入。

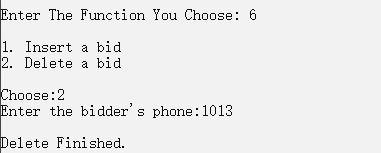




再次输入6，选择2，删除数据。

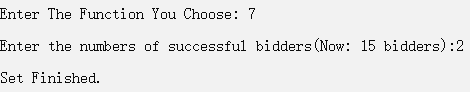
输入要删除的号码1013，删除完毕。

再次输出数据，可以看到该数据已被删除。

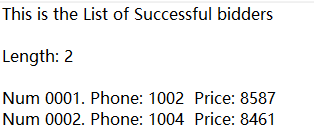




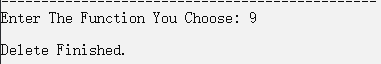
输入7，确定中标人数为2，即输入2。



输入8，生成一个包含中标者信息的文件，结果如图：



最后输入9，销毁文件



# 总结

在本次课程设计中，大家经过讨论很快地确定了题目以及实现题目所需要的主要算法，随后分配任务，每个人负责相应的模块。在接下来的几天里大家一起讨论、分析，经过几天的努力，顺利得到了最终的代码。代码经过调试后，每个部分都能运行并实现相应的功能。此次课程设计让大家对于排序和查找这方面的知识有了更深层次的了解。

# 6.参考文献

[1] 严蔚敏，吴伟民.数据结构（C语言版）[M]. 北京：清华大学出版社，1997.

[2] 严薇敏，吴卫民.数据结构题集（C 语言版）[M].北京：清华大学出版社，1997.

[3] 谭浩强.C程序设计（第二版）[M]. 北京：清华大学出版社，2001.

# 7.附录

# 

bool InitList\_Sq(SqList &L)

{

    // 构造一个空的线性表L

    L.bidderGroup = new Bidder[MAXSIZE];

    if (!L.bidderGroup)

        exit(OVERFLOW); // 分配空间失败

    L.length = 0; //表示长度为 0

    return OK;

    // 由于函数运行至return就已经截止, 以下部分并不会执行, 仅作为后续函数编写时的参考, 便于诸位理解

    L.bidderGroup[1].phoneNumber = 0000; // 表示当前读取的竞价表中第 1 条竞价信息的手机序列号赋值为 0000

    L.bidderGroup[1].price = 1000;       // 表示当前读取的竞价表中第 1 条竞价信息的出价为 1000

    L.length = 100;                      // 表示当前竞价表中有 100 条竞价信息

    // 示例:如何交换 1 号竞价者和 2 号竞价者在竞价表"bidderGroup"中的位置

    Bidder T; // 新建一个临时竞价信息(临时变量)

    T = L.bidderGroup[1];

    L.bidderGroup[1] = L.bidderGroup[2];

    L.bidderGroup[2] = T;

    // 对比常见的交换变量 a 和 b 的值

    int t, a = 1, b = 2;

    t = a;

    a = b;

    b = t;

} // InitList\_Sq

// -----------------------------------

// Create/Initialize the BidderGroup

// -----------------------------------

bool createTheBidderGroup(SqList &L)

{

    InitList\_Sq(L);

    int i, n;

    cout << "\nEnter the number of bidders: ";

    cin >> n;

    L.length = n;

    srand((unsigned int)time(0)); // Initialize the 'randseed'

    for (i = 1; i <= n; ++i)

    {

        L.bidderGroup[i].phoneNumber = i + 1000;

        L.bidderGroup[i].price = rand() % 9001 + 1000;

        evalCount += 2;

    }

    ifSorted = 0;

    return OK;

} // createTheBidderGroup

// -----------------------------------

// Print the BidderGroup

// -----------------------------------

void printTheAllBidderGroup(SqList L)

{

    int i;

    if (L.length == 0)

    {

        cout << "\nThere is no bid." << endl;

        return;

    }

    cout << "\nThe BidderList:" << endl;

    for (i = 1; i <= L.length; ++i)

    {

        printf("Num %04d. ", i);

        cout << "Phone:" << L.bidderGroup[i].phoneNumber << setw(10) << "Price:" << L.bidderGroup[i].price << endl;

    }

    cout << "\nFinished.\n"

         << endl;

} // printTheAllBidderGroup

// -----------------------------------

// SORTTING

// -----------------------------------

void ECore\_Swap(Bidder &A, Bidder &B)

{

    Bidder T;

    T = A;

    A = B;

    B = T;

    ++swapCount;

} // ECore\_Swap

void ECore\_Qsort(Bidder arr[], int low, int high){

    if (high <= low) return;

    int i = low;

    int j = high + 1;

    int key = arr[low].price;

    while (true)

    {

        /\*从左向右找比key大的值\*/

        while (arr[++i].price < key)

        {

            if (i == high){

                break;

            }

            ++compCount;

        }

        /\*从右向左找比key小的值\*/

        while (arr[--j].price > key)

        {

            if (j == low){

                break;

            }

            ++compCount;

        }

        if (i >= j) break;

        /\*交换i,j对应的值\*/

        ECore\_Swap(arr[i], arr[j]);

    }

    /\*中枢值与j对应值交换\*/

    ECore\_Swap(arr[low], arr[j]);

    ECore\_Qsort(arr, low, j - 1);

    ECore\_Qsort(arr, j + 1, high);

} // ECore\_Qsort

void sortTheAllBidderList(SqList &L)

{

    if (L.length == 0)

    {

        cout << "\nThere is no bid.\n" << endl;

        return;

    }

    ECore\_Qsort(L.bidderGroup, 0, L.length);

    ifSorted = 1;

    cout << "\nSort Finished.\n" << endl;

} // sortTheAllBidderList

// ----------------------------

// EXTERNAL INCLUDE

// ----------------------------

#include <iostream>

#include <iomanip>

#include <time.h>

#include <string.h>

using namespace std;

// ----------------------------

// DEFINE

// ----------------------------

#define MAXSIZE 3000

#define TRUE 1

#define FALSE 0

#define ERROR 0

#define OVERFLOW -1

#define OK 1

// ----------------------------

// STRUCT

// ----------------------------

typedef struct Bidder

{                    // 表示每一条竞价信息

    int phoneNumber; // 竞价信息的序列号(手机号)

    int price;       // 竞价信息的出价

} Bidder, \*bidNode;

typedef struct

{

    Bidder \*bidderGroup;

    int length; //当前长度

} SqList;

int ifSorted = 0; // 当前程序内的bidderGroup是否已排序,未排序为 0 ,已排序为 1 ;

int successfulBidders = 0;

int TestMod = 0;    // 开发者模式

int swapCount = 0;  // 交换次数 ;

int evalCount = 0;  // 移动与赋值次数 ;

int compCount = 0;  // 比较次数 ;

// ----------------------------

// FILE INCLUDE

// ----------------------------

#include "CreateAndSort.h"

#include "InsertAndDelete.h"

#include "LoadFileAndOutput.h"

void insertBidder\_1(SqList &L, Bidder newBidder)

{

    L.bidderGroup[L.length + 1] = newBidder;

    ++evalCount;

    ++L.length;

} // insertBidder\_1

void insertBidder\_2(SqList &L, Bidder newBidder)

{

    int low = 1, high = L.length, m, j;

    while (low <= high)

    {

        m = (low + high) / 2;

        if (L.bidderGroup[m].price > newBidder.price)

            high = m - 1;

        else

            low = m + 1;

        ++compCount;

    } // while

    for (j = L.length; j >= high; --j)

    {

        L.bidderGroup[j + 1] = L.bidderGroup[j];

        ++evalCount;

    }

    L.bidderGroup[low] = newBidder;

    ++evalCount;

    ++L.length;

} // insertBidder\_2

void insertBidder(SqList &L)

{

    Bidder newBidder;

    do

    {

        cout << "\nEnter the biider's price(1000~9999):";

        cin >> newBidder.price;

        cout << "\nEnter the bidder's phone(1000~9999):";

        cin >> newBidder.phoneNumber;

    } while (!(newBidder.phoneNumber >= 1000 && newBidder.phoneNumber <= 9999 && newBidder.price >= 1000 && newBidder.price <= 9999));

    if (ifSorted == 0)

        insertBidder\_1(L, newBidder);

    if (ifSorted == 1)

        insertBidder\_2(L, newBidder);

} // insertBidder

// -----------------------------------

// DeleteBidder

// -----------------------------------

void deleteBidder(SqList &L)

{

    int t, i = 1, j;

    if (L.length == 0)

    {

        cout << "\nThere is no bid." << endl;

        return;

    }

    cout << "Enter the bidder's phone:";

    cin >> t;

    while (TRUE)

    {

        if (L.bidderGroup[i].phoneNumber != t)

            i++;

        else

        {

            for (; i < L.length; ++i)

            {

                L.bidderGroup[i] = L.bidderGroup[i + 1];

                ++evalCount;

            }

            --L.length;

            cout << "\nDelete Finished.\n"

                 << endl;

            return;

        }

        if (i > L.length)

            break;

    }

    cout << "\nBidder not found.\n"

         << endl;

} // deleteBidder

// -----------------------------------

// Choose insert or delete

// -----------------------------------

void insertOrDelete(SqList &L)

{

    int choose = 0;

    while (!(choose == 1 || choose == 2))

    {

        cout << "\n1. Insert a bid\n"

             << "2. Delete a bid\n\n"

             << "Choose:";

        cin >> choose;

    }

    if (choose == 1)

        insertBidder(L);

    if (choose == 2)

        deleteBidder(L);

} // insertOrDelete

void outputTheAllbidderList(SqList L)

{

    int i;

    if (L.length == 0)

    {

        cout << "\nThere is no bid." << endl;

        return;

    }

    FILE \*fp;

    fp = fopen("BidderList", "w+");

    fprintf(fp, "This is the List of bidders\nWARNING: DO NOT MODIFY THE FORMAT!\n\n");

    fprintf(fp, "Length: %d\n\n", L.length);

    fprintf(fp, "ifSorted: %d\n\n", ifSorted);

    for (i = 1; i <= L.length; ++i)

    {

        fprintf(fp, "Num %04d. Phone: %d  Price: %d\n", i, L.bidderGroup[i].phoneNumber, L.bidderGroup[i].price);

    }

    fclose(fp);

    cout << "\nOutput Finished.\n\n"

         << endl;

} // outputTheAllbidderList

// -----------------------------------

// Load File

// -----------------------------------

void loadTheAllbidderList(SqList &L)

{

    InitList\_Sq(L);

    int i, t;

    FILE \*fp;

    if ((fp = fopen("BidderList", "r")) == NULL)

    {

        cout << "\nFile not found!\n"

             << endl;

        return;

    }

    fscanf(fp, "This is the List of bidders\nWARNING: DO NOT MODIFY THE FORMAT!\n\n");

    fscanf(fp, "Length: %d\n\n", &L.length);

    fscanf(fp, "ifSorted: %d\n\n", &ifSorted);

    for (i = 1; i <= L.length; ++i)

    {

        fscanf(fp, "Num %04d. Phone: %d  Price: %d\n", &t, &L.bidderGroup[i].phoneNumber, &L.bidderGroup[i].price);

    }

    fclose(fp);

    cout << "\nLoad Finished.\n\n"

         << endl;

} // loadTheAllbidderList

// -----------------------------------

// DESTROY THEM ALL!Wryyyyyyyyyyyyyy!!!

// -----------------------------------

void destroyTheTemp(SqList &L)

{

    delete L.bidderGroup;

    L.length = 0;

} // destroyTheTemp

void destroyTheFile(void)

{

    FILE \*fp;

    if ((fp = fopen("BidderList", "r")) == NULL)

    {

        cout << "\nFile not found!\n"

             << endl;

        return;

    }

    fclose(fp);

    system("del BidderList");

    cout << "\nDelete Finished.\n"

         << endl;

} // destroyTheFile

// -----------------------------------

// SuccessfulBidders

// -----------------------------------

void setTheNumberOfSuccessfulBidders(SqList L)

{

    if (L.length == 0)

    {

        cout << "\nThere is no bid\n"

             << endl;

        return;

    }

    do

    {

        cout << "\nEnter the numbers of successful bidders(Now: " << L.length << " bidders):";

        cin >> successfulBidders;

    } while (successfulBidders < 0 || successfulBidders > L.length);

    cout << "\nSet Finished.\n"

         << endl;

} // SetTheNumberOfSuccessfulBidders

bool outputTheListOfSuccessfulBidders(SqList L)

{

    if (L.length == 0)

    {

        cout << "\nThere is no bid\n"

             << endl;

        return ERROR;

    }

    if (ifSorted == 0)

    {

        cout << "\nThe List have not sorted.\n"

             << endl;

        return ERROR;

    }

    if (successfulBidders <= 0 || successfulBidders > L.length)

        setTheNumberOfSuccessfulBidders(L);

    FILE \*fp;

    int i;

    fp = fopen("SucBidderList", "w+");

    fprintf(fp, "This is the List of Successful bidders\n\n");

    fprintf(fp, "Length: %d\n\n", successfulBidders);

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

    {

        fprintf(fp, "Num %04d. Phone: %d  Price: %d\n", i, L.bidderGroup[L.length - i].phoneNumber, L.bidderGroup[L.length - i].price);

    }

    fclose(fp);

    cout << "\nOutput Finished.\n\n"

         << endl;

    return OK;

} // outputTheListOfSuccessfulBidders

// -----------------------------------

// TestMod

// -----------------------------------

// void setTestMod(void)

// {

//     char s[13] = {"Kibo no hana"};

//     char a[20];

//     printf("The key: ");

//     fgets(a, 13, stdin);

//     if (strcmp(s, a) == 0)

//     {

//         TestMod = 1;

//         cout << "\nSucceed." << endl;

//     }

// }

void msg(SqList L)

{

    cout

        << "-----------------------------------------------\n"

        << "\*   1. Load the file of BidderList            \*\n"

        << "\*   2. Create a new BidderList                \*\n"

        << "\*   3. Output the file of BidderList          \*\n"

        << "\*   4. Sort the BidderList                    \*\n"

        << "\*   5. Print the BidderList                   \*\n"

        << "\*   6. Insert/Delete bid                      \*\n"

        << "\*   7. Set the number of successful bidders   \*\n"

        << "\*   8. Output the List of successful bidders  \*\n"

        << "\*   9. Destroy the BidderList                 \*\n"

        << "\*   0. Exit                                   \*\n"

        << "-----------------------------------------------\n"

        << "The highest price  :";

    (ifSorted == 1) ? (printf("%d\n", L.bidderGroup[L.length].price)) : (printf("Havn't Sorted\n"));

    cout

        << "The highest bidder :";

    (ifSorted == 1) ? (printf("%d\n", L.bidderGroup[L.length].phoneNumber)) : (printf("Havn't Sorted\n"));

    cout

        << "-----------------------------------------------\n"

        << "Enter The Function You Choose: ";

} // msg

// -----------------------------------

// chooseFuction

// -----------------------------------

void chooseFuction(void)

{

    SqList L;

    InitList\_Sq(L);

    int choose;

    int n;

    do

    {

        msg(L);

        cin >> choose;

        switch (choose)

        {

        case 1:

            loadTheAllbidderList(L);

            break;

        case 2:

            createTheBidderGroup(L);

            break;

        case 3:

            outputTheAllbidderList(L);

            break;

        case 4:

            sortTheAllBidderList(L);

            break;

        case 5:

            printTheAllBidderGroup(L);

            break;

        case 6:

            insertOrDelete(L);

            break;

        case 7:

            setTheNumberOfSuccessfulBidders(L);

            break;

        case 8:

            outputTheListOfSuccessfulBidders(L);

            break;

        case 9:

            destroyTheFile();

            destroyTheTemp(L);

            break;

        case 0:

            exit(1);

            // case -1:

            //     setTestMod();

            break;

        }

    } while (choose != 0);

} // chooseFuction

// -----------------------------------

// THE START

// -----------------------------------

int main(int argc, char const \*argv[])

{

    chooseFuction();

    return 0;

} // main

# 8.项目分工

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **序号** | **成员姓名** | **承担任务** | **完成情况** | **评分** |
| **1** | **陈宇航** | **数据生成与排序模块** | **良好** | **B** |
| **2** | **杭子奇** | **文件导入输出模块**  **总体框架搭建**  **代码的整合和调试** | **优秀** | **A** |
| **3** | **曾时铸** | **总操作次数统计模块,代码整理 与 界面修饰** | **良好** | **B** |
| **4** | **尹逸凡** | **插入/删除数据模块**  **课程实验报告**  **答辩ppt** | **良好** | **B** |
| **5** | **梁洪源** | **总操作次数统计模块,代码整理 与 界面修饰** | **一般** | **C** |
| **6** | **赵子航** | **数据生成与排序模块** | **一般** | **C** |
| **7** | **田祺** | **数据生成与排序模块** | **一般** | **C** |

小组成员签名： 陈宇航 杭子奇 曾时铸 尹逸凡

梁洪源 田祺 赵子航

日 期： 2020-1-2

**评分表**

**数学与计算机学院课程设计评分表**

课程名称： 数据结构

|  |  |
| --- | --- |
| **考核项目** | **考核分数** |
| 设计方案的合理性与创造性（15分） |  |
| 设计与调试结果（30分） |  |
| 课程设计报告的质量（30分） |  |
| 答辩陈述与回答问题情况（25分） |  |
| **综合成绩** |  |

教师签名：

日 期：