Experiment topic: Design and Realization of Medicine Sales Statistics System in Pharmacy

1. Experimental content

1. Description of the problem: Design a system to realize that pharmaceutical companies regularly collect statistics on the records of sales of each medicine, and can make rankings according to the number, unit price, sales volume or sales of medicines.
2. basic requirements
(1) Medicine information includes: medicine number, medicine name, medicine unit price, sales volume, and sales. Among them, the medicine number has a total of 4 digits, using a mixture of letters and numbers, such as: A125, the first digit is a capital letter, and the last three digits are numbers;
(2) Read the information records of each medicine from the data file and store them in the sequence table;
(3) When sorting by medicine number, the radix sorting method can be used.
(4) When sorting the unit price, sales volume or sales of each medicine, various sorting methods can be used. It is required to use the bubble sort method for the sorting of the unit price, the quick sorting method for the sorting of the sales volume, and the heap sorting method for the sorting of the sales volume.
(5) Design independently according to the requirements of the topic, and write a design report as required after the design is completed.

二、Design ideas:

3. Original code:

```
/*
1. Description of the problem: Design a system to realize that pharmaceutical companies regularly collect statistics on the records of sales of each medicine, and can make rankings according to the number, unit price, sales volume or sales of medicines.
2. basic requirements
(1) Medicine information includes: medicine number, medicine name, medicine unit price, sales volume, and sales. Among them, the medicine number has a total of 4 digits, using a mixture of letters and numbers, such as: A125, the first digit is a capital letter, and the last three digits are numbers;
(2) Read the information records of each medicine from the data file and store them in the sequence table;
(3) When sorting by medicine number, the radix sorting method can be used.
(4) When sorting the unit price, sales volume or sales of each medicine, various sorting methods
```

can be used. It is required to use the bubble sort method for the sorting of the unit price, the quick sorting method for the sorting of the sales volume, and the heap sorting method for the sorting of the sales volume.
(5) Design independently according to the requirements of the topic, and write a design report as required after the design is completed.
*/

```
#include <stdio.h>
#include <stdlib.h>
#include <iostream>
using namespace std;
#define OVERFLOW 0
#define ERROR 0
#define OK 1
#define MAXSIZE 100
#define MAXNUM_KEY 5
#define RADIX 10
#define MAX_SPACE 100
typedef int ArrType[RADIX];


//---------------------------------------------- ----------------------------------//
typedef struct node //Storage structure type definition of medicine information
{
	char num[5]; //medicine number
	char name[30];
	float price; //unit price
	int count; //sales volume
	float sale; //sales
	int next;
}DataType;
typedef struct
{
	DataType *r;
	int length;
}SqList;
typedef struct
{
	DataType r[MAX_SPACE];
	int keynum;
	int recnum;
}SLList;
//---------------------------------------------- ----------------------------------//
```

```
int InitList(SqList &L)
{
	Lr=new DataType[MAXSIZE];
	if(!Lr) exit(OVERFLOW);
	L.length=0;
	return OK;
}
void menu()
{
	printf("\n********************************************** **********\n");
	printf("**** 1--------sort by medicine number****\n");
	printf("**** 2--------Sort by medicine unit price****\n");
	printf("**** 3--------Sort by medicine quantity****\n");
	printf("**** 4-------Sort by medicine sales****\n");
	printf("**** 0--------exit****\n");
	printf("********************************************** **********\n");
	printf("Please enter your choice:");
}
void printresult(SqList &L)
{
	int i;
	printf("The sorting result is (medicine):\nnumber\tname\tunit price\tquantity\tsales\n");
	for(i=1;i<=L.length;i++)
		printf("%s\t%s\t%.2f\t%d\t%.2f\n",Lr[i].num,Lr[i].name,Lr[i].price,Lr[ i].count,Lr[i].sale);

}
void Rad(SqList &L)
{;}
int Read(SqList &L)
{
	int i;
	FILE *fp;
	if((fp=fopen("medicine.txt","r"))==NULL)
	{
		printf("Open the file failure...\n");
		exit(0);
	}
	printf("Read file: medicine.txt\n");
	fscanf(fp,"%d",&L.length);
	printf("The content of the file is (medicine):\nnumber\tname\tunit
price\tquantity\tsales\n");
	for(i=1;i<=L.length;i++) //??????????
	{
		fscanf(fp,"%s",&L.r[i].num);
```

```
            fscanf(fp,"%s",&L.r[i].name);
            fscanf(fp,"%f",&L.r[i].price);
            fscanf(fp,"%d",&L.r[i].count);
            fscanf(fp,"%f",&L.r[i].sale);

            printf("%s\t%s\t%.2f\t%d\t%.2f\n",Lr[i].num,Lr[i].name,Lr[i].price,Lr[ i].count,Lr[i].sale);

      }
      fclose(fp);
      return 0;
}
//---------------------------------------------- ----------------------------------//

void NumSort(SqList &L)
{
      Rad(L);
      printresult(L);
}

//---------------------------------------------- ----------------------------------//
void BubbleSort(SqList &L) //Bubble sort the unit price
{
      int m,flag,j;
      DataType t;
      m=L.length-1;flag=1;
      while((m>0) && (flag==1))
      {
            flag=0;
            for(j=1;j<=m;j++)
                  if(Lr[j].price>Lr[j+1].price)
                  {
                        flag=1;
                        t=Lr[j];
                        Lr[j]=Lr[j+1];
                        Lr[j+1]=t;
                  }
                  --m;
      }
      printresult(L);
}
//---------------------------------------------- ----------------------------------//
// quick sort
int Partition(SqList &L,int low,int high)
{
```

```
int pivotkey;
	Lr[0]=Lr[low];
	pivotkey=Lr[low].count;
	while(low<high)
	{
		while(low<high&&L.r[high].count>=pivotkey) --high;
		Lr[low]=Lr[high];
		while(low<high&&L.r[low].count<=pivotkey) ++low;
		Lr[high]=Lr[low];
	}
	Lr[low]=Lr[0];
	return low;
}

void QSort(SqList &L,int low,int high)
{
	int pivotloc;
	if(low<high)
	{
		pivotloc=Partition(L,low,high);
		QSort(L,low,pivotloc-1);
		QSort(L, pivotloc+1, high);
	}
}
void QuickSort(SqList &L)
{
	QSort(L,0,L. length);
	printresult(L);
}
void updown(SqList &L)
{
	int i;
	DataType e;
	for(i=0;i<L.length/2;i++)
	{
		e=Lr[i];
		Lr[i]=Lr[L.length-i];
		Lr[L.length-i]=e;
	}
}
//---------------------------------------------- ----------------------------------//
void HeapAdjust(SqList &L,int s,int m)
{
	DataType rc;
```

```
    int j;
    rc=Lr[s];
    for(j=2*s;j<=m;j*=2)
    {
        if(j<m&&L.r[j].sale<Lr[j+1].sale) ++j;
        if(rc. sale>=Lr[j]. sale) break;
        Lr[s]=Lr[j];s=j;
    }
    Lr[s]=rc;
}
void CreateHeap(SqList &L)
{
    int i,n;
    n=L.length;
    for(i=n/2;i>0;--i)
        HeapAdjust(L,i,n);
}
void HeapSort(SqList &L)
{
    DataType x;
    int i;
    CreateHeap(L);
    for(i=L.length;i>1;--i)
    {
        x=Lr[1];
        Lr[1]=Lr[i];
        Lr[i]=x;
        HeapAdjust(L,1,i-1);
    }
    printresult(L);
}

//---------------------------------------------- ----------------------------------//
int main()
{
    int m;
    printf("**********************Program Start********************** ***\n");
    SqList L;
    InitList(L);
    Read(L);
    void menu();
    while(1) //loop structure
    {
        menu();
```