

作业 9

必做题

第一题

程序代码：

注 1：两个函数代表了两种算法，*****处代表选择其一填入

注 2：为了调试窗口显示方便，把 100 只羊单独列成二维数组，每十个一组显示

```
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <math.h>
int iAddSheep(short sSheep[1000], short);
int iSortSheepBubble(short sSheep[1000], short, short);
int iSortSheepSelect(short sSheep[1000], short, short);
int main() {
    short sSheep[1000];
    short sSheepMax[10][10];
    short sRound=0;
    iAddSheep(sSheep, 1000);
    iSortSheep***** (sSheep, 1000, 100);
    /*把最重100只羊放进一个数组*/
    for (; sRound<100; sRound++) *(sSheepMax[0]+sRound)=*(sSheep+sRound);
    return 0;
}
/*随机生成一千只羊*/
int iAddSheep(short sSheep[1000], short sAddRoundMax) {
    short sAddRound=0; /*循环变量*/
    short sRandRound; /*循环变量*/
    for (; sAddRound<sAddRoundMax/100; sAddRound++) {
        for (sRandRound=0; sRandRound<(rand()%1000+10000); sRandRound++) {
            srand(time(0));
            printf(" ");
        }
        for (sRandRound=0; sRandRound<100; sRandRound++) {
            srand(time(0)+sRandRound*10000);
            *(sSheep+sAddRound*100+sRandRound)=rand()%999+1;
        }
    }
    return 0;
}
/*冒泡排序选出最大100只*/
int iSortSheepBubble(short sSheep[1000], short sLength, short sSelect) {
    short sRoundFinish, sRoundPosition;
    for (sRoundFinish=0; sRoundFinish<sSelect; sRoundFinish++) {
        for (sRoundPosition=sLength-
```

```

2; sRoundPosition>=sRoundFinish; sRoundPosition--){
    if (*(sSheep+sRoundPosition)<*(sSheep+sRoundPosition+1)) { /*交换*/
        short sTemp=*(sSheep+sRoundPosition);
        *(sSheep+sRoundPosition)=*(sSheep+sRoundPosition+1);
        *(sSheep+sRoundPosition+1)=sTemp;
    }
}
}
return 0;
}
}
/*选择排序选出最大100只*/
int iSortSheepSelect(short sSheep[1000], short sLength, short sSelect){
    short sRoundFinish, sRoundPosition;
    for (sRoundFinish=0; sRoundFinish<sSelect; sRoundFinish++){
        for
(sRoundPosition=sRoundFinish+1; sRoundPosition<sLength; sRoundPosition++){
            if (*(sSheep+sRoundFinish)<*(sSheep+sRoundPosition)) { /*交换*/
                short sTemp=*(sSheep+sRoundFinish);
                *(sSheep+sRoundFinish)=*(sSheep+sRoundPosition);
                *(sSheep+sRoundPosition)=sTemp;
            }
        }
    }
    return 0;
}
}

```

运行结果截图：

法 1——冒泡

```

short sSheepMax[10][10];
short sRound=0;
iAddSheep(sSheep, 1000);
iSortSheepBubble(sSheep, 1000, 100);
/*把最重100只羊放进一个数组*/
for (; sRound<100; sRound++) *(sSheepMax[0]+sRound)=*(sSheep+sRound);
return 0;
}
/*随机生成一千只羊*/
int iAddSheep(short sSheep[1000], short sAddRoundMax){
    short sAddRound=0; /*循环变量*/
    short sRandRound; /*循环变量*/
    for (; sAddRound<sAddRoundMax/100; sAddRound++){

```

名称	值	类型
sRound	100	shc
sSheep	0x0027f2ac (998, 998, 995, 995, 992, 992, 992, 991, 991, 991, 987, 987, 984, 984, 981, 981, 980, ...)	shc
sSheepMax	0x0027f1dc (998, 998, 995, 995, 992, 992, 992, 991, 991, 991, 987, 987, 984, 984, 981, 981, 980, ...)	shc
[0]	0x0027f1dc (998, 998, 995, 995, 992, 992, 992, 991, 991, 991)	shc
[1]	0x0027f1f0 (987, 987, 984, 984, 981, 981, 981, 980, 980, 980)	shc
[2]	0x0027f204 (976, 976, 973, 973, 970, 970, 969, 969, 969)	shc
[3]	0x0027f218 (966, 966, 962, 962, 959, 959, 958, 958, 958)	shc
[4]	0x0027f22c (955, 955, 951, 951, 948, 948, 947, 947, 947)	shc
[5]	0x0027f240 (944, 944, 941, 941, 937, 937, 936, 936, 936)	shc
[6]	0x0027f254 (933, 933, 930, 930, 926, 926, 925, 925, 925)	shc
[7]	0x0027f268 (922, 922, 919, 919, 916, 916, 914, 914, 914)	shc
[8]	0x0027f27c (911, 911, 908, 908, 905, 905, 904, 904, 904)	shc
[9]	0x0027f290 (900, 900, 900, 900, 897, 897, 897, 897, 894)	shc

法 2——选择

The screenshot shows a C program in a debugger. The main function is defined as follows:

```

int iAddSheep(short sSheep[1000], short);
int iSortSheepBubble(short sSheep[1000], short, short);
int iSortSheepSelect(short sSheep[1000], short, short);
int main() {
    short sSheep[1000];
    short sSheepMax[10][10];
    short sRound=0;
    iAddSheep(sSheep, 1000);
    iSortSheepSelect(sSheep, 1000, 100);
    /*把最重100只羊放进一个数组*/
    for (; sRound<100; sRound++) *(sSheepMax[0]+sRound)=*(sSheep+sRound);
    return 0;
}

```

The local variable window shows the following variables and their values:

名称	值	类型
sRound	100	short
sSheep	0x00afefcc {998, 997, 997, 997, 997, 994, 994, 990, 990, 990, 989, 987, 986, 986, 986, 983, 983, 979, 979, ...}	short
sSheepMax	0x00afeefc {0x00afeefc {998, 997, 997, 997, 994, 994, 990, 990, 990, 989, 987, 986, 986, 986, 983, 983, 979, 979, ...}, ...}	short
sSheepMax[0]	0x00afeefc {998, 997, 997, 997, 994, 994, 990, 990, 990, 989, 987, 986, 986, 986, 983, 983, 979, 979, ...}	short
sSheepMax[1]	0x00afeefc {987, 986, 986, 986, 986, 983, 983, 979, 979, 979, 978}	short
sSheepMax[2]	0x00afeefc {976, 975, 975, 975, 972, 972, 969, 969, 969, 967}	short
sSheepMax[3]	0x00afeefc {965, 964, 964, 964, 961, 961, 958, 958, 958, 957}	short
sSheepMax[4]	0x00afeefc {954, 953, 953, 953, 950, 950, 947, 947, 947, 946}	short
sSheepMax[5]	0x00afeefc {944, 942, 942, 942, 939, 939, 936, 936, 936, 935}	short
sSheepMax[6]	0x00afeefc {933, 932, 932, 932, 928, 928, 925, 925, 925, 924}	short
sSheepMax[7]	0x00afeefc {922, 921, 921, 921, 917, 917, 914, 914, 914, 913}	short
sSheepMax[8]	0x00afeefc {911, 910, 910, 910, 910, 907, 907, 906, 906, 906}	short
sSheepMax[9]	0x00afeefc {903, 903, 903, 903, 903, 900, 900, 900, 900, 899}	short

第二题

程序代码: (被注释掉的函数是可以进行对象输入的部分, 题目要求初始化定义, 故注释掉)

```

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
/*矩阵及其参量定义*/
double
rMat[128][128]={ {1. 1161, 0. 1254, 0. 1397, 0. 1490}, {0. 1582, 1. 1675, 0. 1768, 0. 1871}, {0.
2368, 0. 2471, 0. 2568, 1. 2671}, {0. 1968, 0. 2071, 1. 2168, 0. 2271}};
short sRow=4, sColumn=4;
/*向量定义*/
double rConst[128]={1. 5471, 1. 6471, 1. 8471, 1. 7471};
double rSolution[128];
/*行处理进展静态变量*/
short sRowsNotNull; /*一次消元过程首列非零的行数*/
short sSolve=1; /*是否有唯一解*/
short sNull=0; /*零空间维数*/
/*定义函数*/
char cInput(void); /*矩阵和向量输入保存*/
char cSwitch(short); /*单列对换变换*/
char cGauss(short); /*单列倍加变换*/
char cPLDU(void); /*求阶梯形矩阵并判断解的情况*/
char cBack(short); /*单次回代*/
char cBackTotal(void); /*完成回代*/

```

```

char cfOutput(void);/*输出结果*/
/*主函数*/
char main() {
    //cfInput();/*输入矩阵和向量*/
    cPLDU();/*化为阶梯型，输出非唯一解情形*/
    if (sSolve==1){/*有解情形*/
        cBackTotal();/*回代法*/
        cfOutput();/*输出唯一解*/
    }
    system("pause");
    return 0;
}
/*矩阵和向量输入保存*/
char cfInput() {
    short sRowRound, sColumnRound; /*循环变量*/
    printf("请输入矩阵的行(列)数: ");
    scanf("%hd", &sRow);
    sColumn=sRow;
    /*矩阵输入*/
    printf("请输入矩阵，同行用\\t隔开，异行用\\n隔开: \\n");
    for (sRowRound=0; sRowRound<sRow; sRowRound++) {
        for (sColumnRound=0; sColumnRound<sColumn; sColumnRound++)
            scanf("%lf", &rMat[sRowRound][sColumnRound]);
    }
    /*向量输入*/
    printf("请输入向量，用空格或制表符隔开: \\n");
    for (sRowRound=0; sRowRound<sRow; sRowRound++)
        scanf("%lf", &rConst[sRowRound]);
    return 0;
}
/*判断行交换的必要性并进行行交换*/
char cSwitch(short sColumnTested) {
    short sRowAbove=0, sRowBelow=0; /*非零区行数和零区行数指示变量*/
    short sSwitchRound, sColumnCopyRound; /*循环变量*/
    sRowsNotNull=0;
    if (sColumnTested>=sColumn) { /*由于先前平移行数大于列数，列数超限*/
        sRowsNotNull=1;
        return 0;
    }
    else { /*分首列是否为0向下平移*/
        for (sSwitchRound=sColumnTested; sSwitchRound<sRow; sSwitchRound++) {
            if (fabs(rMat[sSwitchRound][sColumnTested])<1e-9) { /*首项为零行向下
            平移两倍*/
                /*矩阵操作*/

```

```

        for
(sColumnCopyRound=sColumnTested;sColumnCopyRound<sColumn;sColumnCopyRound++) {

    rMat[sRowBelow+2*sRow][sColumnCopyRound]=rMat[sSwitchRound][sColumnCopyRound];

    }
    /*向量操作*/
    rConst[sRowBelow+2*sRow]=rConst[sSwitchRound];
    sRowBelow++;/*首项零行多一个*/
}
else {/*首项非零行向下平移一倍*/
    /*矩阵操作*/
    for
(sColumnCopyRound=sColumnTested;sColumnCopyRound<sColumn;sColumnCopyRound++) {

        rMat[sRowAbove+sRow][sColumnCopyRound]=rMat[sSwitchRound][sColumnCopyRound];
        ;

        }
        /*向量操作*/
        rConst[sRowAbove+sRow]=rConst[sSwitchRound];
        sRowAbove++;/*首项非零行多一个*/
        sRowsNotNull++;/*需要处理的非零行数增加*/
    }
}
/*回填补齐*/
for (sSwitchRound=0;sSwitchRound<sRowsNotNull;sSwitchRound++) {/*首项非
零行填在上面*/
    /*矩阵操作*/
    for
(sColumnCopyRound=sColumnTested;sColumnCopyRound<sColumn;sColumnCopyRound++) {

        rMat[sColumnTested+sSwitchRound][sColumnCopyRound]=rMat[sSwitchRound+sRow][sColumnCopyRound];
        }
        /*向量操作*/
        rConst[sColumnTested+sSwitchRound]=rConst[sSwitchRound+sRow];
    }
    for (sSwitchRound=0;sSwitchRound+sRowsNotNull<sRow;sSwitchRound++) {/*首
项为零行填在下面*/
        /*矩阵操作*/
        for
(sColumnCopyRound=sColumnTested;sColumnCopyRound<sColumn;sColumnCopyRound++) {

            rMat[sColumnTested+sSwitchRound+sRowsNotNull][sColumnCopyRound]=rMat[sSwitc

```

```

hRound+2*sRow][sColumnCopyRound];
    }
    /*向量操作*/

    rConst[sColumnTested+sSwitchRound+sRowsNotNull]=rConst[sSwitchRound+2*sRow]
;
    }
    return 0;
}
}
/*一步高斯消元*/
char cGauss(short sRowStart) {
    cSwitch(sRowStart);
    if (sRowsNotNull==0) { /*不满秩*/
        short sMoveRow=sRow-1; /*行循环变量*/
        short sMoveColumn; /*列循环变量*/
        for (; sMoveRow>=sRowStart; sMoveRow--) { /*向下平移使主元还在对角线上*/
            /*矩阵操作*/
            for (sMoveColumn=sRowStart+1; sMoveColumn<sColumn; sMoveColumn++) {
                rMat[sMoveRow+1][sMoveColumn]=rMat[sMoveRow][sMoveColumn];
            }
            /*向量操作*/
            rConst[sMoveRow+1]=rConst[sMoveRow];
        }
        sRow++; /*扩大行数*/
        sSolve=0; /*解不唯一*/
        sNull++; /*缺一个主元零空间维数加一*/
        return 0;
    }
    else { /*暂时满秩*/
        double rOne=rMat[sRowStart][sRowStart]; /*归一基准值*/
        short sGaussRowRound, sGaussColumnRound; /*循环变量*/
        /*归一化*/
        /*矩阵操作*/
        for
(sGaussColumnRound=sRowStart; sGaussColumnRound<sColumn; sGaussColumnRound++) {
            rMat[sRowStart][sGaussColumnRound]/=rOne;
        }
        /*向量操作*/
        rConst[sRowStart]/=rOne;
        /*加减消元*/
        for
(sGaussRowRound=sRowStart+1; sGaussRowRound<sRowStart+sRowsNotNull; sGaussRowRound++) {

```

```

        double
rRowConstK=rMat[sGaussRowRound][sRowStart]/rMat[sRowStart][sRowStart];/*倍加比
例系数*/

        /*矩阵操作*/
        for
(sGaussColumnRound=sRowStart;sGaussColumnRound<sColumn;sGaussColumnRound++){
            rMat[sGaussRowRound][sGaussColumnRound]-
=rMat[sRowStart][sGaussColumnRound]*rRowConstK;
        }
        /*向量操作*/
        rConst[sGaussRowRound]-=rConst[sRowStart]*rRowConstK;
    }
}
return 0;
}

/*化为上三角全过程*/
char cPLDU() {
    short sPLDURound=(short)0;/*循环变量*/
    short sFlagNull=0;/*判断增广列是否为主列的指标，0否*/
    for (sPLDURound=0;sPLDURound<sRow;sPLDURound++) cGauss(sPLDURound);/*进行高
斯消元化为阶梯型*/
    if (fabs(rMat[sRow-1][sColumn-1])<1e-9){/*原矩阵不满秩*/
        for (sPLDURound=sRow-1;sPLDURound>=sRow-sNull;sPLDURound--){
            if (fabs(rConst[sPLDURound])>1e-9){/*增广列出现主元*/
                sFlagNull++;
                break;
            }
        }
        if (sFlagNull==0) printf("方程有无穷组解，零空间维数为%d。
\n",sNull);/*增广列非主列*/
        else printf("方程无解。 \n");/*增广列为主列*/
    }
    return 0;
}

/*单次回代法*/
char cBack(short sPosition) {
    short sRoundBack=sRow-1;/*循环变量*/
    double rResultP=rConst[sPosition];/*输出结果变量初始化*/
    for (;sRoundBack>sPosition;sRoundBack--){/*逐列回代*/
        rResultP-=rMat[sPosition][sRoundBack]*rSolution[sRoundBack];
    }
    rSolution[sPosition]=rResultP;/*解向量赋值*/
    return 0;
}

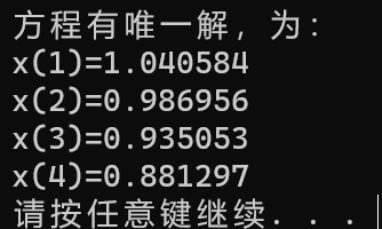
```

```

/*回代法求唯一解汇总*/
char cBackTotal() {
    short sRoundBT=sRow-1; /*循环变量*/
    for (;sRoundBT>=0;sRoundBT--) cBack(sRoundBT); /*逐行回代*/
    return 0;
}
/*输出唯一解*/
char cfOutput() {
    short sRoundOutput=0; /*循环变量*/
    printf("方程有唯一解，为：\n");
    /*逐行输出解向量*/
    for (;sRoundOutput<sRow;sRoundOutput++)
        printf("x(%hd)=%lf\n", sRoundOutput+1, rSolution[sRoundOutput]);
    return 0;
}

```

运行结果截图：



```

方程有唯一解，为：
x(1)=1.040584
x(2)=0.986956
x(3)=0.935053
x(4)=0.881297
请按任意键继续. . . |

```

第三题

思路：穷举遍历找到结果

程序代码：

```

#include <stdio.h>
#include <stdlib.h>
#define TOTAL_HEAD 15
#define TOTAL_FEET 40
int main() {
    int iChicken=0;
    int iRabbit;
    int iFeet;
    for (;iChicken<=15;iChicken++) {
        iRabbit=TOTAL_HEAD-iChicken;
        iFeet=2*iChicken+4*iRabbit;
        if (iFeet==TOTAL_FEET) {
            printf("鸡有%d只，兔有%d只。\\n", iChicken, iRabbit);
            break;
        }
    }
    system("pause");
}

```



```
    return 0;
}
```

运行结果：

```
鸡有10只，兔有5只。
请按任意键继续. . . |
```

选做题

第一题

程序代码：

```
#include <stdio.h>
#include <stdlib.h>
int iCalculate(int);
int iInclude(int);
int iSaver[100000];/*各次结果保存*/
/*主函数*/
int main() {
    int iTest;/*测试数*/
    int iRound=0;/*循环变量*/
    printf("请输入被测整数：");
    scanf("%d",&iTest);
    for (;1;iRound++){
        if (iTest==1){
            printf("1");
            break;
        }
        else if (iInclude(iTest)){
            printf("0");
            break;
        }
        iSaver[iRound]=iTest;
        iTest=iCalculate(iTest);
    }
    system("pause");
    return 0;
}

/*计算各位平方和*/
int iCalculate(int iNum) {
    int iResult=0;/*存储结果*/
    while (iNum>0) {
        iResult+=(iNum%10)*(iNum%10);/*某位平方*/
        iNum/=10;
    }
    return iResult;
}
```

```

}
/*判断数是否在数组中*/
int iInclude(int iNum) {
    int iIncludeRound=0; /*循环变量*/
    int iIncludeFlag=0; /*标志, 1为在里面*/
    for (; iSaver[iIncludeRound]!=0; iIncludeRound++) {
        if (iNum==iSaver[iIncludeRound]) { /*相等*/
            iIncludeFlag=1;
            break;
        }
    }
    return iIncludeFlag;
}

```

运行结果:

```

请输入被测整数: 201004
0请按任意键继续. . . |

```

第二题

答案: 10, 9, 8, 7, 6, 1, 2, 3, 4, 5

第三题

程序代码:

```

#include<stdio.h>
#include<stdlib.h>
short sID[18]; /*身份证号转变*/
short sWeight[17]={7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2}; /*权重*/
int main() {
    int iRound=0; /*循环变量*/
    int iResult=0; /*求解末位结果*/
    printf("请输入身份证号: ");
    for (; iRound<18; iRound++) sID[iRound]=getchar()-48;
    if (sID[17]=='X'-48) sID[17]=10;
    for (iRound=0; iRound<17; iRound++) iResult+=sID[iRound]*sWeight[iRound]; /*计算加权乘法*/
    iResult=(12-iResult%11)%11; /*计算余数*/
    /*输出结果*/
    if (iResult==sID[17]) printf("合法.\n");
    else if (iResult==10) printf("非法, 正确校验位是X.\n");
    else printf("非法, 正确校验位是%d.\n", iResult);
    system("pause");
    return 0;
}

```

运行结果:

请输入身份证号：110109202002172254
非法，正确校验位是7。
请按任意键继续. . . |