第一题

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

struct Account //定义账户结构体

{

int Number; //分别定义编号，ID，姓名，地址，钱数

char ID[20] = { 0 };

char Name[20] = { 0 };

char Address[30] = { 0 };

int Money;

};

int main()

{

Account User; //初始化结构体

printf("请输入编号\n");

scanf("%d", &User.Number);

printf("请输入ID\n");

scanf("%s", &User.ID);

printf("请输入姓名\n");

scanf("%s", &User.Name);

printf("请输入地址\n");

scanf("%s", &User.Address);

printf("请输入金额\n");

scanf("%d", &User.Money);

return 0;

}

第二题

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

struct StudentInfo //定义学生结构体

{

char Name[20]; //分别定义姓名，中文，数学，英语成绩

float Chinese;

float Math;

float English;

float Avg() //获得平均值

{

return (Chinese + Math + English) / 3;

}

};

int main()

{

StudentInfo Student; //初始化

printf("请输入名字，中文，数学，英语成绩\n"); //输入

scanf("%s%f%f%f", &Student.Name, &Student.Math, &Student.Chinese, &Student.English);

printf("%s的平均成绩是%f", Student.Name, Student.Avg()); //输出

return 0;

}

第三题

#define \_CRT\_SECURE\_NO\_WARNINGS

#include<stdio.h>

#include<string.h>

#define N 3

#define ll long long

struct EmployeeInfo //定义员工信息

{

ll Phone; //分别定义手机，姓名，工资

char Name[20] = { 0 };

float Salary;

};

bool CompareEmployee(EmployeeInfo E1, EmployeeInfo E2) //比较工资，A>B为true

{

if (E1.Salary > E2.Salary)

{

return 1;

}

return 0;

}

void Bubblesort(EmployeeInfo Enployee[]) //冒泡排序

{

bool Flag = 1;

EmployeeInfo Temp;

for (int i = 0; i < N - 1 && Flag;i++)

{

Flag = false;

for (int j = 0; j < N - 1 - i;j++)

{

if (CompareEmployee(Enployee[j], Enployee[j + 1])) //利用工资比较函数进行排序

{

Temp = Enployee[j];

Enployee[j] = Enployee[j + 1]; //交换

Enployee[j + 1] = Temp;

Flag = true;

}

}

}

}

int search(EmployeeInfo Employee[], char Name[]) //根据姓名在列表中查找

{

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

{

if (!strcmp(Employee[i].Name, Name))

{

printf("姓名:%s\n手机:%lld\n工资:%f\n", Employee[i].Name,Employee[i].Phone,Employee[i].Salary);

return 0; //找到则输出

}

}

printf("没有找到\n"); //没找到则返回

return 0;

}

int main()

{

EmployeeInfo Employee[N];

printf("请输入 姓名 手机 工资\n");

for (int i = 0; i < N; i++) //输入所有信息

{

scanf("%s %lld %f",&Employee[i].Name, &Employee[i].Phone, &Employee[i].Salary);

}

Bubblesort(Employee); //冒泡排序

printf("排序过后:\n");

for (int i = 0;i < N;i++) //输出排序结果

{

printf("%s %lld %f\n", Employee[i].Name, Employee[i].Phone, Employee[i].Salary);

}

char Name[20];

printf("请输入查找姓名:\n"); //查找指定姓名

scanf("%s", &Name);

search(Employee, Name);

return 0;

}

第四题

#include <stdio.h>

#include <string.h>

#define ll long long

struct Fraction //定义分数结构

{

ll Numerator = 0; //分子

ll Denominator = 0; //分母

void ToPrint() //输出分数

{

printf("%lld/%lld", Numerator, Denominator);

}

};

Fraction operator + (Fraction a, Fraction b) //定义分数加法

{

Fraction Temp = { a.Numerator \* b.Denominator + b.Numerator \* a.Denominator,a.Denominator \* b.Denominator };

// a/b + c/d = (ad+bc)/bd

return Temp; //返回临时

}

Fraction operator \* (Fraction a, Fraction b) //定义乘法

{

Fraction Temp = { a.Numerator \*b.Numerator,a.Denominator \* b.Denominator };

// a/b \* c/d = ac/bd

return Temp; //返回

}

int main()

{

Fraction F1 = { 9,30 }; //定义两个分数

Fraction F2 = { 14,18 };

Fraction F3 = F1 + F2; //相加并输出

F3.ToPrint();

printf("\n");

Fraction F4 = F1 \* F2; //相乘并输出

F4.ToPrint();

return 0;

};

第五题

#include <stdio.h>

#include <string.h>

#define ll long long

struct Fraction //定义分数结构

{

ll Numerator = 0; //分子

ll Denominator = 0; //分母

void ToPrint() //输出分数

{

printf("%lld/%lld", Numerator, Denominator);

}

void Simplize() //化简程序

{

ll Min = (Numerator > Denominator) ? Denominator : Numerator;//得到分子分母较小的Min

int Flag; //下面进行化简，用于标记化简是否完成

do

{

Flag = false; //假设化简完成

for (int i = 2; i < Min / 2 + 1; i++)

{ //循环到Min的一半，判断是否有分子分母公因数

if (Numerator % i == 0 && Denominator % i == 0)

{

Numerator /= i; //有就都除

Denominator /= i;

Flag = true; //标记化简未完成

break; //跳出循环，重新遍历

}

}

} while (Flag); //循环到化简完成

}

};

Fraction operator + (Fraction a, Fraction b) //定义分数加法

{

Fraction Temp = { a.Numerator \* b.Denominator + b.Numerator \* a.Denominator,a.Denominator \* b.Denominator };

// a/b + c/d = (ad+bc)/bd

return Temp; //返回临时

}

Fraction operator \* (Fraction a, Fraction b) //定义乘法

{

Fraction Temp = { a.Numerator \* b.Numerator,a.Denominator \* b.Denominator };

// a/b \* c/d = ac/bd

return Temp; //返回

}

int main()

{

Fraction F1 = { 9,30 }; //定义两个分数

Fraction F2 = { 14,18 };

Fraction F3 = F1 + F2; //相加并输出

F3.ToPrint();

printf("\n");

F3.Simplize();

F3.ToPrint();

printf("\n");

Fraction F4 = F1 \* F2; //相乘并输出

F4.ToPrint();

printf("\n");

F4.Simplize();

F4.ToPrint();

printf("\n");

return 0;

};

第六题

#include <stdio.h>

#include <math.h>

struct Point

{

float X;

float Y;

};

float Distance(Point A, Point B) //普通距离

{

return(sqrt(pow(A.X - B.X, 2) + pow(A.Y - B.Y, 2)));

}

float ManhattanDistance(Point A, Point B) //麦哈顿距离

{

return(fabs(A.X - B.X) + fabs(A.Y - B.Y));

}

int main()

{

Point A = { 1,5 }, B = { 5,9 };

printf("距离:%f 麦哈顿距离:%f", Distance(A, B), ManhattanDistance(A, B));

return 0;

}

第七题

#include <stdio.h>

#include<math.h>

#define N 3

void GetHist(int psrc[], int nlen, int phist[8]) //获取N\*N矩阵

{

for (int i = 0; i < pow(nlen, 2);i++)

{

scanf\_s("%d", psrc + i); //读取

phist[\*(psrc + i)]++; //对应统计增加一次记录

}

}

void Negative(int psrc[], int pdst[], int nlen) //每一位取反

{

for (int i = 0; i < nlen \* nlen; i++)

{

\*(pdst + i) = 7 - \*(psrc + i);

}

}

int main()

{

int psrc[N \* N] = { 0 }; //初始化

int nlen = N;

int phist[8] = { 0 };

GetHist(psrc, nlen, phist); //输入矩阵

int pdst[N \* N] = { 0 };

Negative(psrc, pdst, nlen); //取反

for (int i = 0; i < 8; i++) //输出统计信息

{

printf("%d 出现了 %d 次\n", i, \*(phist + i));

}

for (int i = 0; i < nlen \* nlen; i++) //输出取反矩阵

{

if (i % N == 0)printf("\n");

printf("%d ", \*(pdst + i));

}

return 0;

}

第八题

#include <stdio.h>

#define N 5

void Sort(int\* Scoreary, int nLen) //冒泡排序

{

int Temp;

bool Flag = true;

for (int i = 0; i <= nLen - 2 && Flag; i++)

{

Flag = false;

for (int j = 0; j <= nLen - 2 - i;j++)

{

if (\*(Scoreary + j) > \*(Scoreary + j + 1))

{

Temp = \*(Scoreary + j);

\*(Scoreary + j) = \*(Scoreary + j + 1);

\*(Scoreary + j + 1) = Temp;

Flag = true;

}

}

}

}

void Statictic(int\* Scoreary, int nLen, int Pa[3]) //成绩分段

{

for (int i = 0; i < nLen; i++) //循环每一份成绩

{

if (\*(Scoreary + i) <= 100 && \*(Scoreary + i) >= 85) //在对应区间写入

{

Pa[0]++;

continue;

}

if (\*(Scoreary + i) <= 84 && \*(Scoreary + i) >= 60)

{

Pa[1]++;

continue;

}

if (\*(Scoreary + i) <= 59 && \*(Scoreary + i) >= 0)

{

Pa[2]++;

continue;

}

}

}

int main()

{

int ScoreArr[N]; //初始化成绩表

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

{

scanf\_s("%d", ScoreArr + i); //输入

}

Sort(ScoreArr, N); //排序

int Pa[3] = { 0 };

Statictic(ScoreArr, N, Pa); //分级

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

{

if (i % N == 0)printf("\n"); //输出

printf("%d ", \*(ScoreArr + i));

}

char Text[3][50] = { "100- 85分","84 - 60分" ,"59 - 0分" };

for (int i = 0; i < 3; i++) //按照分数输出

{

if (i % N == 0)printf("\n");

printf("%s \t %d \n", Text[i], \*(Pa + i));

}

}

第九题

#include <stdio.h>

int Isparallel(float\* L1, float\* L2)

{

if (\*L1 \* \*(L2 + 1) == \*L2 \* \*(L1 + 1)) // a1\*b2 == a2\*b1条件

{

return true; //返回结果

}

return false;

}

int main()

{

float L1[3] = { 2,6,10 };

float L2[3] = { 4,12,-5 };

printf("L1 L2 %s 平行\n", Isparallel(L1, L2) ? "是" : "否"); //根据结果输出

float L3[3] = { 1,2,10 };

float L4[3] = { 2,1,-5 };

printf("L3 L4 %s 平行\n", Isparallel(L3, L4) ? "是" : "否");

}