第一题

#include <stdio.h>

#define N 10

int main()

{

float Arr[N]; //记录输入数据

float Sum = 0; //记录总数据

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

{

scanf\_s("%f",&Arr[i]); //输入

Sum += Arr[i];

}

Sum = Sum / (float)N; //获得平均值

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

{

if (Arr[i] > Sum)

{

printf("%.2f ", Arr[i]); //大于平均值输出

}

}

}

第二题

#include <stdio.h>

int Arr[5][5] = { {1 ,2 ,3 ,4 ,5 },

{6 ,7 ,8 ,9 ,10},

{11,12,13,14,15},

{16,17,18,19,20},

{21,22,23,24,25} };

int main()

{

int Count = 0;

for (int i = 0; i < 5; i++)

{

for (int j = 0; j < 5; j++)

{ //双循环每一个数据

if (i == 0 || j == 0 || i == 4 || j == 4 || i == j || i + j == 4)

{ //如果边框（i/j 一个为 0/4） ，对角线（i=j / i + j =4）

Count += Arr[i][j]; //统计加入

}

}

}

printf("%d", Count);

}

第三题

#include <stdio.h>

void MatrixMultiplicate(int M1[3][4], int M2[4][3], int M3[3][3])

{

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{ //双循环，计算结果矩阵M3[i][j]

for (int k = 0; k < 4; k++)

{

M3[i][j] += M1[i][k] \* M2[k][j]; //根据M3[i][j] = Sum(M1[i][k]\*M2[k][j])计算

}

}

}

}

int main()

{

int Matrix1[3][4] = { 2, 5, 9, 8, //初始化两个矩阵

11, 86, 22, 11,

65, 59, 88, 44 };

int Matrix2[4][3] = { 4, 6, 22,

33, 51, 21,

22, 3, 4,

5, 7, 1 };

int Matrix3[3][3] = { 0 }; //用来输出结果的矩阵

MatrixMultiplicate(Matrix1, Matrix2, Matrix3); //计算

for (int i = 0; i < 3; i++)

{

for (int j = 0; j < 3; j++)

{

printf("%d\t", Matrix3[i][j]); //输出结果矩阵

}

printf("\n");

}

return 0;

}

第四题

#include <stdio.h>

#define N 200

int main()

{

bool Arr[N + 1]; //判断index是不是一个素数

for (int i = 1; i <= N; i++)

{

Arr[i] = true; //假设所有的都是素数

}

for (int i = 2;i <= N;i++) //从二开始循环

{

if (Arr[i] == true) //如果那个数是素数（2肯定是)

{

for (int j = 2 \* i; j <= N; j += i) //遍历它后面的数只要是倍数，都不是素数

{

if (j % i == 0)

{

Arr[j] = false; //不是素数进行标记

}

}

}

}

for (int i = 1; i <= N; i++) //输出

{

if (Arr[i] == true)

{

printf("%d ", i);

}

}

return 0;

}

第五题

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

#include <string.h>

int main()

{

char Str1[666] = { 0 }, Str2[666] = { 0 }; //注意，scanf只会补充一个\0.为了避免i=2n跳过溢出，需要全为0

printf("请输入一个字符串\n"); //输入字符串

scanf("%s", &Str1);

int i = 0, j = 0;

for (; Str1[i] != '\0'; i += 2, j += 1) //i一次增两个，j增一个，对应str1，str2赋值

{

Str2[j] = Str1[i];

}

printf("%s", Str2);

return 0;

}

第六题

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

void GetString(char Arr[], int Start, int Long, char Result[]) //将Start开始Arr截取Long长度到Result

{

for (int i = 0; i < Long; i++) //从Start-1开始读取long个长度

{

Result[i] = Arr[i + Start - 1]; //用i标识并写入

}

}

int main()

{

char Arr[200], Result[200] = {0}; //初始化字符串

int Start, Long;

printf("请输入字符串，开始，长度：\n"); //获取字符串，开始位置、长度

scanf("%s", &Arr);

scanf("%d", &Start);

scanf("%d", &Long);

GetString(Arr, Start, Long, Result); //计算

printf("%s",Result); //输出

return 0;

}

第七题

#include <stdio.h>

#define MaxN 100

void PascalTriangle\_1DArr(int N) //一维数组生成三角

{

long Arr[MaxN] = {0, 1}; //初始化第一行的1

for (int i = 1; i <= N;i++) //循环到N行

{

for (int j = i; j > 0 ;j--) //逆向刷新

{

Arr[j] = Arr[j] + Arr[j - 1]; //每一个都是Arr[j] = Arr[j] + Arr[j - 1]

}

for (int j = 1; j <= i;j++) //正向

{

printf("%ld\t", Arr[j]); //输出

}

printf("\n");

}

}

void PascalTriangle\_2DArr(int N) //二维数组生成三角

{

long Arr[MaxN][MaxN] = {0, 1}; //准备好第一行的N

for (int i = 1; i <= N;i++)

{

for (int j = i; j > 0;j--)

{

Arr[i][j] = Arr[i-1][j] + Arr[i-1][j - 1]; //等于上面的两项和

printf("%ld\t", Arr[i][j]); //输出

}

printf("\n");

}

}

int main()

{

PascalTriangle\_1DArr(10); //分别输出一维和二维的

PascalTriangle\_2DArr(10);

return 0;

}

第八题

#include <stdio.h>

#include <math.h>

int Int2String(int Num, char Str[]) //数字转文本

{

if (Num == 0)

{

return 0; //Num=0，说明超过最大位 返回0

}

int N = Int2String(Num / 10, Str); //递归直到最左端位

Str[N] = (Num % 10) + '0'; //写入

return N + 1; //写入下一个数

}

bool IsPalindrome(char Str[]) //如果一段文字是回文

{

int Len = 0;

for (;Str[Len] != 0;Len++); //首先获得Len长度

for (int i = 0; i < Len / 2 ; i++)

{

if (Str[i] != Str[Len - 1 - i]) //和对称位判断

{

return false; //只要一个不对称就false

}

}

return true; //全对称，true

}

int main()

{

for (int i = 1; i <= 216; i++)

{

char Str[200] = {0};

Int2String(pow(i, 2), Str); //先转文字

if (IsPalindrome(Str)) //判断文字对称

{

printf("%d 的平方回文数\n", i);

}

}

}

第九题

#define \_CRT\_SECURE\_NO\_WARNINGS

#include <stdio.h>

int main()

{

char PassWord[20];

scanf("%s", &PassWord); //输入

int PerNum[8];

for (int i = 0;i <= 7;i++) //每一位得到数字，+5 %10

{

PerNum[i] = PassWord[i] - '0';

PerNum[i] = (PerNum[i] + 5) % 10;

}

int Mov3PerNum[8]; //移动三位

for (int i = 5, j = 0; j <= 7;i++, j++)

{

Mov3PerNum[j] = PerNum[i % 8];

}

for (int i = 7; i >= 0; i--) //逆向输出

{

printf("%d", Mov3PerNum[i]);

}

return 0;

}

第十题

#include <stdio.h>

#define N 10

#define M 3

int Cycle(bool GoAway[], int Pos, int Count, int Left) //递归方式解决约瑟夫环

{

Pos = Pos % N; //避免数组越界，实现下标循环

if (Left == 0) //没有人了就返回

{

return 0;

}

if (GoAway[Pos])

{

Cycle(GoAway, Pos + 1, Count, Left); //如果Pos这个位置已经走了，去找Pos+1

}

else

{

Count++; //Pos没走，那么计数

if (Count == M)

{

GoAway[Pos] = true; //如果计数等于M，标记走了

printf("%d 走了\n", Pos + 1); //因为Pos是下标，对应人要+1

Cycle(GoAway, Pos + 1, 0, Left - 1); //从下一个开始从新计数，并记录少一个人

}

else

{

Cycle(GoAway, Pos + 1, Count, Left); //如果计数累加不到M，继续记载下一个Pos

}

}

}

int FindTheWinner(int n, int k) //直接利用公式法去寻找最后走的人

{

int p = 0; //f(N,M)=(f(N−1,M)+M)%N

for (int i = 2; i <= n; i++) {

p = (p + k) % i;

}

return p + 1; //对应人是下标+1

}

int main()

{

bool Goaway[N] = { 0 }; //初始化约瑟夫环

Cycle(Goaway, 0, 0, N); //递归方式找顺序

printf("Winner : %d\n", FindTheWinner(N, M)); //直接公式得到结果

return 0;

}