程序设计基础

实验与习题参考答案

（最终版）

2015 年 12 月 16 日

注：未经全面测试，仅供参考，发现错误，请及时修正！

目录

第二章

2.2 用赋值表达式表示下列计算。

（1） $y = x^{a+b^c}$

（2） $x = (\ln\sqrt{a+d^2} - e^{26})^{5/2}$

（3） $y = \frac{\sin X}{aX} + \left|\cos\frac{\pi X}{2}\right|$

（4） $R = \frac{1}{\frac{1}{R_1}+\frac{1}{R_2}+\frac{1}{R_3}}$

（5） $y = \cfrac{x}{1+\cfrac{x}{3+\cfrac{(2x)^2}{5+\cfrac{(2x)^3}{7+(4x)^2}}}}$

（1）y=pow（x,（a+pow（b，c））
（2）x=pow（log（sqrt（a+d*d））-exp（26）,5.0/2.0）
（3）y=sin（X）/（a*X）+fabs（cos（3.14159265*X/2.0））
（4）R=1.0/（1.0/R1+1.0/R2+1.0/R3）
（5）y=x/（1+x/（3+（2*x*2*x）/（5+pow（2*x,3）/（7+4*x*4*x））））

2.3 编程序，输入一个字符，然后顺序输出该字符的前驱字符、该字符本身、它的后继字符。

```
#include "stdio.h"
void main()
{
    char cIn;
    scanf("%c",&cIn);
    if(cIn>33&&cIn<126)
    {
        printf("%c,%c,%c\n",cIn-1,cIn,cIn+1);
    }
    else
    {
        printf("三个字符不都是可显示字符\n");
    }
}
```

2.6 编写程序，输入两个整数，分别求它们的和、差、积、商、余数并输出

```
#include "stdio.h"
void main()
{
    int a,b;
    int He,Cha,Ji;
    int Yu;
    float Shang;
    printf("\nInput Two Int Type Number:\n");
    scanf("%d%d",&a,&b);
    He=a+b;
    Cha=a-b;
    Ji=a*b;
    Yu=a%b;
    Shang=float(a)/float(b);
    printf("Result Is:%d,%d,%d,%f,%d\n",He,Cha,Ji,Shang,Yu);
}
```

2.9 已知摄氏温度（℃）与华氏温度（℉）的转换关系是：

$$C=\frac{5}{9}(F-32)$$

编写一个摄氏温度（℃）与华氏温度（℉）进行转换的程序，输入摄氏温度，输出华氏温度。

```
#include"stdio.h"
void main()
{
    float C,F;
    printf("请输入摄氏温度：\n");
    scanf("%f",&C);
    F=9.0*C/5+32;
    printf("F=%f\n",F);
}
```

第三章

3.1 编写程序，输入一个字母，若其为小写，将其转换成相应的大写字母，然后输出。

```
#include"stdio.h"
void main()
{
    char c1,c2;
    printf("请输入一个字母：\n");
    c1=getchar();
    if((c1>='a')&&(c1<='z'))
    {
        c2=c1-32;
        printf("%c\n",c2);
    }
    else if((c1>='A')&&(c1<='Z'))
        printf("%c\n",c1);
    else
        printf("输入的不是字母！\n");
}
```

3.9 编写程序，判断给定的 3 位数是否为 Armstrong 数，Armstrong 数是指其值等于它本身每位数字立方和的数，如 153 就是一个 Armstrong 数。

$$153=1^3+5^3+3^3$$

```
#include<stdio.h>
void main(void)
{
    int InData;
    int a,b,c;

    printf("Input a Number:");
    scanf("%d",&InData);

    a=InData/100;
    b=(InData-100*a)/10;
    c=InData%10;
    if(InData==a*a*a+b*b*b+c*c*c)
        printf("%d 是 Armstrong 数\n",InData);
    else
        printf("%d 不是 Armstrong 数\n",InData);
}
```

3.10 编写程序，读入一个点的坐标 X，Y，计算

$$Z=\begin{cases}\ln X+\ln Y & \text{当}x\text{、y在第一象限}\\ \sin X+\cos Y & \text{当}x\text{、y在第二象限}\\ e^{2x}+e^{3y} & \text{当}x\text{、y在第三象限}\\ \tan(x+y) & \text{当}x\text{、y在第四象限}\end{cases}$$

```
#include"stdio.h"
#include"math.h"
void main()
{
    float x,y,z;
    printf("输入两个数");
    scanf("%f%f",&x,&y);
    if(x>0&&y>0)
        printf("%f\n",log(x)+log(y));
    if(x<0&&y>0)
        printf("%f\n",sin(x)+sin(y));
    if(x<0&&y<0)
        printf("%f\n",exp(2*x)+exp(3*y));
    if(x>0&&y<0)
        printf("%f\n",tan(x+y));
}
```

3.11 编程序，输入一个整数，判断它能否被 3 、5 、7 整除，并输出如下信息。

（1）能同时被 3 、5 、7 整除；
（2）能同时被两个数整除，并指明是被哪两个数整除；
（3）能被一个数整除，并指明是哪个数；
（4）不能被所有 3 个数整除。

```
#include "stdio.h"
void main ()
{
   int n;
   int a,b,c;
   printf("please input n:");
   scanf("%d",&n);
   a=b=c=0;
   if (n%3==0) a=1;
   if (n%5==0) b=1;
   if (n%7==0) c=1;
   switch (a+b+c)
{
case 3:    printf("3,5,7");
                    break;
```

```
        case 2:
                if (a==0) printf("5,7");
                if (b==0) printf("3,7");
                if (c==0) printf("3,5");
                break;
        case 1:
                if (a==1) printf("only 3");
                if (b==1) printf("only 5");
                if (c==1) printf("only 7");
                    break;
        default:printf("never can be!");
   }
}
```

3.20　编程序，当输入数值月份时，显示相应英文月份名称。例如当输入 1 时输出 January ，当输入 5 时输出 May ， 等等。

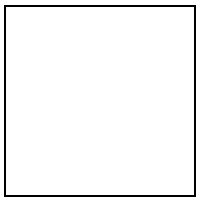
参考答案：

```
#include "stdio.h"
void main(){
    int x;
    printf("Please input the number:");
    scanf("%d",&x);
    switch(x){
        case 1:  printf("JAN\n");
                 break;
        case 2:  printf("FEB\n");
                 break;
        case 3:  printf("MAR\n");
                 break;
        case 4:  printf("APR\n");
                 break;
        case 5:  printf("MAY\n");
                 break;
        case 6:  printf("JUN\n");
                 break;
        case 7:  printf("JUL\n");
                 break;
        case 8:  printf("AUG\n");
                 break;
        case 9:  printf("SEP\n");
                 break;
        case 10: printf("OCT\n");
                 break;
        case 11: printf("NOV\n");
```

```
                break;
        case 12:  printf("DEC\n");
                break;
        default: printf("Wrong number!\n");
    }
}
```

第四章

4.3 利用展开式



计算 ex，到第 100 项；到余项小于 10-8（考虑当 0<x<1；x<0；x>1 时各应该如何计算)。

```
#include<stdio.h>
void main(void){
    int i;
    float p=1.0, sum=1.0, x;
    printf("输入 x 的值\n");
    scanf("%f", &x);
    for(i=1;i<=100;i++){
        p=p*x/i;
        sum+=p;
    }
    printf("到第 100 项的结果为:%f\n",sum);
    }
```

到余项小于 10-8

```
#include<stdio.h>
#define esp 1e-8
void main(void){
    int i;
    float p=1.0, sum=1.0, x;
    printf("输入 x 的值\n");
    scanf("%f", &x);
    for(i=1;fabs(p)>=esp;i++){
        p=p*x/i;
        sum+=p;
    }
    printf("到余项小于 10-8 的结果为:%f\n",sum);
    }
```

4.13 编写程序，打印所有小于 100 的可以被 11 整除的自然数。

```
#include"stdio.h"
void main()
{
    int i=0,n=100;
    for(i=1;i<100;i++)
```

```
        if(i%11==0)
            printf("%d\n",i);
}
```

4.16 编写程序，打印所有 3 位的 Armstrong 数。Armstrong 数是指其值等于它本身每位数字立方和的数，如 153 就是一个 Armstrong 数。

$$15^3=1^3+5^3+3^3$$

```
#include<stdio.h>
void main(void)
{
    int a,b,c,k=100;
    while(k<999)
    {
        a=k/100;
        b=(k-100*a)/10;
        c=k%10;
        if(k==a*a*a+b*b*b+c*c*c)
            printf("结果是:%d\n",k);
        k++;
    }
}
```

4.19 编程序，打印下图形式的数字金字塔。

```
                1
              1 2 1
            1 2 3 2 1
          1 2 3 4 3 2 1
        1 2 3 4 5 4 3 2 1


          … …   … …
        … … … … … … …
    1 2 3 4 5 6 7 8 9 0 9 8 7 6 5 4 3 2 1
```

参考答案：

```
#include "stdio.h"
void main()
{
    int i,j,k,l;
    for (i=1;i<=10;i++)
    {
        for(j=1;j<=10-i;j++)
            printf("    ");
```

```
        for(k=1;k<=i;k++)
        {
            if ( k==10)                                    // 或者是 printf("%2d",k%10)
                printf("%2d",k-10);
            else
                printf("%2d",k);
        }
        for (l=i-1;l>0;l--)
            printf("%2d",l);
        printf("\n");
    }
}
```

4.20 斐波纳契序列问题。

- **有一对小兔子，出生一个月后变成大兔子开始怀孕；**
- **两个月后，生出一对小兔子，这时共有两对兔子(一对大兔子，一对小兔子），同时大兔子又再次怀孕；**
- **三个月后，以前出生的小兔子变成大兔子，以前怀孕的大兔子又生出一对小兔子，这时共有三对兔子(两对大兔子，一对小兔子），所有大兔子又全部怀孕；**
- **四个月后，以前出生的小兔子变成大兔子，以前怀孕的大兔子又各生出一对小兔子，这时共有五对兔子(三对大兔子，两对小兔子），所有大兔子又全部怀孕；**
- **五个月后，以前出生的小兔子变成大兔子，以前怀孕的大兔子又各生出一对小兔子，这时共有八对兔子(五对大兔子，三对小兔子），所有大兔子又全部怀孕；**
    - **…… ……**
- **假设在兔子的生养过程中没有死亡。编程序，输入 n，计算 n 个月后，有多少对兔子，并输出。**

参考答案：

```
#include "stdio.h"
void main()
{
    int x,y,z;
    int i;
    int n;
    printf("please input n : ");
    scanf("%d",&n);
    x=0; //大兔子个数
    y=1; //小兔子个数
    for (i=1;i<=n;i++)
    {
        z=x;
        x=x+y;
        y=z;
    }
    printf("the totle number is %d\n",x+y);   //大兔子加小兔子个数
}
```

4.23 求解非线性方程 f（x）=0 的牛顿迭代法的迭代公式是：

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

编写程序，用此方法求方程 f（x）=x^{41}+x^3+1=0 在 x_0=-1 附近的根。

```
#include"stdio.h"
#include"math.h"
#define eps 1e-6
double f(float x)
{
    return pow(x,41)+pow(x,3)+1;
}
double ff(float x)
{
    return 41*pow(x,40)+3*x*x;
}
void main()
{
    double x0,x1;
    x0=0.0;
    x1=-1.0;
    do
    {
        x0=x1;
        x1=x0-f(x0)/ff(x0);
    }
    while(fabs(x0-x1)>eps);
    printf("x=%f\n",x0);
}
```

4.29 棋盘麦粒

```
#include"stdio.h"
#include"math.h"
void main()
{
    double v,i,n,r,w;
    i=1;
    n=64;
    r=1;
    w=0;
    for(i=1;i<=n;i++)
    {
        w=w+r;
        r=2*r;
    }
```

```
    v=w/1.4e8;
    printf("%e\n",v);
}
```

4.6 编程序，打印斐波纳契序列前 20 项。斐波纳契序列是第 1、2 两项为 1，以后每项为前两项之和，如下：

1、1、2、3、5、8、13、22、… …

```
#include "stdio.h"
int main(int argc, char* argv[])
{
    int i;
    int Fi=1,Se=1;
    int Cu;
    printf("%8d%8d",Fi,Se);
    for(i=3;i<=20;i++)
    {
        Cu=Fi+Se;
        printf("%8d",Cu);
        if(i%5==0)
        {
            printf("\n");
        }
        Fi=Se;
        Se=Cu;
    }
    return 0;
}
```

4.20 编程序，打印图 4.52 字符图形的左侧三分之一部分。

```
A B C D E F G H I          A          B C D E F G H I A
B C D E F G H I A        B B B        C D E F G H I A B
C D E F G I H A B      C C C C C      D E F G H I A B C
D E F G H I A B C    D D D D D D D    E F G H I A B C D
E F G H I A B C D  E E E E E E E E E  F G H I A B C D E
F G H I A B C D E F F F F F F F F F F G H I A B C D E F
E F G I H A B C D  E E E E E E E E E  F G H I A B C D E
D E F G I H A B C    D D D D D D D    E F G H I A B C D
C D E F G I H A B      C C C C C      D E F G H I A B C
B C D E F G H I A        B B B        C D E F G H I A B
A B C D E F G H I          A          B C D E F G H I A
```

图 4.52　字符图形

```
#include "stdio.h"
int main(int argc, char* argv[])
{
	char i;
	int j;
	char PrintChar;

	for(i='A';i<='F';i++)
	{
		for(j=0;j<9;j++)
		{
			PrintChar=((i+j)-'A')%9+'A';
			printf("%c  ",PrintChar);
		}
		printf("\n");
	}
	for(i='E';i>='A';i--)
	{
		for(j=0;j<9;j++)
		{
			PrintChar=((i+j)-'A')%9+'A';
			printf("%c  ",PrintChar);
		}
		printf("\n");
	}
	return 0;
}
```

4.29 古印度有一位十分好玩的国王要奖励为他发明国际象棋的宰相达依尔，问他要什么。达依尔回答:“陛下，只要在国际象棋棋盘的第一个格子中放一粒麦子，第二个格子中放两粒麦子，第三个格子中放四粒麦子，第四个格子中放八粒麦子，… … 如此下去，以后每个格子中都放它前一个格子中麦子数的两倍，这样放满棋盘的 64 个格子为止即可。”。国王觉得这很容易，于是开始奖赏，没想到一袋麦子很快就用完了，下一袋也很快就用完了，最后一算全印度的麦子全部用上也不够。请编程序计算所需麦子的体积（一立方米麦子约 1.4×10^8 粒）。

```
#include"stdio.h"
#include"math.h"
void main()
{
	double v,i,n,r,w;
	i=1;
	n=64;
```

```
    r=1;
    w=0;
    for(i=1;i<=n;i++)
    {
        w=w+r;
        r=2*r;
    }
    v=w/1.4e8;
    printf("%e\n",v);
}
```

P22_45.（习题集）判断给定字符序列中（与）、[与]、{与}是否配对（个数相等）。

```
#include "stdio.h"
int main(int argc, char* argv[])
{
    char InChar;
    InChar='A';
    int i,j,k;
    i=0;
    j=0;
    k=0;
    do
    {
        scanf("%c",&InChar);
        if(InChar=='(')
            i++;
        if(InChar==')')
            i--;
        if(InChar=='[')
            j++;
        if(InChar==']')
            j--;
        if(InChar=='{')
            k++;
        if(InChar=='}')
            k--;
    }while(InChar!='#');
    if(i==0)
        printf("()配对\n");
    else
        printf("()不配对\n");
    if(j==0)
        printf("[]配对\n");
    else
```

```
            printf("[]不配对\n");
        if(k==0)
            printf("{}配对\n");
        else
            printf("{}不配对\n");
        return 0;
}
```

P23_50（1）（习题集）. 读入 n 和 x，输出序列的前 n 项和直到余项小于 10^{-8}

$Sh(x)=x+x^3/3!+x^5/5!+x^7/7!+...+x^{2n+1}/(2n+1)!$

```
#include "stdio.h"
#include "math.h"
int main(int argc, char* argv[])
{
    int n;
    float x;
    int i;
    printf("输入 n:\n");
    scanf("%d",&n);
    printf("输入 x:\n");
    scanf("%f",&x);
    float Res=x;
    float Temp=x;;
    for(i=1;i<=n;i++)
    {
        Temp=Temp*x*x/((2*i)*(2*i+1));
        if(fabs(Temp)<1e-8)
        {
            break;
        }
        Res=Res+Temp;
    }
    printf("计算结果为:%f\n",Res);
    return 0;
}
```

P23_50（6）（习题集）. 读入 n 和 x，输出序列的前 n 项和直到余项小于 10^{-8}

$arth(x)=x+x^3/3+x^5/5+x^7/7+...+x^{2n+1}/(2n+1)$

```
#include "stdio.h"
#include "math.h"
int main(int argc, char* argv[])
{
    int n;
    float x;
```

```
    int i;
    printf("输入 n:\n");
    scanf("%d",&n);
    printf("输入 x:\n");
    scanf("%f",&x);
    float Res=x;
    float Temp1=x;
    float Temp=x;;
    for(i=1;i<=n;i++)
    {
        Temp1=Temp1*x*x;
        Temp=Temp1/(2*i+1);
        if(fabs(Temp)<1e-8)
        {
            break;
        }
        Res=Res+Temp;
    }
    printf("计算结果为:%f\n",Res);
    return 0;
}
```

P24_51（2）（习题集）. 用展开式计算圆周率 PI 到 10^{-5} 精度，

PI/2=(2/1)X(2/3)X(4/3)X(4/5)X(6/5)X(6/7)X(8/7)....(2n/2n-1)X(2n/2n+1).....

```
#include "stdio.h"
#include "math.h"
int main(int argc, char* argv[])
{
    int i;
    float TempP=0;
    float TempC=1;
    i=1;
    while(fabs(TempC-TempP)>=1e-5)
    {
        TempP=TempC;
        TempC=TempP*(2*i)*(2*i)/(2*i-1)/(2*i+1);
        i++;
    }
    printf("计算结果为:%.6f\n",TempC);
    return 0;
}
```

第五章

5.1 编程序计算

$$y(x)=\frac{p^2(x)+5x}{p(x+5)-\sqrt{x}}\cdot p(x+2)$$

其中

$$p(u)=\frac{f(u\times 0.3,u)+u/2}{2u}$$

$$f(v,w)=\frac{w+v}{7v}$$

```
#include "stdio.h"
#include "math.h"
float f(float v,float w)
{
    return (w+v)/(7*v);
}
float p(float u)
{
    return (f(0.3*u,u)+u/2)/(2*u);
}
float y(float x)
{
    return (p(x)*p(x)+5*x)*p(x+2)/(p(x+5)-sqrt(x));
}

void main()
{
    float x;
    float Result;
    printf("\nInput x:\n");
    scanf("%f",&x);
    Result=y(x);
    printf("\nThe result is: %f\n",Result);
}
```

5.2 编写程序，输入实数 a、b、c 的值，计算并输出以下算式的值。

$$T=\frac{4.25(a+b)+\ln(a+b+\sqrt{a+b}+\frac{1}{a+b})}{4.25+\ln(c+\sqrt{c}+\frac{1}{c})}$$

```
#include "math.h"
#include "stdio.h"
float T_ABC(float a,float b,float c)
{
    float Result;
    Result=(4.25*(a+b)+log(a+b+sqrt(a+b)+1.0/(a+b)))/(4.25+log(c+sqrt(c)+1.0/c));
    return Result;
}
void main()
{
    float x,y,z;
    float T;
    printf("Input a,b,c\n");
    scanf("%f%f%f",&x,&y,&z);
    T=T_ABC(x,y,z);
    printf("Result:%f\n",T);
}
```

P24_54（习题集）. 编写程序，当 1.0、2.0、3.0、。。。20.0 时，计算如下函数到 10 层嵌套。

F(x)=1+1/(1+1/(1+1/(1+....+1/(1+1/x)...)))

```
#include "stdio.h"
int main(int argc, char* argv[])
{
    float TempResult;

    for(float x=1.0;x<=20.0;x++)
    {
        TempResult=x;
        for(int i=0;i<10;i++)
        {
            TempResult=1+1/TempResult;
            printf("第%2d 步：结果：%10.6f\n",i,TempResult);
        }
        printf("x=%10.6f,Result:%10.6f\n\n\n",x,TempResult);
    }
    return 0;
}
```

5.3 编程序，输入 m、n 的值，计算并输出：

$$\frac{m!}{(m-n)!\ n!}$$

```
#include "stdio.h"
```

```
int fac(int n){
    int p=1;
    for(;n>1;n--)
        p=p*n;
    return p;
}
main(){
    int m,n,t;
    float f;
    printf("input m ,n\n");
    scanf("%d%d",&m,&n);
    if(m<n){
        t=m;
        m=n;
        n=t;
    }
    printf("f=%5.2f\n",(float)fac(m)/(fac(m-n)*fac(n)));
}
```

5.4 分别编写函数，计算复数的加法、减法、乘法、除法，复数 z 表示成 z=a+ib.

```
#include<stdio.h>
#include<stdlib.h>
double RealP,ImageP;
void add(double a,double b,double c,double d)/* 求和 */
{
    RealP=a+c;
    ImageP=b+d;
}
void sub(double a,double b,double c,double d)/* 求差 */
{
    RealP=a-c;
    ImageP=b-d;
}
void product(double a,double b,double c,double d)/* 求乘积 */
{
    RealP=a*c-b*d;
    ImageP=b*c+a*d;
}
void divi(double a,double b,double c,double d)/* 除法*/
{
    double Temp1;
    double Temp2;
    double Temp3;
    Temp1=a*c+b*d;
```

```
    Temp2=b*c-a*d;
    Temp3=c*c+d*d;
    RealP=Temp1/Temp3;
    ImageP=Temp2/Temp3;
}
int main()
{
    double a,b,c,d;
  //  int i,n;
    int choose;
    puts("This program is about the calculation of two complexes.");/* 菜单 */
    puts("1:Do + of two complexes");
    puts("2:Do - of two complexes");
    puts("3:Do * of two complexes");
    puts("4:Do / of two complexes");
    puts("5:Exit ");
    while(1) /* 循环计算 */
    {
        while(1)/* 输入格式检测，并输入第一个复数 */
        {
            puts("Please input the real part and virtual part of first digit:");
            if(scanf("%lf%lf",&a,&b)==2)
                break;
            fflush(stdin);
        }
        while(1)/* 输入格式检测，并输入第二个复数 */
        {
            puts("Please input the real part and virtual part of second digit:");
            if(scanf("%lf%lf",&c,&d)==2)
                break;
            fflush(stdin);
        }
        while(1) /* 菜单选择 */
        {
            puts("Input your choose:(1-5)");
            if(scanf("%d",&choose)==1)
                break;
            fflush(stdin);
        }
        switch(choose)
        {
        case 1:
            add(a,b,c,d);
            puts("The sum of two digits is:"); /* 显示相加的结果 */
```

```
            printf(ImageP>=0?"%lf+%lfi\n":"%lf%lfi\n",RealP,ImageP);
            break;
        case 2:
            sub(a,b,c,d);
            puts("The sub of two digits is:"); /*  显示相减的结果  */
            printf(ImageP>=0?"%lf+%lfi\n":"%lf%lfi\n",RealP,ImageP);
            break;
        case 3:
            product(a,b,c,d);
            puts("The product of two digits is:"); /*  显示相乘的结果  */
            printf(ImageP>=0?"%lf+%lfi\n":"%lf%lfi\n",RealP,ImageP);
            break;
        case 4:
            divi(a,b,c,d);
            puts("The division of two digits is:"); /*  显示相乘的结果  */
            printf(ImageP>=0?"%lf+%lfi\n":"%lf%lfi\n",RealP,ImageP);
            break;
        default: return 0;
        }
    }
//  system("pause");
    return 0;
}
```

5.6 分别编写函数，检测一个字符是否空格、数字、元音字母

```
#include "stdio.h"
#include "math.h"
void check(char ch)
{
    if(ch==' ')
        printf("%c is a blank",ch);
    else if(ch>='0'&&ch<='9')
        printf("%c is a number",ch);
    else if(ch=='a'||ch=='A'||ch=='o'||ch=='O'||ch=='e'||ch=='E'||ch=='i'||ch=='I'||ch=='u'||ch=='U')
        printf("%c is a yuanyin",ch);
    else
        printf("%c is a other char",ch);
}
main(){
    char ch;
    printf("input a char\n");
    scanf("%ch",&ch);
    check(ch);
}
```

5.7 编写一个函数 f（n），求任意 4 位正整数 n 的逆序数。例如，当 n=2345 时，函数值为

5432。

```
#include "stdafx.h"
#include "stdio.h"
int f(int n)
{
	int QianW,BaiW,ShiW,GeW;
	int Result;
	int Temp;
	Temp=n;
	GeW=Temp%10;
	Temp=Temp/10;
	ShiW=Temp%10;
	Temp=Temp/10;
	BaiW=Temp%10;
	QianW=Temp/10;
	Result=1000*GeW+100*ShiW+10*BaiW+QianW;
	return Result;
}
void main()
{
	int InData;
	printf("Input data:\n");
	scanf("%d",&InData);
	printf("The Result is: %d\n",f(InData));
}
```

第六章

6.1 编写函数，把给定的整数数组中的 0 元素全部移到后面，且所有非 0 元素的顺序不变。

```
#include<stdio.h>
#define N 10

void MoveZero(int InArray[],int EleMents)
{
    int i,j,k;
    int Temp;
    for(i=EleMents-2;i>=0;i--)
    {
        if(InArray[i]==0)
        {
            for(j=i;j<EleMents-1;j++)
            {
                Temp=InArray[j];
                InArray[j]=InArray[j+1];
                InArray[j+1]=Temp;
            }
        }
    }
    printf("移动后结果：\n");
    for(k=0;k<EleMents;k++)
    {
        printf("%-5d",InArray[k]);
    }
    printf("\n");
}

void main(void)
{
    int m;
    int AnyArray[N];
    printf("输入%d 个整数：\n",N);
    for(m=0;m<N;m++)
    {
        scanf("%d",&AnyArray[m]);
    }
    MoveZero(AnyArray,N);
}
```

6.6 编函数，把整数组中值相同的元素删除得只剩一个；并把剩余元素全部串到前边。

```
#include "stdio.h"
void DeleteSameEls(int Arr[],int ElNum)
{
	int i;
	int CurElNum;

	int j;
	int k;

	CurElNum=ElNum;
	for(i=0;i<CurElNum;i++)
	{
		for(j=i+1;j<CurElNum;j++)
		{
			if(Arr[j]==Arr[i])
			{
				for(k=j;k<(CurElNum-1);k++)
				{
					Arr[k]=Arr[k+1];
				}
				Arr[CurElNum-1]=-9999;
				CurElNum--;
				printf("i=%d,j=%d:",i,j);
				for(int m=0;m<ElNum;m++)
				{
					printf("%2d",Arr[m]);
				}
				printf("\n");
				j--;
			}
		}
	}
}

int main(int argc, char* argv[])
{
	int TestArr[1000];
	int RealArrElNum;
	printf("输入数组元素数（小于 1000）：");
	scanf("%d",&RealArrElNum);
	printf("输入数组元素：");
	for(int l=0;l<RealArrElNum;l++)
	{
```

```
        scanf("%d",&TestArr[l]);
    }
    printf("原始数据： ");
    for(int n=0;n<RealArrElNum;n++)
    {
        printf("%3d",TestArr[n]);
    }
    printf("\n\n");

    DeleteSameEls(TestArr,RealArrElNum);
    printf("\n 删后数据： ");
    for(int m=0;m<RealArrElNum;m++)
    {
        printf("%3d",TestArr[m]);
    }
    printf("\n");
    return 0;
}
```

6.8 编写函数判断任意给定的二维整数组（100x100）中是否有相同元素

```
#include "stdio.h"

#define RealRow 100
#define Column 100

int OutArr[Column*RealRow];
int Counter=0;
bool IsHaveSameEle(int InArray[][Column],int row)
{

    int i,j;
    int k,l;
    bool flag=false;


    for(i=0;i<row;i++)
    {
        for(j=0;j<Column;j++)
        {
            int key=InArray[i][j];
            bool HaveSame;
            HaveSame=false;

            for(k=i;k<row;k++)
            {
```

```
                    if(HaveSame)
                    {
                        break;
                    }

                    if(k==i)
                    {
                        l=j+1;
                    }
                    else
                    {
                        l=0;
                    }
                    for(l;l<Column;l++)
                    {
                        if(InArray[k][l]==key)
                        {
                            flag=true;
                            if(!HaveSame)
                            {
                                bool TempFlag=false;
                                for(int Temp=0;Temp<Counter;Temp++)
                                {
                                    if(OutArr[Temp]==key)
                                    {
                                        TempFlag=true;
                                        break;
                                    }
                                }
                                if(!TempFlag)
                                {
                                    OutArr[Counter]=key;
                                    Counter++;
                                    HaveSame=true;
                                    break;
                                }
                            }
                        }
                    }
                }
            }
        }
        return flag;
}
```

```
void main(void)
{
    int AnArray[RealRow][Column];
    int m,n;
    printf("输入%d X %d 的二维数组：\n",RealRow,Column);
    for(m=0;m<RealRow;m++)
    {
        for(n=0;n<Column;n++)
        {
            scanf("%d",&AnArray[m][n]);
        }
    }
    if(IsHaveSameEle(AnArray,RealRow))
    {
        printf("有%d 个相同的元素如下：\n",Counter);
        for(int p=0;p<Counter;p++)
        {
            printf("%d    ",OutArr[p]);
        }
        printf("\n");
    }
    else
    {
        printf("无相同的元素!\n");
    }
}
```

6.10 编写函数，把矩阵 A 转置 A'，并存入 A 中。

```
#include "stdio.h"

#define RealRow      4
#define column       3

void ToTranspose(int InMatrix[][column],int row)
{
    int TempMatrixP[column][RealRow];
    int i,j;
    for(i=0;i<row;i++)
    {
        for(j=0;j<column;j++)
        {
            TempMatrixP[j][i]=InMatrix[i][j];
        }
    }
```

```
    int TempArray[RealRow*column];
    int counter=0;

    for(i=0;i<column;i++)
    {
        for(j=0;j<row;j++)
        {
            TempArray[counter]=TempMatrixP[i][j];
            counter++;
        }
    }

    counter=0;

    for(i=0;i<row;i++)
    {
        for(j=0;j<column;j++)
        {
            InMatrix[i][j]=TempArray[counter];
            counter++;
        }
    }
}

void main(void)
{
    int AnyMatrix[RealRow][column];
    int m,n;
    int CalCounter;

    CalCounter=0;
    printf("输入%d X %d 的矩阵：\n",RealRow,column);
    for(m=0;m<RealRow;m++)
    {
        for(n=0;n<column;n++)
        {
            scanf("%d",&AnyMatrix[m][n]);
        }
    }
    ToTranspose(AnyMatrix,RealRow);

    printf("输出转置后的%d X %d 的矩阵如下：\n",column,RealRow);
    for(m=0;m<RealRow;m++)
    {
```

```
        for(n=0;n<column;n++)
        {
            printf("%-4d",AnyMatrix[m][n]);
            CalCounter++;
            if(CalCounter%RealRow==0)
            {
                printf("\n");
            }
        }
    }
}
```

6.11 编写函数，找出给定二维整数组 A 中的所有鞍点。若一个数组元素 A[i,j]正好是矩阵 A 第 i 行的最小值，第 j 列的最大值，则称其为 A 的一个鞍点。

```
#include "stdio.h"
#define RealRow          3
#define column           4
void FindAnPoint(int InArray[][column],int row)
{
    int i,j;
    int AnPoints=0;
    int TempAnPoints[RealRow*column][3];

    for(i=0;i<RealRow;i++)
    {
        for(j=0;j<column;j++)
        {
            int key;
            key=InArray[i][j];

            bool RowMini;
            bool ColumnMax;

            RowMini=true;
            ColumnMax=true;

            for(int m=0;m<column;m++)
            {
                if(InArray[i][m]<key)
                {
                    RowMini=false;
                    break;
                }
            }
```

```
                for(int n=0;n<RealRow;n++)
                {
                    if(InArray[n][j]>key)
                    {
                        ColumnMax=false;
                        break;
                    }
                }

                if(RowMini&&ColumnMax)
                {
                    TempAnPoints[AnPoints][0]=key;
                    TempAnPoints[AnPoints][1]=i+1;
                    TempAnPoints[AnPoints][2]=j+1;
                    AnPoints++;
                }
            }
        }
        if(AnPoints>0)
        {
            printf("\n%d 个鞍点及其下标如下;\n",AnPoints);
            printf("%-6s%-6s%-6s%-6s\n","序号","鞍点","行","列");
            for(int p=0;p<AnPoints;p++)
            {

        printf("%-6d%-6d%-6d%-6d",p+1,TempAnPoints[p][0],TempAnPoints[p][1],TempAnPoints
[p][2]);
                printf("\n");
            }
        }
        else
        {
            printf("\n 无鞍点\n");
        }
}
void main(void)
{
        int AnyArray[RealRow][column];
        int x,y;

        printf("输入%d X %d 的二维数组：\n",RealRow,column);
        for(x=0;x<RealRow;x++)
        {
            for(y=0;y<column;y++)
```

```
        {
            scanf("%d",&AnyArray[x][y]);
        }
    }
    FindAnPoint(AnyArray,RealRow);
}
```

6.18 数组 A 未排序；今有一个索引数组 B 保存 A 的下标。编程序，不改变数组 A，只改变数组 B 完成对 A 的排序。如下图所示:

A	B
10	1
7	2
5	3
8	4
4	5

排序前

A	B
10	5
7	3
5	2
8	4
4	1

排序后

```
#include <stdio.h>
void main()
{
int a[5]={10,7,5,8,4};
int b[5]={1,2,3,4,5};
int i,j,k,r;
for(i=0;i<4;i++)
{
    k=i;
    for(j=i+1;j<5;j++)
    {
        if(a[b[j]-1]<a[b[k]-1])
        {
            k=j;
        }
    }
    if(k!=i)
    {
        r=b[i];
        b[i]=b[k];
        b[k]=r;
    }
}
for(i=0;i<5;i++)
{
    printf("%-4d",a[b[i]-1]);
```

```
}
printf("\n");
}
```

6.20 设整数集合 M 定义如下：

1）1 ∈M ；

2）若 x ∈M ，则 2x+1 ∈M ，3x+1 ∈M ；

3）没有别的整数属于集合 M 。

编程序按递增顺序生成并输出集合 M 的前 30 项。再编一个函数，对任意给定整数 z ，判断它是否属于集合 M 。

参考答案：

(1)编程序按递增顺序生成并输出集合 M 的前 30 项。

```
#include <stdio.h>
void main(){
   int count;
   int n;
   int i;
   int a[30];
   a[0]=1;
   count=1;
   n=2;
   while (count<=30)
   {
        for (i=0;i<=count-1;i++)
             if ((n==a[i]*2+1)||(n==a[i]*3+1))
             {
                  count++;
                  a[count-1]=n;
                  break;
             }
        n++;
   }
   for (i=0;i<30;i++)
        printf("%4d",a[i]);
}
```

(2) 再编一个函数，对任意给定整数 z ，判断它是否属于集合 M 。

```
#include <stdio.h>
int fun(int m)
{
      int result1=0,result2=0;
      if(m==1)
           return 1;
      if(m>1)
```

```
    {
        int a=(m-1)%2,b=(m-1)%3;
        if(a==0)
            result1=fun((m-1)/2);
        if(b==0)
            result2=fun((m-1)/3);
        return(result1||result2);
    }
    else
        return 0;
}
void main(){
   int data;
   scanf("%d",&data);
   if(fun(data)==1)
        printf("yes\n");
   else
        printf("no\n");
}
```

第（2）题的另一种解法

```
#include "stdio.h"
void Out_M_N(int Output_Nums)//输出集合中排序后的数据
{
    int ArrRes[10000];//最多可输出 10000 个 M 集合中的数据

    int i,j,k;

    ArrRes[0]=1;

    i=j=0;
    k=1;
    while(k<Output_Nums)
    {
        if((2*ArrRes[i]+1)<(3*ArrRes[j]+1))
        {
            ArrRes[k]=2*ArrRes[i]+1;
            i++;
        }
        else if((2*ArrRes[i]+1)==(3*ArrRes[j]+1))
        {
            ArrRes[k]=2*ArrRes[i]+1;
            i++;
            j++;
```

```
            }
            else
            {
                ArrRes[k]=3*ArrRes[j]+1;
                j++;

            }
            k++;
        }
        for(int m=0;m<Output_Nums;m++)
        {
            if(m%10==0)
            {
                printf("\n");
            }
            printf("%-8d",ArrRes[m]);
        }
}

void Delete_OneElement(int Arr[],int Delete_Num,int Arr_Nums)//删除数组中的元素
{
        for(int i=Delete_Num;i<Arr_Nums-1;i++)
        {
            Arr[i]=Arr[i+1];
        }

}
bool IsInclude(int Z)//判断是否属于集合 M
{
        int TempArr[100]={0};

        int Current_Nums=1;
        TempArr[0]=Z;
        if(Z==1)
            return true;
        while(Current_Nums!=0)
        {
            int Pre_2_Value;
            int Pre_3_Value;
            Pre_2_Value=(TempArr[0]-1)%2;
            Pre_3_Value=(TempArr[0]-1)%3;

            if((Pre_2_Value!=0)&&(Pre_3_Value!=0)&&(Current_Nums==1))
                return false;
```

```
            else if((Pre_2_Value!=0)&&(Pre_3_Value!=0)&&(Current_Nums>1))
            {
                Delete_OneElement(TempArr,0,Current_Nums);
                Current_Nums--;
                continue;//从头开始判断数组中的元素
            }
            if(Pre_2_Value==0)
            {
                int Temp_2;
                Temp_2=(TempArr[0]-1)/2;
                if(Temp_2==1)
                    return true;
                else if(Temp_2>1)
                {
                    TempArr[Current_Nums]=Temp_2;
                    Current_Nums++;
                }

            }
            if(Pre_3_Value==0)
            {
                int Temp_3;
                Temp_3=(TempArr[0]-1)/3;
                if(Temp_3==1)
                    return true;
                else if(Temp_3>1)
                {
                    TempArr[Current_Nums]=Temp_3;
                    Current_Nums++;
                }
            }
            Delete_OneElement(TempArr,0,Current_Nums);
            Current_Nums--;
        }
        return false;
}
int main(int argc, char* argv[])
{
        int TestNum;
        TestNum=0;
        while(TestNum!=-1)
        {
            printf(" 输入要判断的数(输入-1 退出)\n");
            scanf("%d",&TestNum);
```

```
        if(TestNum==-1)
            break;
        printf("  下面是 M 集合的前%d 项\n",TestNum);
        Out_M_N(TestNum);
        printf("\n");

        if(IsInclude(TestNum))
            printf("%d 属于集合 M\n\n",TestNum);
        else
            printf("%d 不属于集合 M\n\n",TestNum);
    }

    return 0;
}
```

6.22 判断一个字符序列中(与)、[与]、{与}是否配对且互不相交。

注：程序未做全面测试，仅供参考。

```
#include "string.h"

#define SZ 500

void InitStack(int *StackTop)
{
    *StackTop=0;
}
bool IsStackEmpty(int *StackTop)
{
    if((*StackTop)==0)
    {
        return true;
    }
    else
    {
        return false;
    }
}
bool PushStack(char CharStack[],char InChar,int *StackTop,int StackSize)
{
    if((*StackTop)>(StackSize-1))
    {
        printf("栈满，压入失败！\n");
        return false;
    }
    else
    {
```

```
            CharStack[*StackTop]=InChar;
            *StackTop=*StackTop+1;
            return true;
        }
}
bool PopStack(char CharStack[],char* PopChar,int *StackTop)//出栈并删除栈顶元素
{
        if((*StackTop)<=0)
        {
            printf("栈空，弹出失败！\n");
            return false;
        }
        else
        {
            *StackTop=*StackTop-1;
            *PopChar=CharStack[*StackTop];
            return true;
        }
}
bool GetTopEleMentOfStack(char CharStack[],char* TopCharOfStack,int *StackTop)//只获取栈顶元素，不改变栈顶
{
        if((*StackTop)<=0)
        {
            printf("栈空，获取栈顶元素失败！\n");
            return false;
        }
        else
        {
            *TopCharOfStack=CharStack[*StackTop-1];
            return true;
        }
}

//目前能判断两种情况：1.返回值为 true, 配对且互不相交；2. 返回值为 false, 不配对或相交
//特殊情况：默认要判断的字符串至少有一个括号成分，如果字符串不包含任何括号成分，则
//视为配对且不相交，函数返值为 true,但会有具体提示信息
bool IsPaAndNotInt(char *TheString)
{
        int    StackSize=SZ;
        char CharStack[SZ];
        int StackTop=0;
```

```
bool IsPairingAndNotIntersecting=true;
bool HaveBracketEle=false;

int StringLenth;

StringLenth=strlen(TheString);

InitStack(&StackTop);

for(int i=0;i<StringLenth;i++)
{
    if(TheString[i]=='('||TheString[i]=='['||TheString[i]=='{')
    {
        HaveBracketEle=true;
        if(!PushStack(CharStack,TheString[i],&StackTop,StackSize))
        {
            return false;
        }
    }
    else if(TheString[i]==')')
    {
        HaveBracketEle=true;
        if(IsStackEmpty(&StackTop))//栈空
        {
            printf("有()不配对,缺少(。\n");
            return false;
        }
        char TempTopStackChar;
        GetTopEleMentOfStack(CharStack,&TempTopStackChar,&StackTop);
        if(TempTopStackChar!='(')//栈不空但不配对或相交
        {
            if(TempTopStackChar=='[')
            {
                printf("可能有[]与()相交或)不配对,缺少(，如([)]\n");
                return false;
            }
            if(TempTopStackChar=='{')
            {
                printf("可能有{}与()相交或)不配对,缺少(，如({)}\n");
                return false;
            }
        }
        else//暂时配对
        {
```

```
            PopStack(CharStack,&TempTopStackChar,&StackTop);
        }
    }
    else if(TheString[i]==']')
    {
        HaveBracketEle=true;
        if(IsStackEmpty(&StackTop))//栈空
        {
            printf("有[]不配对,缺少[。\n");
            return false;
        }
        char TempTopStackChar;
        GetTopEleMentOfStack(CharStack,&TempTopStackChar,&StackTop);
        if(TempTopStackChar!='[')//栈不空但不配对或相交
        {
            if(TempTopStackChar=='(')
            {
                printf("可能有()与[]相交或]不配对,缺少[，如[(])\n");
                return false;
            }
            if(TempTopStackChar=='{')
            {
                printf("可能有{}与[]相交或]不配对,缺少[，如[{]}\n");
                return false;
            }
        }
        else//暂时配对
        {
            PopStack(CharStack,&TempTopStackChar,&StackTop);
        }
    }
    else if(TheString[i]=='}')
    {
        HaveBracketEle=true;
        if(IsStackEmpty(&StackTop))//栈空
        {
            printf("有{}不配对,缺少{。\n");
            return false;
        }
        char TempTopStackChar;
        GetTopEleMentOfStack(CharStack,&TempTopStackChar,&StackTop);
        if(TempTopStackChar!='{')//栈不空但不配对或相交
        {
            if(TempTopStackChar=='(')
```

```
                    {
                        printf("可能有()与{}相交或}不配对,缺少{，如{(})\n");
                        return false;
                    }
                    if(TempTopStackChar=='[')
                    {
                        printf("可能有[]与{}相交或}不配对,缺少{，如{[}]\n");
                        return false;
                    }
                }
                else//暂时配对
                {
                    PopStack(CharStack,&TempTopStackChar,&StackTop);
                }
            }
        }

        if(!IsStackEmpty(&StackTop))
        {
            printf("有括号不配对，可能缺少)、]或}\n");
            return false;
        }
        if(HaveBracketEle&&IsPairingAndNotIntersecting)
        {
            printf("括号配对且互不相交。\n");
            return true;
        }
        if(!HaveBracketEle)
        {
            printf("字符串不包含任何括号成分。\n");
            return true;
        }
}
void main()
{
        char TestString[SZ];

        char WillContinue='Y';//为'N'表示不再判断下一个字符串
        while(WillContinue!='N'&&WillContinue!='n')
        {
            printf("\n 请输入一个字符串:");
            scanf("%s",TestString);
            IsPaAndNotInt(TestString);
            printf("\n 继续判断下一个字符串吗(Y/N)？： ");
```

```
            getchar();
            scanf("%c",&WillContinue);
            printf("\n");
        }
}
```

6.23 编写函数 strchange，把给定的字符串反序。

```
#include "string.h"
#include "stdio.h"
#define Len 1000

bool StrChange(char InString[])
{
int StrLen=strlen(InString);
int Mid;
int i;
Mid=StrLen/2;

for(i=0;i<Mid;i++)
{
        char TempCh;
        TempCh=InString[i];
        InString[i]=InString[StrLen-i-1];
        InString[StrLen-i-1]=TempCh;
}
return true;
}

void main(void)
{
        char AnyString[Len];
        printf("\n 输入字符串：");
        scanf("%s",AnyString);
        StrChange(AnyString);

        printf("\n 字符串反序后结果：\n%s\n",AnyString);
}
```

6.24 编写函数 strcat，把给定的两个字符串连接起来。

```
#include "string.h"
#include "stdio.h"

#define Len1 100
#define Len2 100
#define Len    1000
```

```
int StrCat(char InString1[],char InString2[],char ResString[])
{
    int StrLen1,StrLen2,ResLen;
    int i;
    StrLen1=strlen(InString1);
    StrLen2=strlen(InString2);
    ResLen=StrLen1+StrLen2;

    for(i=0;i<StrLen1;i++)
    {
        ResString[i]=InString1[i];
    }
    for(i=0;i<StrLen2;i++)
    {
        ResString[StrLen1+i]=InString2[i];
    }
    ResString[ResLen]='\0';

    return ResLen;
}

void main(void)
{
    char String1[Len1];
    char String2[Len2];
    char CatString[Len];

    printf("\n 输入第一个字符串:\n");
    scanf("%s",String1);

    printf("\n 输入第二个字符串:\n");
    scanf("%s",String2);

    StrCat(String1,String2,CatString);
    printf("\n 连接后结果：\n%s\n",CatString);

}
```

6.27 整理名字表。编写程序，输入任意顺序的名字表，将其按字典顺序排序并输出。

```
#include<stdio.h>
#include <string.h>
#define N 5

bool NameSort(char InName[][20],int Count)
```

```
{
    int i,j,k;
    char Temp[20];
    for(i=0;i<Count;i++)
    {
        k=i;
        for(j=i+1;j<Count;j++)
        {
            if(strcmp(InName[j],InName[k])<0)
            {
                k=j;
            }
        }
        if(k!=i)
        {
            strcpy(Temp,InName[i]);
            strcpy(InName[i],InName[k]);
            strcpy(InName[k],Temp);
        }
    }
    printf("\n 名字排序后：\n");
    for(i=0;i<Count;i++)
    {
        puts(InName[i]);
    }
    return true;
}


void main()
{
    char AnyName[N][20];
    printf("\n 输入%d 个名字：\n",N);

    for(int i=0;i<N;i++)
    {
        gets(AnyName[i]);
    }
    NameSort(AnyName,N);
}
```

6.28 约瑟夫(Josephus)问题

古代某法官要判决 n 个犯人死刑，他有一条荒唐的逻辑，将犯人首尾相接排成圆圈，然后从第 s 个人开始数起，每数到第 m 个犯人，则拉出来处决；然后再数 m 个，数到的

犯人再处决；. . . ；但剩下的最后一个犯人可以赦免。编程序，给出处决顺序，并告知哪一个人活下来。

```
#include "stdio.h"
void Jhus(int n,int s,int m,int Fan[])
{
	int i;
	int TempN;
	int TempOut;
	int TempCount=1;
	TempOut=s;
	TempN=n;
	while(TempN>1)
	{
		TempOut=(m+TempOut-1)%TempN;
		if(TempN==n)
			TempOut--;
		printf("第%d 个被处决的：%d\n",TempCount,Fan[TempOut]);
		TempN--;
		TempCount++;

		for(i=TempOut;i<TempN;i++)
		{
			Fan[i]=Fan[i+1];
		}
	}
	printf("活下来的人是：%d\n",Fan[0]);

}
int main(int argc, char* argv[])
{
	int a[]={1,2,3,4,5,6,7,8,9,10,11,12,13};
	Jhus(13,4,3,a);
	return 0;
}
```

P63_21（习题集）. 编写函数，用“逐步增加递增子序列”法对整形数组进行排序

```
#include "stdio.h"
void AddSort(int Arr[],int ElNum)
{
	int i,j,k,r;
	for(i=1;i<ElNum;i++)
	{
		j=i-1;
		while((Arr[j]>Arr[i])&&(j>=0))
```

```
            {
                j=j-1;
            }
            r=Arr[i];
            for(k=i-1;k>=j+1;k--)
            {
                Arr[k+1]=Arr[k];
            }
            Arr[j+1]=r;
        }
}

int main(int argc, char* argv[])
{
    int TestArr[1000];
    int RealArrElNum;
    printf("输入数组元素数（小于 1000）：");
    scanf("%d",&RealArrElNum);
    printf("输入数组元素：");
    for(int l=0;l<RealArrElNum;l++)
    {
        scanf("%d",&TestArr[l]);
    }
    printf("原始数据：");
    for(int n=0;n<RealArrElNum;n++)
    {
        printf("%3d",TestArr[n]);
    }
    printf("\n\n");

    AddSort(TestArr,RealArrElNum);
    printf("\n 排序数据：");
    for(int m=0;m<RealArrElNum;m++)
    {
        printf("%3d",TestArr[m]);
    }
    printf("\n");
    return 0;
}
```

第七章

7.6 下述程序执行后，输出结果是什么？

```
#include   <stdio.h>

char *p[2][3]={ "abc","defgh","ijkl","mnopqr","stuvw","xyz"};
main()
{
      printf("%c\n",***(p+1));
      printf("%c\n",**p[0]);
      printf("%c\n",(*(*(p+1)+1))[4]);
      printf("%c\n",*(p[1][2]+2));
      printf("%s\n",**(p+1));
}
```

参考答案：

```
m
a
w
z
mnopqr
```

7.8 编写函数，求给定字符串的长度

```
#include "stdio.h"
#include "string.h"
int length(char *str)
{
       int i=0;
        while(str[i]!='\0')
            i++;
        return i;
}
void main()
{
       char s[20];
       printf("输入\n");
      gets(s);
      printf("串长=%3d\n",length(s));
}
```

7.27 编写比较两个字符串 s1、s2 大小的程序。当 s1>s2 时，输出 1；当 s1<s2 时，输出-1；当 s1==s2 时，输出 0。

参考答案：

```
#include "stdio.h"
int str_cmp(char* str1,char* str2)
```

```
{
    while (*str1!='\0'&&*str2!='\0')
    {
        if (*str1>*str2)
            return 1;
        else if (*str1<*str2)
            return -1;
        str1++;
        str2++;
    };
    if (*str1=='\0'&&*str2=='\0')
        return 0;
    else if (*str1!='\0')
            return 1;
    else return -1;
}

void main()
{
    char s1[100],s2[100];
    printf("please input the first string:");
    scanf("%s",s1);
    printf("\n");
    printf("please input the second string:");
    scanf("%s",s2);
    printf("\n");
    printf("the result is: %d",str_cmp(s1,s2));
    printf("\n");
}
```

7.9 编写函数，分别求给定字符串中的大写字母、小写字母、空格数字、其他字符的数目

```
#include "stdio.h"
void    countchar(char *p)
{
    int s[5]={0,0,0,0,0};
    while(*p!='\0')
    {
        if(*p>='A'&&*p<='Z')
            s[0]++;
        else if (*p>='a'&&*p<='z')
            s[1]++;
        else if(*p>='0'&&*p<='9')
            s[2]++;
        else if(*p==' ')
            s[3]++;
```

```
            else s[4]++;
            p++;

        }
        printf("大写字母的个数为：%d\n 小写字母的个数为：%d\n 数字的个数为：%d\n 空格的个数为：%d\n 其他字符的个数为：%d\n",s[0],s[1],s[2],s[3],s[4]);
}
void main()
{

        char str[100];
        printf("输入字符串\n");
        gets(str);
        countchar(str);
}
```

7.10 编函数，把给定字符串的从 m 开始以后的字符复制到另一个指定的字符串中。

```
#include "stdio.h"
//默认 m 不大于原字符串长度
void CopySomeChar(char *Source,int m,char *Desti)
{
        int i=0;
        while(*(Source+i)!='\0')
        {
            *(Desti+i)=*(Source+i+m-1);
            i++;
        }
        *(Desti+i)='\0';
}

int main(int argc, char* argv[])
{
        char Source[1000];
        char Desti[1000];
        int StartCopy;
        printf("Input Soruce String:      ");
        scanf("%s",Source);
        printf("Input Start:      ");
        scanf("%d",&StartCopy);
        CopySomeChar(Source,StartCopy,Desti);
        printf("Deti String:      %s\n",Desti);

        return 0;
}
```

7.12 编写函数，用指针作参数，实现把字符串 str1 复制到字符串 str2 中

```
#include "stdio.h"
void    copysr(char *str1,char *str2)
{
      while(*str1!='\0')
            *str2++=*str1++;
      *str2='\0';

}

void main()
{
      char str1[100],str2[100];
      printf("输入字符串\n");
      scanf("%s%s",str1,str2);
      copysr(str1,str2);
      printf("\n%s\n",str2);
}
```

7.16 编写函数，分别用指针传递参数，实现两个字符串变量值的交换

```
#include "stdio.h"
#include "string.h"
void    changestr(char *str1,char *str2)
{
      char temp;
      int len,i=0;
      len=strlen(str1)>strlen(str2)?strlen(str1):strlen(str2);
      while(i<len+1)
      {
            temp=*(str2+i);
            *(str2+i)=*(str1+i);
            *(str1+i)=temp;
            i++;
      }
}
void main()
{
      char str1[100],str2[100];
      printf("输入字符串\n");
      scanf("%s%s",str1,str2);
      changestr(str1,str2);
      printf("\n%s\n%s\n",str1,str2);
}
```

7.22 编写函数，用指针形式访问数组元素，把给定的 int 型矩阵转置。

```
#include "stdio.h"
#define N 5
void    reverse(int *a)
{
      int m;
      for(int i=0;i<N;i++)
            for(int j=0;j<i;j++)
            {
                  m=*(a+i*N+j);
                  *(a+i*N+j)=*(a+j*N+i);
                  *(a+j*N+i)=m;
            }
}
void main()
{
      int a[N][N],i,j;
      printf("输入矩阵\n");
      for(i=0;i<N;i++)
            for(j=0;j<N;j++)
                  scanf("%d",&a[i][j]);

      reverse(&a[0][0]);
      for(i=0;i<N;i++)
      {
            for(j=0;j<N;j++)
                  printf("%3d",a[i][j]);
            printf("\n");
      }
}
```

7.26 已知以指针数组形式给出的字符串列表 table。编写函数，输入一个字符串，查 table 表，若查到则输出此字符串的位置，否则输出 0.

```
#include "stdio.h"
#include "string.h"
#define N 5
#define M 25
int    search(char *table[N],char *str)
{
      int i=0;

      while(strcmp(table[i],str)!=0&&i<N) i++;
      if(i==N)
            return 0;
      else
            return i+1;
```

```
}
void main()
{
    char *str_table[N]={"china","noon","english","hello","morning"},str[M];
    printf("字符串表为：\n");
    printf("序号    字符串\n");
    for(int i=0;i<N;i++)
    {
        printf("%d        %s\n",i,str_table[i]);
    }

    printf("输入欲查询的串\n");
    scanf("%s",str);
    printf("查询的位置为：%d\n",search(str_table,str));
}
```

P74_23（习题集）. 一个古怪的.....酒窖只能装 120 瓶酒，酒的库存是怎样变化的.....

答案一：

```
#define MaxNum 24
char EmptyChar='@';
int CurNum=0;//当前已经有的酒数量
char WineCellarArr[MaxNum+1];

/*
如果存在 OneType 类型，函数返回 true,不存在返回 false
*/
bool HaveTheType(char SearchType,
                 int *LeftEmptyNum,int *LeftEmptyStartIndex,
                 int *RightEmptyNum,int *RightEmptyStartIndex,
                 int *HaveNum,int *StartIndex
                )
{
    int i;

    int TempLeftNum;
    int TempLeftIndex;
    int TempRightNum;
    int TempRightIndex;

    TempLeftNum=0;
    TempLeftIndex=-1;
    TempRightNum=0;
    TempRightIndex=-1;
```

```
int TempNum;
int TempStartIndex;
int TempEndIndex;

TempNum=0;
TempStartIndex=-1;
TempEndIndex=-1;

bool FirstFindTheType;
bool TheTypeSearchEnd;

FirstFindTheType=false;
TheTypeSearchEnd=false;

for(i=0;i<MaxNum;i++)
{
    if(WineCellarArr[i]==SearchType)
    {
        if(!FirstFindTheType)
        {
            FirstFindTheType=true;
            TempNum++;
            TempStartIndex=i;
            TempEndIndex=i;
        }
        else
        {
            TempNum++;
            TempEndIndex=i;
        }
    }
    else
    {
        if(FirstFindTheType)
        {

            FirstFindTheType=false;
            TheTypeSearchEnd=true;

            int L;
            int R;
            //处理找到左边
            L=TempStartIndex;
            if(L!=0)
```

```
{
    if(WineCellarArr[L-1]==EmptyChar)
    {
        while((L>0))
        {
            L--;
            if(WineCellarArr[L]==EmptyChar)
            {
                TempLeftNum++;
                TempLeftIndex=L;
            }
            else
            {
                break;
            }
        }
    }
}
//处理找到右边
R=TempEndIndex;
if(R!=(MaxNum-1))
{
    if(WineCellarArr[R+1]==EmptyChar)
    {
        TempRightIndex=R+1;
        while((R<(MaxNum-1)))
        {
            R++;
            if(WineCellarArr[R]==EmptyChar)
            {
                TempRightNum++;
            }
            else
            {
                break;
            }
        }
    }
}
//准备返回
*LeftEmptyNum=TempLeftNum;
*LeftEmptyStartIndex=TempLeftIndex;
*RightEmptyNum=TempRightNum;
*RightEmptyStartIndex=TempRightIndex;
```

```
                *HaveNum=TempNum;
                *StartIndex=TempStartIndex;
                return true;
            }
        }
    }

    *LeftEmptyNum=TempLeftNum;
    *LeftEmptyStartIndex=TempLeftIndex;
    *RightEmptyNum=TempRightNum;
    *RightEmptyStartIndex=TempRightIndex;
    *HaveNum=TempNum;
    *StartIndex=TempStartIndex;

    if((TempEndIndex==(MaxNum-1))&&(FirstFindTheType)&&(!TheTypeSearchEnd))// 最 后找到且后面无空位置
    {
        //只需要处理左边的空位置
        int L;
        //处理找到左边
        L=TempStartIndex;
        if(L!=0)
        {
            if(WineCellarArr[L-1]==EmptyChar)
            {
                while((L>0))
                {
                    L--;
                    if(WineCellarArr[L]==EmptyChar)
                    {
                        TempLeftNum++;
                        TempLeftIndex=L;
                    }
                    else
                    {
                        break;
                    }
                }
            }
        }
        *LeftEmptyNum=TempLeftNum;
        *LeftEmptyStartIndex=TempLeftIndex;
        return true;
    }
```

```
    return false;
}
/*
寻找第一个大于等于 NeedEmptyNum 的空位置，找到返回 true，未找到返回 false,
当买的酒不存在时用来查给定数量的连续空位
*/
bool FindFirstNeedEmptyNumAndPosition(int NeedEmptyNum,int
*FirstFindEmptyPositionIndex)
{
    int i;
    int CurEmptyNum;
    int CurEmptyStartIndex;

    bool EmptyStart=false;
    bool EmptyEnd=true;
    CurEmptyNum=0;
    CurEmptyStartIndex=-1;

    for(i=0;i<MaxNum;i++)
    {
        if(WineCellarArr[i]==EmptyChar)
        {
            if(!EmptyStart&&EmptyEnd)
            {
                EmptyStart=true;
                EmptyEnd=false;
                CurEmptyStartIndex=i;
                CurEmptyNum++;
            }
            else
            {
                CurEmptyNum++;
            }
            if(CurEmptyNum>=NeedEmptyNum)
            {
                *FirstFindEmptyPositionIndex=CurEmptyStartIndex;
                return true;
            }
        }
        else
        {
            if(EmptyStart&&!EmptyEnd)
            {
                EmptyStart=false;
```

```
                    EmptyEnd=true;
                    CurEmptyNum=0;
                    CurEmptyStartIndex=-1;
                }
            }
        }
        return false;
}

void OutPutCurState()
{
        printf("\n 当前酒窖状态为：\n");
        for(int i=1;i<=MaxNum;i++)
        {
            printf("%3d",i);
        }
        printf("\n");
        for(i=0;i<MaxNum;i++)
        {
            printf("%3c",WineCellarArr[i]);
        }
        printf("\n");
        printf("当前共有 %d 瓶酒;%d 个空位置;%c 表示空位置;酒窖共可存放%d 瓶酒。
",CurNum,MaxNum-CurNum,EmptyChar,MaxNum);
        printf("\n");
}
void InitWineCellarArr()
{
        WineCellarArr[MaxNum]='\0';
        int i;
        printf("\n 请输入用于表示空位置的字符:");
        scanf("%c",&EmptyChar);
        getchar();
        for(i=0;i<MaxNum;i++)
        {
            WineCellarArr[i]=EmptyChar;
        }
        printf("\n 每次只能买 1、3、6、12、24 瓶酒，每次只能喝 1、2 瓶酒；酒窖只能装%d
瓶酒。\n",MaxNum);
        printf("\n 买酒或喝酒的输入方式:数量,酒类型：例如买酒: 3,A ;    喝酒: -2,B .\n");
        CurNum=0;
        OutPutCurState();
}
```

```
void main()
{
    InitWineCellarArr();

    int BuyOrDrink;
    char WineType;

    int mTempLeftNum;
    int mTempLeftIndex;
    int mTempRightNum;
    int mTempRightIndex;

    int mTempNum;
    int mTempStartIndex;

    int mFirstFindEmptyPositionIndex;

    printf("请输入买/喝酒数据(买/喝酒数为 0 时结束):");
    scanf("%d,%c",&BuyOrDrink,&WineType);

    while(BuyOrDrink!=0)
    {

    if((BuyOrDrink==-1||BuyOrDrink==-2||BuyOrDrink==1||BuyOrDrink==3||BuyOrDrink==6||

    BuyOrDrink==12||BuyOrDrink==24)&&(((WineType>='A')&&(WineType<='Z'))||((WineTy
pe>='a')&&(WineType<='z'))))
        {
            if(BuyOrDrink<0)//喝酒
            {
                if(HaveTheType(WineType,&mTempLeftNum,&mTempLeftIndex,
                    &mTempRightNum,&mTempRightIndex,
                    &mTempNum,&mTempStartIndex))
                {
                    BuyOrDrink=-BuyOrDrink;
                    if(mTempNum<BuyOrDrink)
                    {
                        printf("\n%C 类型酒只剩%d 瓶,对不起，不够喝，请及时买这种
酒!\n",WineType,mTempNum);
                        OutPutCurState();
                    }
                    else
                    {
                        for(int i=mTempStartIndex;i<mTempStartIndex+BuyOrDrink;i++)
```

```
                    {
                        WineCellarArr[i]=EmptyChar;
                    }
                    printf("\n 喝掉%d 瓶%C 类型酒，目前还剩%d 瓶，请及时买这种
酒!\n",BuyOrDrink,WineType,mTempNum-BuyOrDrink);
                    CurNum=CurNum-BuyOrDrink;
                    OutPutCurState();
                }
            }
            else
            {
                printf("\n 对不起，酒窖目前没有%C 类型酒,请购买后再
喝!\n",WineType);
                OutPutCurState();
            }
        }
        else//买酒
        {
            if(HaveTheType(WineType,&mTempLeftNum,&mTempLeftIndex,
                &mTempRightNum,&mTempRightIndex,
                &mTempNum,&mTempStartIndex))
            {
                //已有该类酒的情况
                if((mTempLeftNum+mTempRightNum)>=BuyOrDrink)
                {

    if((mTempLeftNum>=BuyOrDrink)||(mTempRightNum>=BuyOrDrink))
                    {
                        if(mTempLeftNum>=BuyOrDrink)
                        {
                            for(int
q=mTempStartIndex-1;q>=mTempStartIndex-BuyOrDrink;q--)
                            {
                                WineCellarArr[q]=WineType;
                            }
                        }
                        else if(mTempRightNum>=BuyOrDrink)
                        {
                            for(int p=0;p<BuyOrDrink;p++)
                            {
                                WineCellarArr[mTempRightIndex+p]=WineType;
                            }
                        }
                        CurNum=CurNum+BuyOrDrink;
```

```
                    OutPutCurState();
                }
                else
                {
                    int TempLeftNeedStore=mTempLeftNum;
                    int TempRightNeedStore=BuyOrDrink-TempLeftNeedStore;
                    for(int m=0;m<TempLeftNeedStore;m++)
                    {
                        WineCellarArr[mTempLeftIndex+m]=WineType;
                    }
                    for(int n=0;n<TempRightNeedStore;n++)
                    {
                        WineCellarArr[mTempRightIndex+n]=WineType;
                    }
                    CurNum=CurNum+BuyOrDrink;
                    OutPutCurState();
                }
            }
            else
            {
                printf("\n 对不起，按存放要求酒窖目前没有足够的空间存放%c
类酒，请先喝掉一些再买!\n",WineType);
                OutPutCurState();
            }
        }
        else
        {

    if(FindFirstNeedEmptyNumAndPosition(BuyOrDrink,&mFirstFindEmptyPositionIndex))
            {
                //有连续空间可存
                for(int
j=mFirstFindEmptyPositionIndex;j<mFirstFindEmptyPositionIndex+BuyOrDrink;j++)
                {
                    WineCellarArr[j]=WineType;
                }
                CurNum=CurNum+BuyOrDrink;
                OutPutCurState();
            }
            else
            {
                printf("\n 对不起，酒窖目前没有足够的空间存放%c 类酒，请先
喝掉一些再买!\n",WineType);
                OutPutCurState();
```

```
                        }
                    }
                }
            }
            else
            {
                printf("可能输入数据有错误,请重新输入数据\n");
            }
            printf("请输入买/喝酒数据(买/喝酒数为 0 时结束):");
            scanf("%d,%c",&BuyOrDrink,&WineType);
            getchar();
        }
}
```

答案二（由李泽兴同学提供）：

```
/注意：本程序所有数组从 1 开始命名
//在有 check point 标记的地方加断点调试效果更好
//本程序相比上一个版本修改了 void fill();函数中的一些错误并修复了不能填充左边空酒位的问题
/*版权信息
本程序版权为吉林大学计科 4 版李泽兴所有
感谢李河老师提供的 BUG REPORT
感谢计科 15 级 7 班一位不愿署名的同学提供的 BUG REPORT
本程序的被授权者可以
1、在不删除任何注释的前提下使用或编辑本程序
2、在程序设计基础课上上交不含本注释的作业
3、与同学交流本程序的思路
本程序的被授权者不得将本程序及其修改版本用于商业目的、在论坛或其他地方发布以及用于除上述提及的任何目的
本程序为批量授权版本
本程序授权给：吉林大学计算机学院 2015 级全体同学
*/

#include"math.h"
#include"stdio.h"

#define M 121


int address_start[M], address_last[M], address_end[M];
int recent_number = 5, recent_address=1;
int stock_amount = 0;
char storage[M];
bool exist[M];
bool flag = false, flag_plus = false, flag_storage = false, flag_exist = false, virgin = true,
```

```
flag_stock_exist = false, flag_left_exist = false;
char new_wine;
void get_ready();
void get();
void check_plus();
void check_exist();
void check_stock();
void check_storage();
void fill();
void fill_new();
void pull();
void screenoperation();
void check_the_left();
void change_location_and_fill_left_only();
int main()
{
    printf("请输入买酒的数量和种类（如 3,A）或喝酒的数量和种类（如-2,B）\n");
    printf("买酒的数量为 1 或 3 或 6 或 12 或 24\n");
    printf("喝酒的数量为 1 或 2\n");
    while (recent_number != 0)              //如果之前输入不为零
    {
        get();      //读输入函数
        check_plus();           //检查取酒/拿酒函数
        if (flag_plus)        //如果取酒
        {
            check_exist();          //检查是否有酒
                if (flag_exist)       //如果有酒
                {
                    check_storage();                    //检查空间
                    if (flag_storage)                                   //如果有空间
                    {
                        fill();                     //装酒
                    }
                    else              //右边没空间就查左边
                    {
                        check_the_left();
                        if (flag_left_exist)
                        {
                            change_location_and_fill_left_only();          //装酒
                        }
                            else
                        {
                            printf("overflow\n");
                            return 1;
```

```
                        }
                    }
                }
                else                    //如果没这种酒
                {
                    fill_new();             //装第一次出现的酒
                }
            }
            else
            {
                check_exist();          //检查是否有酒
                if (flag_exist)                             //要是有酒
                {
                    check_stock();                          //检查酒够不够
                    if (flag_stock_exist=true)              //酒够耶 棒极了
                    {
                        pull();                             //取酒喝
                    }
                    else                //不够喝？呵呵。
                    {
                        printf("Overflow\n");
                        return 1;
                    }
                }
                else                        //说好的有酒呢？
                {
                    printf("Overflow\n");
                    return 1;
                }
            }
            screenoperation();                          //打印结果
        }
        getchar();                              //使屏幕停留
}
void get_ready()
{
        char temp1;
        for (temp1 = 65; temp1 <= 90; temp1++)
        {
            address_start[temp1] = 0;
            address_last[temp1] = 0;
            address_end[temp1] = 0;
            exist[temp1] = 0;
        }
```

```
    for (temp1 = 0; temp1 <= 101; temp1++)
    {
        storage[temp1] = 122;
    }
}
void get()
{
    char useless_comma;
    scanf("%d%c%c", &recent_number, &useless_comma, &new_wine);
}
void check_plus()
{
    if (recent_number >= 0)
    {
        flag_plus = true;
    }
    else
    {
        flag_plus = false;
    }
}
void check_exist()
{
    if (exist[new_wine])
    {
        flag_exist = 1;
    }
    else
    {
        flag_exist = 0;
    }
}
void check_storage()
{
    if ((address_last[new_wine] + recent_number <= address_end[new_wine])||(address_end[new_wine]==recent_address))
    {
        flag_storage = true;
    }
    else
    {
        flag_storage = false;
    }
}
```

```
void check_stock()
{
    stock_amount = address_end[new_wine] - address_last[new_wine];
    if (stock_amount + recent_number >= 0)
    {
        flag_stock_exist = true;
    }
    else
    {
        flag_stock_exist = false;
    }
}
void fill()
{
    char temp1;
    if (address_last[new_wine] + recent_number <= address_end[new_wine])
    {
        for (temp1 = address_last[new_wine] + 1; temp1 <= address_last[new_wine] + recent_number; temp1++)
        {
            storage[temp1] = new_wine;
        }
        address_last[new_wine] = address_last[new_wine] + recent_number;
    }
    else
    {
        for (temp1 = address_last[new_wine] + 1; temp1 <= address_last[new_wine] + recent_number; temp1++)
        {
            storage[temp1] = new_wine;
        }
        address_last[new_wine] = address_last[new_wine] + recent_number;
        address_end[new_wine] = address_last[new_wine];
        recent_address = address_end[new_wine];
    }
}
void fill_new()
{
    exist[new_wine] = 1;
    char temp1;
    if (virgin)
    {
        address_start[new_wine] = recent_address;
        virgin = false;
```

```
            address_end[new_wine] = recent_address + recent_number-1;
            address_last[new_wine] = address_end[new_wine];
            for (temp1 = address_start[new_wine]; temp1 <= address_end[new_wine]; temp1++)
            {
                storage[temp1] = new_wine;
                storage[temp1];//check point
            }
            recent_address = address_end[new_wine];
        }
        else
        {
            address_start[new_wine] = recent_address+1;
            address_end[new_wine] = recent_address + recent_number;
            address_last[new_wine] = address_end[new_wine];
            for (temp1 = address_start[new_wine]; temp1 <= address_end[new_wine]; temp1++)
            {
                storage[temp1] = new_wine;
                storage[temp1];//check point
            }
            recent_address = address_end[new_wine];
        }
}
void pull()
{
        char temp1,TEMP2;
        int temp5, temp8 = address_start[new_wine]-1;
        char temp6;
        bool temp7 = true;
        if (abs(recent_number) <= (address_last[new_wine] - address_start[new_wine]))
        {
            if (address_end[new_wine] != recent_address)
            {
                for (temp1 = address_last[new_wine]; temp1 >= address_last[new_wine] +
recent_number + 1; temp1--)
                {
                    TEMP2 = address_last[new_wine];//CHECK POINT
                    TEMP2 = temp1;//CHECK POINT
                    storage[temp1] = 0;
                    TEMP2 = storage[temp1];//check point
                }
                address_last[new_wine] = address_last[new_wine] + recent_number;
            }
            else
            {
```

```
                for (temp1 = address_last[new_wine]; temp1 >= address_last[new_wine] + recent_number + 1; temp1--)
                {
                    TEMP2 = address_last[new_wine];//CHECK POINT
                    TEMP2 = temp1;//CHECK POINT
                    storage[temp1] = 0;
                    TEMP2 = storage[temp1];//check point
                }
                address_last[new_wine] = address_last[new_wine] + recent_number;
                recent_address = address_last[new_wine];
                address_end[new_wine] = address_last[new_wine];
            }
        }
        else
        {
            for (temp5 = address_start[new_wine]; temp5 <= address_end[new_wine]; temp5++)
            {
                storage[temp5] = 0;
            }
            exist[new_wine] = 0;
            while ((temp8>=1)&&(temp7))
            {
                if ((storage[temp8] <= 90) && (storage[temp8] >= 65))
                {
                    temp7 = false;
                    temp6 = storage[temp8];
                    address_end[temp6] = address_end[new_wine];
                }
                temp8--;
            }
        }
}
void screenoperation()
{
    char temp1;
    for (temp1 = 1; temp1 <= recent_address; temp1++)
    {
        if ((storage[temp1] <= 90) || (storage[temp1] >= 65))
        {
            printf("%c", storage[temp1]);
        }
        else
        {
            printf(" ");
```

```
            }
        }
        printf("\n");
}
void check_the_left()
{
        int temp,temp1;
        temp = recent_number - (address_end[new_wine] - address_last[new_wine]);
        if ((address_start[new_wine] == 0) || (address_start[new_wine] == 1))
        {
            flag_left_exist = false;
            return;
        }
        for (temp1 = address_start[new_wine]-1; temp1>=address_start[new_wine]-temp-1; temp1--)
        {
            if ((storage[temp1] >= 90) && (storage[temp1] >= 65))
            {
                flag_left_exist = false;
                return;
            }
        }
        flag_left_exist = true;
}
void change_location_and_fill_left_only()
{
        int temp, temp1, temp2,temp4=address_start[new_wine]-1,temp5;
        char temp3=0;
        temp = recent_number - (address_end[new_wine] - address_last[new_wine]);
        temp1 = address_start[new_wine] - temp;
        while ((temp3 == 0)&&(temp4>=1))
        {
            temp3 = storage[temp4];
            temp4--;
        }
        address_end[temp3] = temp1-1;
        address_last[new_wine] = address_end[new_wine];
        address_start[new_wine] = temp1;
        for (temp5 = address_start[new_wine]; temp5 <= address_last[new_wine]; temp5++)
        {
            storage[temp5] = new_wine;
        }
}
```

P85_19（习题集）. 定义含两个参数 x、y 的函数，把字符串 x 中的所有字符 y 删除，同时把删除次数作为函数值

```
#include "stdio.h"
int DeleteSameChar(char *x,char y)
{
	int DeleteCounts=0;
	while(true)
	{
		bool isNoDelete=true;
		int i;

		for(i=0;*(x+i)!='\0';i++)
		{
			if(*(x+i)==y)
			{
				DeleteCounts++;
				isNoDelete=false;
				int j=0;
				while(*(x+i+j)!='\0')
				{
					*(x+i+j)=*(x+i+j+1);
					j++;
				}
			}
		}
		if(isNoDelete)
		{
			return DeleteCounts;
		}
	}
}

int main(int argc, char* argv[])
{
	char x[1000];
	char y;
	printf("Input x String:	");
	scanf("%s",x);
	printf("Input delete char y:	");
	getchar();
	scanf("%c",&y);
	int del=DeleteSameChar(x,y);
	printf("After Delete:	%s\nDelete char:	%d\n",x,del);
```

```
    return 0;
}
```

P85_22（习题集）. 使用指针访问数组元素，编写函数，求给定巨阵的对角线元素之和。

```
#include "stdio.h"
#define Range 4
int GADS(int Arr[][Range])
{
    int TempSum=0;
    int i,j;
    for(i=0;i<Range;i++)
    {
        for(j=0;j<Range;j++)
        {
            if((i==j)||((i+j)==(Range-1)))
            {
                TempSum+=Arr[i][j];
            }
        }
    }
    //如果考虑两对角线可能有一元素重复且需要计算两次的情况
    //需加下面代码
    if(Range%2!=0)
    {
        TempSum+=Arr[Range/2][Range/2];
    }
    return TempSum ;
}

int main(int argc, char* argv[])
{
    int Arr[Range][Range]={{1,5,3,9},{4,8,6,7},{7,7,9,10},{10,11,18,13}};
    int Res;
    Res=GADS(Arr);
    printf("the Array:\n");
    for(int i=0;i<Range;i++)
    {
        for(int j=0;j<Range;j++)
        {
            printf("%5d",Arr[i][j]);
        }
        printf("\n");
    }
    printf("\nthe Result:    %d\n",Res);
```

```
    return 0;
}
```

第八章

8.3 某单位进行选举，有 5 位候选人： zhang,wang,zhao,liu,miao。编写程序，统计每人所得的票数。要求每个人的信息用一个结构体来表示，5 个人的信息使用结构体数组。

参考答案：

```
#include<stdio.h>
#include<string.h>

#define M 5

enum nameType{zhang,wang,zhao,liu,miao};
// 定义结构体类型
typedef struct Candidate
{    nameType name;//姓名
     int count;//票数
}canType;

/* 声明全局结构体数组，count 默认值为 0
若声明为局部变量，则值为随机数，需要在 initial()中对 count 赋 0*/
canType person[M];
int other;//无效票数，初值默认为 0

//函数声明
void out();
void voting();

//主函数
void main()
{    printf("投票前:\n");
     out();

     printf("\n 开始投票:\n");
     voting();//模拟投票

     printf("\n 投票后:\n");
     out();//输出统计结果
}

//函数定义
void out()//输出票数
{    for(int i=0;i<M;i++)
          printf("第%d 位候选人的票数为：%d\n",i+1,person[i].count);
```

```
    printf("无效票数：%d\n",other);
}
void voting()//模拟投票
{   int k;
    printf("请选择候选人：1 zhang,2 wang,3 zhao,4 liu,5 miao（输入 0 结束输入）：");
    scanf("%d",&k);

    while(k!=0)
    {   if(k==1||k==2||k==3||k==4||k==5)
            person[k-1].count++;
        else
            other++;
        printf("请选择候选人：1 zhang,2 wang,3 zhao,4 liu,5 miao（输入 0 结束输入）：");
        scanf("%d",&k);
    }
}
```

8.6 利用结构体类型描述扑克牌。编函数，对任意给定的一副牌进行排序。（去掉王牌；假定梅花<方块<红桃<黑桃）

```
#include "stdio.h"
#define N 4
enum flower{club,diamonds,hearts,spade};//梅花，方块，红桃，黑桃
struct card
{
    int n;
    enum flower f;
    char FlowerName[10];
};
void sort(struct card *p,int n)
{
    struct card temp;
    for(int i=0;i<n-1;i++)
        for(int j=i+1;j<n;j++)
            if(p[i].n>p[j].n)
            {
                temp=p[i];
                p[i]=p[j];
                p[j]=temp;
            }
            else if (p[i].n==p[j].n&&p[i].f>p[j].f)
            {
                temp=p[i];
                p[i]=p[j];
                p[j]=temp;
```

```
                }
}
void main()
{
    int i;
    struct  card  a[N]={{4,club,"梅花"},{2,spade,"黑桃"},{1,hearts,"红桃"},{2,diamonds,"方块"}};
    sort(a,N);
    for(i=0;i<N;i++)
        printf("点数为%d，花色为 %s\n",a[i].n,a[i].FlowerName);
}
```

8.7 设计描述学生成绩单（包括学号、姓名、4 门课程成绩）的数据类型，编写以下函数。
(1)统计每位学生的各门课程(设只有 4 门课程)的成绩及总成绩。
(2)统计全班每门课程的平均分。
(3)输入一名学生的信息。
(4)输出一名学生的信息。
参考答案：

```
#include<stdio.h>
#include<string.h>

#define M 2       //班级总人数
#define L 20     // 姓名长度
#define N 4       //课程总数

//定义结构体类型
typedef struct student{
    int num;                    //学号
    char name[L];               //姓名
    float score[N];             //各科成绩
}stuType;

//函数声明
void initial();                 //初始化
void staOfStudent();            //个人统计
void staOfClass();              //班级统计
void addStudent();              //增加一名学生的信息
void outStuInformation(char str[L]);        //输出一名学生的信息
void search();                  //查找某个学生

//全局变量声明
stuType stu[M];
int curStuCount=0;//当前学生人数

//主函数
```

```
void main()
{    initial();                    //初始化
     printf("\n");

     staOfStudent();                    //个人统计
     printf("\n");

     staOfClass();              //班级统计
     printf("\n");

     search();                  //查找某个学生
}
//函数定义
void initial()//初始化
{    int i;
     printf("该班共%d 人,\n",M);
     for(i=0;i<M;i++)
          addStudent();//增加一名学生的信息
}
void staOfStudent()//个人统计
{    int i,j;
     float ave[N]={0.0};//存放每位学生的平均成绩
     float sum[N]={0.0};//存放每位学生的总成绩
     for(i=0;i<M;i++)//每个学生
     {    for(j=0;j<N;j++)
               sum[i]+=stu[i].score[j];
          ave[i]=sum[i]/N;
          printf("学生%s 的总分为：%.2f,平均分为：%.2f\n",stu[i].name,sum[i],ave[i]);
     }
}
void staOfClass()//班级统计
{    int i,j;
     float ave[N]={0.0};
     float sum[N]={0.0};
     for(i=0;i<N;i++)//每门课程
     {    for(j=0;j<M;j++)
               sum[i]+=stu[j].score[i];
          ave[i]=sum[i]/M;
          printf("该班级第%d 门课程的平均分为：%.2f\n",i+1,ave[i]);
     }
}
void addStudent()//增加一名学生的信息
{    int k;
     printf("\n 请输入第%d 名学生的学号、姓名:\n",curStuCount+1);
```

```
    scanf("%d%s",&stu[curStuCount].num,stu[curStuCount].name);
    for(k=0;k<N;k++)
    {   printf("第%d 门课程成绩:",k+1);
        scanf("%f",&stu[curStuCount].score[k]);
    }
    curStuCount++;//计数器加 1
}
void outStuInformation(char str[L])//输出一名学生的信息
{   int i,j;
    for(i=0;i<M;i++)
        if(strcmp(str,stu[i].name)==0)
        {   printf("该学生信息如下：\n");
            printf("学号：%3d 姓名：%6s\n",stu[i].num,stu[i].name);
            printf("成绩:");
            for(j=0;j<N;j++)
                printf("%.2f   ",stu[i].score[j]);
            printf("\n");
            break;
        }
    if(i>=M)
        printf("没有找到该学生的信息。\n");
}
void search()//查找某个学生
{   char tmpStr[L];
    printf("请输入要查找的学生姓名:");
    scanf("%s",tmpStr);
    outStuInformation(tmpStr);//调用 outStuInformation 函数
}
```

8.12 学生成绩表包含如下信息：学号、姓名、考试科目、平时作业成绩、期中成绩、期末成绩、课程成绩。建立描述一个学生一科成绩的数据类型。若

课程成绩=平时作业成绩*10%+期中成绩*30%+期末成绩*60%

且全部学生的全部科目的考试成绩都以这种形式保存，分别编出实现如下功能的函数：

1）输入平时作业成绩、期中成绩、期末成绩，计算课程成绩；

2）输出某个学生的成绩单，该成绩单包括该学生所有考试科目的课程成绩；

3）输出某课程的成绩单，该成绩单包括所有参加本课程考试的学生的成绩；

4）输出某课程的不及格学生名单及其成绩；

5）输出平均成绩统计表。该表把所有学生按平均成绩按递减顺序输出。

```
#include<stdio.h>
#include<string.h>

#define L 20     // 姓名长度
#define M 1000

//定义结构体类型
typedef struct student{
    int num;                    //学号
    char StuName[L];            //姓名
    char CouName[L];            //科目名称
    float ExScore;              //平时作业成绩
    float MiScore;              //期中成绩
    float EnScore;              //期末成绩
    float CoScore;              //课程成绩
}stuType;

//函数为添加成绩
//参数分别为：成绩表数组，当前记录数，要加入记录数
//函数会带回添加之后记录数到 CurrEls,自动计算课程成绩
void AddCouSco(stuType StuCouSco[],int *CurrEls,int AddEls);

//函数功能为输出某学生所有课程成绩
//参数分别为：成绩表数组，要输出成绩的学生，当前记录数
void OutStuSco(stuType StuCouSco[],char StuName[],int CurrEls);

//函数功能为输出某门课程所有学生的成绩
//参数分别为：成绩表数组，要输出的课程，当前记录数
void OutCouSco(stuType StuCouSco[],char CouName[],int CurrEls);

//函数功能为输出某门课程不及格学生名单及成绩
//参数分别为：成绩表数组，要输出的课程，当前记录数
void OutCouFailSco(stuType StuCouSco[],char CouName[],int CurrEls);

//函数功能为输出递减排序的所有学生平均成绩单
//参数分别为：成绩表数组，当前记录数
//每个学生可能没有参加全部考试，平均成绩按（总分/实际考试科目）计算
void OutAllStuAveSco(stuType StuCouSco[],int CurrEls);

//主函数
void main()
{
    stuType stu[M]=
    {
```

```
        {2,"lihe","yuwen",90,90,90,90},
        {4,"wangqinghu","yuwen",80,80,80,80},
        {3,"lijunfu","yuwen",98,98,98,98},
        {7,"wangtingwu","yuwen",50,50,50,50},

        {2,"lihe","shuxue",90,90,90,90},
        {4,"wangqinghu","shuxue",80,80,80,80},
        {3,"lijunfu","shuxue",96,96,96,96},
        {7,"wangtingwu","shuxue",60,60,60,60},

        {2,"lihe","yingyu",90,90,90,90},
        {4,"wangqinghu","yingyu",80,80,80,80},
        {3,"lijunfu","yingyu",94,94,94,94},
        {7,"wangtingwu","yingyu",70,70,70,70},

        {2,"lihe","lizong",90,90,90,90},
        {4,"wangqinghu","lizong",80,80,80,80},
        {3,"lijunfu","lizong",92,92,92,92},
        {7,"wangtingwu","lizong",40,40,40,40},
};

int CurrentEls=16;//当前记录数

printf("\n 当前记录数：%d\n",CurrentEls);

int AddEls;
printf("\n 输入添加记录数:     ");
scanf("%d",&AddEls);
AddCouSco(stu,&CurrentEls,AddEls);

char TempStuName[L];
printf("\n 输入需要输出成绩单学生的姓名：");
scanf("%s",TempStuName);
OutStuSco(stu,TempStuName,CurrentEls);

char TempCouName[L];
printf("\n 输入需要输出成绩单的课程名：");
scanf("%s",TempCouName);
OutCouSco(stu,TempCouName,CurrentEls);

char TempCouName2[L];
printf("\n 输入需要统计不及格成绩的课程名：");
scanf("%s",TempCouName2);
OutCouFailSco(stu,TempCouName2,CurrentEls);
```

```
    OutAllStuAveSco(stu,CurrentEls);


}
void AddCouSco(stuType StuCouSco[],int *CurrEls,int AddEls)
{
    printf("\n 当前记录数为：%d,添加记录数为：%d\n",*CurrEls,AddEls);
    int i;
    for(i=0;i<AddEls;i++)
    {
        printf("序号：%d\n",i+1);
        printf("学号：");
        scanf("%d",&StuCouSco[*CurrEls+i].num);
        printf("姓名：");
        scanf("%s",StuCouSco[*CurrEls+i].StuName);
        printf("科目：");
        scanf("%s",StuCouSco[*CurrEls+i].CouName);
        printf("平时作业：");
        scanf("%f",&StuCouSco[*CurrEls+i].ExScore);
        printf("其中成绩：");
        scanf("%f",&StuCouSco[*CurrEls+i].MiScore);
        printf("期末成绩：");
        scanf("%f",&StuCouSco[*CurrEls+i].EnScore);
        StuCouSco[*CurrEls+i].CoScore=StuCouSco[*CurrEls+i].ExScore*0.1
                                      +StuCouSco[*CurrEls+i].MiScore*0.3
                                      +StuCouSco[*CurrEls+i].EnScore*0.6;
        printf("\n");
    }
    *CurrEls=*CurrEls+AddEls;
    printf("添加%d 条记录成功,当前记录数为%d\n",AddEls,*CurrEls);
    printf("\n");
}
void OutStuSco(stuType StuCouSco[],char StuName[],int CurrEls)
{
    printf("\n 输出%s 的成绩单：\n",StuName);
    int i;
    bool isNotheStu=true;
    for(i=0;i<CurrEls;i++)
    {
        if(strcmp(StuCouSco[i].StuName,StuName)==0)
        {
            if(isNotheStu)
            {
```

```
                    isNotheStu=false;
                    printf("学号：%d\n",StuCouSco[i].num);
                    printf("姓名：%s\n",StuCouSco[i].StuName);
                }
                printf("%s    ：%f\n",StuCouSco[i].CouName,StuCouSco[i].CoScore);
            }
        }
        if(isNotheStu)
        {
            printf("未找到学生%s 的成绩。\n",StuName);
        }
}
void OutCouSco(stuType StuCouSco[],char CouName[],int CurrEls)
{
        printf("\n 输出%s 课考试的成绩单：\n",CouName);
        int i;
        int j=0;
        bool isNotheCou=true;
        for(i=0;i<CurrEls;i++)
        {
            if(strcmp(StuCouSco[i].CouName,CouName)==0)
            {
                if(isNotheCou)
                {
                    isNotheCou=false;
                    printf("        序 号                学 号                姓 名        %s
\n",CouName);
                }

        printf("%9d%14d%14s%10.2f\n",++j,StuCouSco[i].num,StuCouSco[i].StuName,StuCouSco[
i].CoScore);
            }
        }
        if(isNotheCou)
        {
            printf("没有课程%s 的信息。\n",CouName);
        }
}
void OutCouFailSco(stuType StuCouSco[],char CouName[],int CurrEls)
{
        printf("\n 输出%s 课不及格学生名单及成绩：\n",CouName);
        int i;
        int j=0;
        bool isNotheCou=true;
```

```
    bool isNoFailStu=true;
    for(i=0;i<CurrEls;i++)
    {
        if(strcmp(StuCouSco[i].CouName,CouName)==0)
        {
            if(isNotheCou)
            {
                isNotheCou=false;
                printf("        序 号                学 号                姓 名        %s
\n",CouName);
            }
            if(StuCouSco[i].CoScore<60.0)
            {
                isNoFailStu=false;

printf("%9d%14d%14s%10.2f\n",++j,StuCouSco[i].num,StuCouSco[i].StuName,StuCouSco[i].Co
Score);
            }
        }
    }
    if(isNotheCou||isNoFailStu)
    {
        printf("%s 课程无不及格学生或无此课程。\n",CouName);
    }
}
void OutAllStuAveSco(stuType StuCouSco[],int CurrEls)
{
    char TempStuName[M][L];
    float StuAve[M];
    int StuNum=0;
    int i;
    int j;

    for(i=0;i<CurrEls;i++)
    {
        bool isNotExist=true;
        for(j=0;j<StuNum;j++)
        {
            if(strcmp(StuCouSco[i].StuName,StuCouSco[j].StuName)==0)
            {
                isNotExist=false;
                break;
            }
        }
```

```
        if(isNotExist)
        {
            strcpy(TempStuName[StuNum],StuCouSco[i].StuName);
            StuNum++;
        }
    }
    for(i=0;i<StuNum;i++)
    {
        float TempAvae=0.0;
        int TempCouNum=0;
        for(j=0;j<CurrEls;j++)
        {
            if(strcmp(StuCouSco[j].StuName,TempStuName[i])==0)
            {
                TempAvae+=StuCouSco[j].CoScore;
                TempCouNum++;
            }
        }
        if(TempCouNum>0)
        {
            TempAvae=TempAvae/TempCouNum;
            StuAve[i]=TempAvae;
        }
    }

    printf("\n 输出平均成绩统计表：\n");

    for(i=0;i<(StuNum-1);i++)
    {
        for(j=i+1;j<StuNum;j++)
        {
            if(StuAve[j]>StuAve[i])
            {
                float TempSco;
                char    TempName[L];

                TempSco=StuAve[i];
                strcpy(TempName,TempStuName[i]);
                StuAve[i]=StuAve[j];
                strcpy(TempStuName[i],TempStuName[j]);
                StuAve[j]=TempSco;
                strcpy(TempStuName[j],TempName);
            }
        }
```

```
    }

    printf("        序号                姓名        平均成绩\n");
    for(i=0;i<StuNum;i++)
    {
        printf("%9d%14s%12.2f\n",i+1,TempStuName[i],StuAve[i]);
    }

    if(StuNum<=0)
    {
        printf("目前没有成绩信息。\n");
    }
}
```

第九章

9.1（略）

9.3（略）

9.6 编写函数，分别用指针传递参数，实现两个字符串变量值的交换

```
#include "stdio.h"
void swap (char * xx,char * yy)
{
	char temp;
	for(;(*xx!='\0')&&(*yy!='\0');++xx,++yy)
	{
		temp=*xx;
		*xx=*yy;
		*yy=temp;
	}
	if((*xx=='\0')&&(*yy!='\0'))
	{

		*xx=*yy;
		*yy='\0';
		++yy;
		++xx;

		for( ;*yy!='\0';++yy,++xx)
		{
			*xx=*yy;
		}
		*xx='\0';
	}
	else if((*xx!='\0')&&(*yy=='\0'))
	{

		*yy=*xx;
		*xx='\0';
		++xx;
		++yy;

		for( ;*xx!='\0';++xx,++yy)
		{
			*yy=*xx;
		}
```

```
            *yy='\0';
        }
}
void main(void)
{
        char a[1000],b[1000];
        printf("输入第一个串\n");
        scanf("%s",a);
        printf("输入第二个串\n");
        scanf("%s",b);
        swap(a,b);
        printf("\n 交换后第一个串:%s\n",a);
        printf("交换后第二个串:%s\n",b);
}
```

9.9 **编写函数，使得仅通过此函数，便可以进行两个整数的加、减、乘、除运算。**

```
#include "stdio.h"
enum OpSymbol{add,sub,mul,div};
float ArithMetic(int x,int y,OpSymbol os)
{
        switch(os)
        {
        case 0:
            return float(x+y);
        case 1:
            return float(x-y);
        case 2:
            return float(x*y);
        case 3:
            return float(x)/float(y);
        }
}

void main(void)
{
        int a,b;
        printf("Input Two Data:\n");
        scanf("%d%d",&a,&b);
        printf("a+b=%f\n",ArithMetic(a,b,add));
        printf("a-b=%f\n",ArithMetic(a,b,sub));
        printf("a*b=%f\n",ArithMetic(a,b,mul));
        printf("a/b=%f\n",ArithMetic(a,b,div));
}
```

第十章

10.1 用递归计算斐波纳契序列第 n 项。该序列可以表示成

$$f(n)=\begin{cases}1 & \text{当n = 1}\\ 1 & \text{当n = 2}\\ f(n-1)+f(n-2) & \text{当n > 2}\end{cases}$$

```
#include<stdio.h>

int f(int n)
{
    if(n==1)
        return 1;
    if(n==2)
        return 1;
    if(n>2)
        return(f(n-1)+f(n-2));
}
void main()
{
    int n;
    printf("输入 n:");
    scanf("%d",&n);
    printf("f(%d)=%d.\n",n,f(n));
}
```

10.3 分别用递归和递推方法编写出计算 Hermite 多项式的函数。Hermite 多项式定义为

$$H_n(x)=\begin{cases}1, n=0\\ 2x, n=1\\ 2xH_{n-1}(x)-2(n-1)H_{n-2}(x), n>1\end{cases}$$

参考答案

```
#include "stdio.h"
float HDG(float x,int n)//函数定义
{
    if(n==0)        return 1;
    if(n==1)        return 2*x;
    return(2*x*HDG(x,n-1)-2*(n-1)*HDG(x,n-2));
}

float HDT(float x,int n)
{
```

```
    float h0,h1,h2;
    h0=1;
    h1=2*x;
    if(n==0)        return h0;
    if(n==1)        return h1;
    do
    {
        h2=2*x*h1-2*(n-1)*h0;
        h0=h1;
        h1=h2;
        n--;
    }while(n>1);
    return h2;
}


void main()
{

    printf("\nTest DG:x=%f,n=%d,Hermite=%f\n",2.0,5,HDG(2.0,5));
    printf("\nTest DT:x=%f,n=%d,Hermite=%f\n\n",2.0,5,HDT(2.0,5));
}
```

10.4 编一个计算 Ackerman 函数的递归函数。然后加一个输出函数参数 m 、n 的输出语句作为函数体的第一个语句，并写一个程序输入 m 、n 后调用该函数，对于 m=3 ， n=2 的情况执行该程序，观察递归调用层次及状况。Ackerman 函数定义为

$$Ack(m,n)=\begin{cases} n+1 & \text{当 m = 0} \\ \text{Ack(m - 1,1)} & \text{当 n = 0} \\ \text{Ack(m - 1, Ack(m, n - 1))} & \text{当 m > 0 且 n > 0} \end{cases}$$

```
#include<stdio.h>
int Ack(int m,int n)
{
    printf("m=%d,n=%d\n",m,n);
    int r,g;
    if(m==0)
        return n+1;
    else if(n==0)
        return Ack(m-1,1);
    else
    {
        return Ack(m-1,Ack(m,n-1));
    }
```

```
}
void main()
{
    int m,n;
    printf("输入 m 和 n:");
    scanf("%d%d",&m,&n);
    printf("Ack(%d,%d)=%d.\n",m,n,Ack(m,n));
}
```

10.5 编函数，用递归方法求 n 个元素数组 a 的最大值。

```
#include "stdafx.h"
#include "stdio.h"

int max_num( int *a ,int num )
{
    if ( num == 1)
        return *a;
    if ( *a > * (a + 1) )
    {
        int Temp;
        Temp=*a;
        *a = *(a + 1);
        *( a+1 ) = Temp;
    }
    return max_num( a+1 , num-1);
};

int main(void)
{
    int a[] = {12,36,20,65,32,65,-45,-21,5235,205,-65,202,10,2};

    printf("max num    %d\n",max_num(a,14));
    return 0;
}
```

10.6 设有数组声明 int a[100]， 试编一个递归函数，求 a 的反序数组并仍保存在 a 中， 即 a[1]与 a[100]交换， a[2]与 a[99]交换,a[3]与 a[98]交换， … ,a[50]与 a[51]交换。

```
#include "stdafx.h"
#include "stdio.h"

void fun(int *a,int len);
```

```
void main()
{
	int i;
	int a[100];
	for(i=0;i<100;i++)
		a[i]=i;
	fun(a,100);
	for(i=0;i<100;i++)
		printf("%d ", a[i]);
}
//支持任意长度
void fun(int a[],int len)
{
	int tmp;
	if(len<=1)
		return;
	else
	{
		tmp = a[0];
		a[0] = a[len-1];
		a[len-1] = tmp;
		fun(a+1, len-2);
	}
}
```

10.12 编写程序，计算下式直至两次计算结果之差的绝对值小于 10^{-5}.

$y = \arctan\cos\arctan\cos...\arctan\cos(\pi/6)$

参考答案

```
#include<stdio.h>
#include<math.h>

#define EPS 1e-5

double fun(double data)
{
    double m=atan(cos(data));
    if(fabs(m-data)<EPS)
        return m;
    else
        return fun(m);
}
void main()
{
```

```
    double data=atan(cos(3.1415926/6));
    printf("result=%f\n",fun(data));
}
```

10.17 Lisp 语言由 S 表达式组成，S 表达式定义如下：

1. 任意字母 S 是 S 表达式；

2. 若 u、v 分别都是 S 表达式，则（u,v）也是 S 表达式。

编函数，判断给定的字符串 L 是否 S 表达式。

```
#include<stdio.h>
#include "stdio.h"
bool    se( char **str )
{
    char **st=str;
    bool    u,v;
    if((('A'<= **st)&&(**st<='Z'))||(('a'<= **st)&&(**st<='z')))
    {
        (*st)++;
        return true;
    }
    if(**st=='(')
    {
        (*st)++;
        v=se(st);
        if(**st!=',')
        {
            return false;
        }
        else
        {
            (*st)++;
            u=se(st);
            if(**st!= ')')
            {
                return false;
            }
            else
            {
                (*st)++;
                return u&&v;
            }
        }
    }
```

```
    return false;
}
void main()
{
    char*st="(((((A,B),C),f),((s,w),a)),((a,b),c))";
    char*p=st;
    if(se(&p)&&*p=='\0')
        printf("是 S 表达式\n");
    else
        printf("不是 S 表达式\n");
}
```

第十一章

11.1 编写程序统计文本文件中字符的个数。

参考答案：

```
#include<stdio.h>
#include<stdlib.h>
void main()
{
    FILE* fp;
    int count=0;//计数器
    char s;//存储取出的字符
    char filename[100];//文件的名字

    printf("请输入文件名(含扩展名):\n");
    scanf("%s",filename);/*输入文件名，可输入路径；缺省路径为当前源程序的路径*/

    if((fp=fopen(filename,"r"))==NULL)
    {   printf("Can not open file %s\n",filename);
        exit(0);
    }
    s=getc(fp);//从文件读一个字符
    while(s!=EOF)
    {   printf("%c",s);//输出该字符
        count++;//计数器加 1(包括换行符)
        s=fgetc(fp);
    }
    printf("\n 该文本文件中共有%d 个字符.\n",count);
    fclose(fp);//关闭文件
}
```

11.6 编写程序，把给定整数文件中所有大于某值的数复制到另一个给定文件中去。

参考答案：

```
#include<stdio.h>
#include<stdlib.h>

//函数声明
void writeToSourceFile();//初始化 soureFile.txt
void writeToDesFile(int data, FILE* fpSource,FILE* fpDes);//将大于 m 的数从 sourFile.txt 写入
desFile.txt
void readFile(FILE* fp);//读出 fp 所指文件中的所有数据

//主函数
void main()
```

```
{    int m;//指定整数
     FILE *fpSource,*fpDes;//文件指针，分别指向源文件和目标文件
     /*初始化 sourceFile.txt, 当 sourceFile.txt 无内容或需要改写时，可调用此函数写入数据;若已经有数据，则不必调用*/
     writeToSourceFile();//函数调用
     //打开文件
     if((fpSource=fopen("sourceFile.txt","r"))==NULL)
     {    printf("无法打开文件:%s\n","sourceFile.txt");
          exit(0);
     }
     if((fpDes=fopen("desFile.txt","w+"))==NULL)//写完后又读出，因此以可读可写方式打开
     {    printf("无法打开文件:%s\n","desFile.txt");
          exit(0);
     }

     printf("请输入一个整数:\n");
     scanf("%d",&m);

     writeToDesFile(m, fpSource,fpDes);//函数调用

     printf("源文件内的数据为:\n");
     readFile(fpSource);//函数调用

     printf("目标文件内的数据为:\n");
     readFile(fpDes);//函数调用

     fclose(fpSource);//关闭文件
     fclose(fpDes);
}
void writeToSourceFile()
{    FILE* fp;
     int i,tmp,count;

     if((fp=fopen("sourceFile.txt","w"))==NULL)
     {    printf("无法打开文件%s\n","sourceFile.txt");
          exit(0);
     }

     printf("请输入要输入整数的个数:");
     scanf("%d",&count);
     printf("请输入%d 个整数:",count);
     for(i=0;i<count;i++)
     {    scanf("%d",&tmp);
          fprintf(fp,"%6d",tmp);
```

```
    }

    fclose(fp);
}
void writeToDesFile(int data, FILE* fpSource,FILE* fpDes)
{   int tmp;
    printf("\n 正在将 sourceFile.txt 中大于%d 的数据写入到 desFile.txt 中.....\n\n",data);

    rewind(fpSource);//使 fpSource 重新定位到文件首

    while(!feof(fpSource))
    {   fscanf(fpSource,"%6d",&tmp);
        if(tmp>data)
            fprintf(fpDes,"%6d",tmp);
    }
}
void readFile(FILE* fp)
{   int tmp;
    rewind(fp);//使 fp 重新定位到文件首
    while(!feof(fp))
    {   fscanf(fp,"%6d",&tmp);
        printf("%6d",tmp);
    }
    printf("\n");
}
```

11.7 分解整数文件 f 到 g1、g2 。g1 保存所有素数；g2 保存其它数。g1、g2 每行五个数。

```
#include<stdio.h>
#include<stdlib.h>
#include"math.h"

//函数声明
void WriteIntFile(char *IntFile,int IntNum);
bool IsPrimeNumber(int Number);
void OutFile(char *FileName);
void SplitFile(char *pSourceFile,char*PrimeFile,char *OtherFile);
void main()
{
    char    SourceFileName[200];
    char    PrimeFileName[200];
    char    OtherFileName[200];
    int IntNum;

    printf("\n 输入要创建的整数文件名：");
    scanf("%s",SourceFileName);
```

```
    printf("输入整数个数：");
    scanf("%d",&IntNum);
    printf("输入%d 个整数：\n",IntNum);
    WriteIntFile(SourceFileName,IntNum);

    printf("\n 输入素数文件名：");
    scanf("%s",PrimeFileName);
    printf("\n 输入其它整数文件名：");
    scanf("%s",OtherFileName);
    SplitFile(SourceFileName,PrimeFileName,OtherFileName);

    printf("原文件内容：\n");
    OutFile(SourceFileName);
    printf("素数文件内容:\n");
    OutFile(PrimeFileName);
    printf("其它整数文件内容；\n");
    OutFile(OtherFileName);
}
void WriteIntFile(char *IntFile,int IntNum)
{
    FILE *fpFile;
    int TempInt;
    if((fpFile=fopen(IntFile,"w"))==NULL)
    {
        printf("\n 文件打开或创建文件失败: %s\n",IntFile);
        exit(0);
    }
    for(int i=0;i<IntNum;i++)
    {
        scanf("%d",&TempInt);
        fprintf(fpFile,"%6d",TempInt);
    }
    fclose(fpFile);
}
bool IsPrimeNumber(int Number)
{
    int i;
    if(Number<2)
        return false;
    if(Number==2)
        return true;
    for(i=2;i<=sqrt(Number);i++)
    {
        if(0==(Number%i))
```

```
                return false;
        }
        return true;
}
void OutFile(char *FileName)
{
        FILE *fp;
        int Temp;
        int Count=0;
        if((fp=fopen(FileName,"r"))==NULL)
        {
                printf("无法打开文件:%s\n",FileName);
                exit(0);
        }
//      rewind(fp);//使 fp 重新定位到文件首
        while(!feof(fp))
        {
                fscanf(fp,"%d",&Temp);
                printf("%6d",Temp);
                Count++;
                if(Count%5==0)
                {
                        printf("\n");
                }
        }
        printf("\n\n");
        fclose(fp);
}

void SplitFile(char *pSourceFile,char*PrimeFile,char *OtherFile)
{
        FILE *fpS;
        FILE *fpP;
        FILE *fpO;
        int Count1=0;
        int Count2=0;

        int Temp;
        if((fpS=fopen(pSourceFile,"r"))==NULL)
        {       printf("无法打开文件:%s\n",pSourceFile);
                exit(0);
        }
        if((fpP=fopen(PrimeFile,"w"))==NULL)
        {       printf("无法打开文件:%s\n",PrimeFile);
```

```
        exit(0);
    }
    if((fpO=fopen(OtherFile,"w"))==NULL)
    {   printf("无法打开文件:%s\n",OtherFile);
        exit(0);
    }
    printf("\n 正在将%s 文件分解.....\n\n",pSourceFile);
    while(!feof(fpS))
    {
        fscanf(fpS,"%d",&Temp);
        if(IsPrimeNumber(Temp))
        {
            fprintf(fpP,"%6d",Temp);
            Count1++;
            if(Count1%5==0)
            {
                fprintf(fpP,"\n");
            }
        }
        else
        {
            fprintf(fpO,"%6d",Temp);
            Count2++;
            if(Count2%5==0)
            {
                fprintf(fpO,"\n");
            }
        }
    }
    printf("\n%s 文件分解结束.....\n\n",pSourceFile);
    fclose(fpS);
    fclose(fpP);
    fclose(fpO);
}
```

第十二章

12.8 若一个整数 a 满足条件：a^2 的尾数等于 a,则称 a 为自守数，例如：
25^2＝625，76^2＝5776，9376^2＝87909376
都是自守数。编写程序，求 10000 以内的所有自守数。
参考答案：

```
#include<stdio.h>

//函数原型
bool check(int data);//判断是否是自守数

void main()
{
	int i;
	for(i=0;i<=10000;i++)
		if(check(i))//函数调用
			printf("%d\n",i);
}

bool check(int data)//函数定义
{
	int num=1;
	int tmp=data;
	while(tmp!=0)//求 num
	{
		tmp/=10;
		num*=10;
	}
	if(data == data*data%num)
		return true;
	return false;
}
```

P141_12（习题集）. 猜数游戏，编写程序，不断生成 1~100 的随机数，游戏者猜该数是多少；程序反馈是大、小或正好。若用户不回答，直接按回车键，程序运行结束。

```
#include<stdio.h>
#include "string.h"
#include "stdlib.h"
void main()
{
	int TempRand;
	int InPutNum;
```

```
    printf("++++++++++++++++++++++++++猜数游戏++++++++++++++++++++++++++\n\n");
    while(true)
    {
        TempRand=rand()%100;
        if(TempRand==0)
        {
            TempRand+=1;
        }
        printf("\n 数已生成，你猜是多少(1~100)：");
        scanf("%d",&InPutNum);
        if(InPutNum>100||InPutNum<1)
        {
            printf("数据超出范围，游戏结束\n");
            return;
        }
        if(InPutNum>TempRand)
        {
            printf("大了\n");
        }
        if(InPutNum==TempRand)
        {
            printf("正好\n");
        }
        if(InPutNum<TempRand)
        {
            printf("小了\n");
        }
        if(InPutNum!=TempRand)
        {
            printf("想知道数是多少吗？(yes/no)");
            char TempRe[10];
            scanf("%s",TempRe);
            if(strcmp("yes",TempRe)==0)
            {
                printf("数是：%d\n\n",TempRand);
            }
            else
            {
                printf("\n\n");
            }
        }
    }
}
```

第十三章

13.2 编写一个函数，把给定的任意单向环形链改成逆向的单向环形链。

参考答案：

```
#include "stdio.h"
#include "malloc.h"
typedef struct item{
	int key;
	struct item * next;
}itemType;

itemType * initial(int n){//构造链表
	itemType* p,*base,*q;
	base=(itemType*)malloc(sizeof(itemType));
	base->key=1;
	q=base;
	for(int i=2;i<=n;i++){
		p=(itemType*)malloc(sizeof(itemType));
		p->key=i;
		p->next=NULL;
		q->next=p;
		q=q->next;
	}

	q->next=base;//环链

	return base;
}
void print(itemType * base){
	itemType * tmp=base;
	do
	{	printf("%d\t",tmp->key);
		tmp=tmp->next;
	}while(tmp!=base);
}
itemType * reverse(itemType* base)//base 为原环链的表头指针
{
	itemType* p0,*p,*q,*rs=NULL,*r0;
	/*申请哨兵变量 r0,用 r0  和 p0 标识要插入的位置
	  构造初始化的 r0、p0 构成的链,而真正开始插入的元素由 p 标识*/
	r0=(itemType*)malloc(sizeof(itemType));
	r0->next=base;
```

```
    p0=base;
    p=base->next;
    base->next=NULL;
    /*将 p 插到 r0 和 p0 之间,修改 p,p0 为下次操作准备*/
    while(p!=base){
        q=p;
        p=p->next;
        q->next=p0;
        r0->next=q;
        p0=q;
    }
    /*释放哨兵变量,返回结果*/
    rs=r0->next;
    free(r0);

    base->next=rs;//环链

    return rs; //rs 为逆向环链表的表头指针
    //return base;
}
void main(){
    int m=9;
    itemType *begin,*re;

    begin=initial(m);

    printf("原环形链表:\n");
    print(begin);
    printf("\n");

    re=reverse(begin);

    printf("逆向环形链表:\n");
    print(re);
    printf("\n");
}
```

13.4 编写函数，删除单向链表中所有值为素数的结点。

参考答案：

```
#include <stdio.h>
#include <malloc.h>
#include<math.h>

typedef struct item{
    int key;
```

```
    struct item * next;
}itemType;

itemType * initial(int n){//构造链表
    itemType* p,*base,*q;
    base=(itemType*)malloc(sizeof(itemType));
    base->key=2;
    q=base;
    for(int i=3;i<=n;i++){
        p=(itemType*)malloc(sizeof(itemType));
        p->key=i;
        p->next=NULL;
        q->next=p;
        q=q->next;
    }
    return base;
}
bool check(int data)
{
    int i,temp=(int)sqrt(data);
    for(i=2;i<=temp;i++)
        if(data%i==0)
            break;//有合数跳出本次循环
    if (i>temp)
        return true;
    return false;
}
void print(itemType * base)
{
    itemType * tmp=base;
    while(tmp!=NULL){
        printf("%d\t",tmp->key);
        tmp=tmp->next;
    }
}
itemType* del(itemType* base)//删除所有素数结点
{
    itemType *tmp=base;
    itemType *q;//指向待删除结点
    itemType *p0=base;//指向前驱结点
    while(tmp!=NULL)
    {
        if(check(tmp->key))//删除素数结点
        {
```

```
                q=tmp;
                tmp=tmp->next;
                p0->next=tmp;
                if(q==base)//要删除的是表头指针
                    base=tmp;
                free(q);
            }
            else
            {
                p0=tmp;
                tmp=tmp->next;
            }
        }
        return base;
}
void main(){
        int m=100;
        itemType *link1,*link2;
        link1=initial(m);

        printf("初始链表：");
        print(link1);
        printf("\n");

        link2=del(link1);

        printf("删除后：");
        print(link2);
        printf("\n");

}
```

13.16 已知一个以单向链表为存储结构的线性表 A，表元素为字符型。编写一个函数 p 实现把表中的数字、字母、其他符号分离，形成 3 个链表 N(数字)、C(字符)、O(其他)。要求分离后的链表中的字符顺序与原链表 A 中的顺序一致。

参考答案：

```
#include <stdio.h>
#include <malloc.h>

typedef struct item
{
        char key;
        struct item * next;
}itemType;
```

```
typedef itemType* pType;

pType linkN,linkC,linkO;//各表头指针

pType initial()
{//构造链表 A
    pType p,q,base;
    int count,i;
    printf("请输入链表 A 的结点个数( >0 )：");
    scanf("%d",&count);
    while(count<1)
    {
        printf("请输入正确的结点个数( >0 )：");
        scanf("%d",&count);
    }
    getchar();

    printf("请输入第 1 个字符：");
    base=(itemType*)malloc(sizeof(itemType));
    scanf("%c",&base->key);
    base->next=NULL;

    q=base;
    for(i=1;i<count;i++)
    {
        printf("请输入第%d 个字符：",i+1);
        getchar();
        p=(itemType*)malloc(sizeof(itemType));
        scanf("%c",&p->key);
        p->next=NULL;
        q->next=p;
        q=q->next;
    }
    p=NULL;
    return base;
}

void print(pType base)//输出链表
{
    pType tmp=base;
    while(tmp!=NULL)
    {
        printf("%c ",tmp->key);
        tmp=tmp->next;
```

```
    }
}
void fun_p(pType linkA)//分离各个字符，创建新的链表
{
    pType tmp=linkA;
    pType p;//为新结点申请空间
    pType pN,pC,pO;//分别标记链表 N,C,0 当前位置
    pN=pC=pO=NULL;

    while(tmp!=NULL)//遍历链表 A
    {
        if(tmp->key>='0'&&tmp->key<='9')//数字
        {
            if(linkN==NULL)//空表，需要创建新链表
            {
                linkN=(itemType*)malloc(sizeof(itemType));
                linkN->key=tmp->key;
                linkN->next=NULL;
                pN=linkN;
            }
            else//非空表，直接加在表尾
            {
                p=(itemType*)malloc(sizeof(itemType));
                p->key=tmp->key;
                p->next=NULL;
                pN->next=p;
                pN=pN->next;
            }
        }
        else if((tmp->key>='a'&&tmp->key<='z')||(tmp->key>'A'&&tmp->key<='Z'))//字母
        {
            if(linkC==NULL)//空表，需要创建新链表
            {
                linkC=(itemType*)malloc(sizeof(itemType));
                linkC->key=tmp->key;
                linkC->next=NULL;
                pC=linkC;
            }
            else//非空表，直接加在表尾
            {
                p=(itemType*)malloc(sizeof(itemType));
                p->key=tmp->key;
                p->next=NULL;
                pC->next=p;
```

```
                pC=pC->next;
            }
        }
        else//其他字符
        {
            if(linkO==NULL)//空表，需要创建新链表
            {
                linkO=(itemType*)malloc(sizeof(itemType));
                linkO->key=tmp->key;
                linkO->next=NULL;
                pO=linkO;
            }
            else//非空表，直接加在表尾
            {
                p=(itemType*)malloc(sizeof(itemType));
                p->key=tmp->key;
                p->next=NULL;
                pO->next=p;
                pO=pO->next;
            }
        }
        tmp=tmp->next;
    }
}

void main()//主函数
{
    pType linkA;
    linkA=initial();
    linkN=linkC=linkO=NULL;

    printf("linkA： ");
    print(linkA);
    printf("\n");

    fun_p(linkA);//函数调用

    //输出各链表
    printf("linkN： ");
    print(linkN);
    printf("\n");

    printf("linkC： ");
    print(linkC);
```

```
    printf("\n");

    printf("linkO： ");
    print(linkO);
    printf("\n");

}
```

P129_11（习题集）. 写子程序，删除有序单向链表中的相同元素。

```
#include "stdlib.h"

typedef struct SNode{
    int Data;
    struct SNode* Next;
}Node;
//删除相同元素
void DelLTabSameEle(Node *head)//head 是头结点
{
    Node * p = head->Next;//p 指向第一个元素节点
    while(p->Next!=NULL)//p 不是最后一个元素节点
    {
        if(p->Data==p->Next->Data)//p 和下一个节点数据相等，则删除下一节点
        {
            Node *temp=p->Next;
            p->Next=temp->Next;
            free(temp);
        }
        else//不相等则 p 指向下一节点
        {
            p=p->Next;
        }
    }
}

//构造 n 个节点有头的有序链表
Node *InitialLTab(int n)
{
    Node *p,*base,*q,*head;
    head=(Node*)malloc(sizeof(Node));

    base=(Node*)malloc(sizeof(Node));
    base->Data=1;
    head->Next=base;
    q=base;
    for(int i=2;i<=n;i++)
```

```
    {
        p=(Node*)malloc(sizeof(Node));
        p->Data=i;
        p->Next=NULL;
        q->Next=p;
        q=q->Next;
    }
    return head;
}
//显示链表
void PrintLTab(Node *head)
{
    int TempNum=0;
    Node *tmp=head->Next;
    while(tmp!=NULL)
    {
        printf("%5d",tmp->Data);
        tmp=tmp->Next;
        TempNum++;
        if(TempNum%10==0)
            printf("\n");
    }
    printf("\n 输出结束\n");
}
//插入一个节点并保持有序
void InsertANode(Node *head, Node *ins)//head  头指针, Ins 要插入结点指针
{
    Node *pre=head;
    Node *p=head->Next;
    while(p!=NULL&&p->Data<ins->Data)
    {
        pre=pre->Next;
        p=pre->Next;
    }
    ins->Next=pre->Next;
    pre->Next=ins;
}

void main()
{
    Node *pHead;
    int NodeNum;
    printf("请输入初始节点数：");
    scanf("%d",&NodeNum);
```

```
    pHead=InitialLTab(NodeNum);
    PrintLTab(pHead);

    Node *TempNode;

    TempNode=(Node*)malloc(sizeof(Node));
    TempNode->Data=5;
    InsertANode(pHead,TempNode);

    TempNode=(Node*)malloc(sizeof(Node));
    TempNode->Data=0;
    InsertANode(pHead,TempNode);

    TempNode=(Node*)malloc(sizeof(Node));
    TempNode->Data=5;
    InsertANode(pHead,TempNode);

    TempNode=(Node*)malloc(sizeof(Node));
    TempNode->Data=8;
    InsertANode(pHead,TempNode);

    TempNode=(Node*)malloc(sizeof(Node));
    TempNode->Data=NodeNum+6;
    InsertANode(pHead,TempNode);

    TempNode=(Node*)malloc(sizeof(Node));
    TempNode->Data=8;
    InsertANode(pHead,TempNode);

    PrintLTab(pHead);

    DelLTabSameEle(pHead);
    PrintLTab(pHead);
}
```