习题 1

1.1 简述 C++语言程序的结构特点。

答：

（1）C++程序由一个或多个函数组成，其中至少有一个主函数 main()，程序从主函数开始执行，由主函数来调用其他函数。

（2）C++函数由说明部分与函数体组成，函数体由变量定义和执行语句两部分组成。其中函数体中语句和变量说明以分号结束。一般函数结构如下：

```
# 编译预处理命令
函数类型 <函数名>（形式参数）
{   /* 注释 */
    变量说明；
    执行语句；    // 注释内容
}
```

（3）程序书写规则为同一层语句同列书写、内层语句缩进两个字符、函数定义在第一列书写。严格区分字母的大小写，即大写与小写代表两个不同变量。

（4）C++输入/输出通过流 cin 和 cout 来实现的

1.2 简述 C++程序开发的步骤。

答：

（1）分析问题。根据实际问题，分析需求，确定解决方法，并用适当的工具描述它。

（2）编辑程序。编写 C++源程序，并利用一个编辑器将源程序输入，保存到计算机中的某一个文件中。文件的扩展名为.cpp。

（3）编译程序。编译源程序，产生目标程序。目标程序文件的扩展名为.obj。

（4）连接程序。将一个或多个目标程序与库函数进行连接后，产生一个可执行文件。可执行文件的扩展名为.exe。

（5）运行调试程序。运行可执行文件，分析运行结果。若有错误进行调试修改。

在编译、连接和运行程序过程中，都有可能出现错误，此时要修改源程序，并重复以上过程，直到得到正确的结果为止。

1.3 设计一个 C++程序，输出以下信息：

```
***************
    Hello！
***************
```

解：

```
# include <iostream.h>
void   main()
{   cout<<" ***************"<<'\n';
    cout<<" Hello！ "<<'\n';
    cout<<" ***************"<<'\n';
}
```

1.4 设计一个 C++程序，输入三个学生的成绩，求其总成绩。

解：

```
# include <iostream.h>
```

```
main()
{   int s1,s2,s3,sum;
    cout<<"请输入三个学生的成绩：";
    cin>>s1>>s2>>s3;
    sum=s1+s2+s3;
    cout<<"sum="<<sum<<'\n';
}
```

1.5 设计一个 C++程序，输入 a、b 二个整数，用 sub()函数求两数之差。

解：

```
#include <iostream.h>
void main(void)
{   int sub(int x,int y);
    int a,b,m;
    cout<<"Input a,b:";
    cin>>a>>b;
    m=sub(a,b);
    cout<<"a-b= "<<m<<endl;
}
int sub(int x,int y)
{   int z;
    z=x-y;
    return(z);
}
```

习题 2

2.1 简述标识符定义。指出下列用户自己定义的标识符中哪些是合法的？哪些是非法的？如果是非法的，为什么？

答：

xy √　　Book √　　3ab×(不能数字开头)　　x_2 √　　switch×（保留字）　integer √

page-1×(不能包含“-”)　　_name √　　MyDesk √　　#NO×（不能包含“#”）

y.5×(不能包含“.”)　　char×（保留字）

2.2. C++语言中有哪些数据类型？

答：

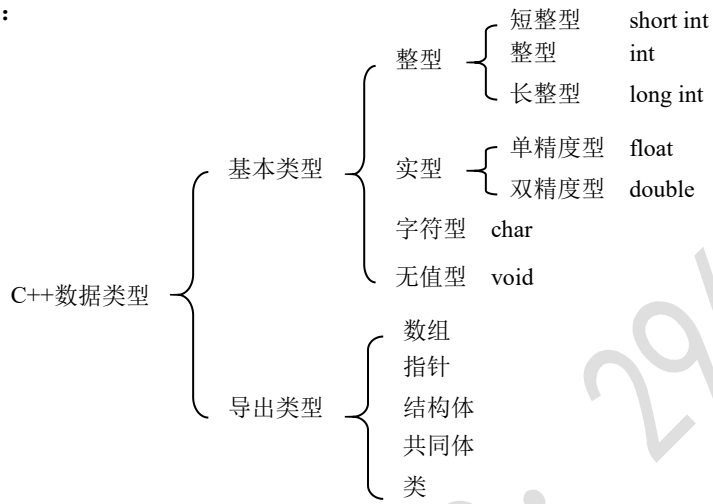

图 2.1 C++的数据类型

2.3 什么是常量？什么是变量？

答：

在程序执行过程中，其值不能被改变的量称为常量。

在程序执行过程中，其值可以改变的量称为变量。变量必须用标识符来命名。

2.4 下列常量的表示在 C++中是否合法？若合法，指出常量的数据类型；若非法，指出原因。

答：

-123 √（整型，十进制）　0321 √（整型，八进制）　.567 √（浮点型，十进制小数）

1.25e2.4 ×（指数部分有“.”，必须是整数）　　32L √（长整型，十进制）

'\t' √（字符型，字符常量）　"Computer" √（字符型，字符串常量）

'x' √（字符型，字符常量）　"x" √（字符型，字符串常量）　'\85'×（85 不是八进数）

2.5 字符常量与字符串常量有什么区别?

答：

C++中字符常量为用单引号括起来的单个字符，而字符串常量为用双引号括起来的多个字符，且字符串最后加入结束标志’\0’。

2.6 求出下列算术表达式的值：

（1）x+a%3*(int)(x+y)%2/4　　设 x=2.5,y=4.7,a=7

答：

按优先级先计算强制类型转换：（int）(2.5+4.7)=7

再按左结合性计算：a%3*7%2/4=7%3*7%2/4=1 *7 % 2/ 4=7%2/4=1/4=0
最后计算 x+0=2.5+0=2.5
（2）(float)(a+b)/2-(int)x%(int)y　　设 a=2,b=3,x=3.5,y=2.5
答：
按优先级先计算强制类型转换：（float）(a+b)/2=5.0 /2=2.5
(int)x%(int)y=(int)3.5%(int)2.5=3%2=1
最后计算(float)(a+b)/2-(int)x%(int)y=2.5-1=1.5
（3）'a'+x%3+5/2-'\24'　　设 x=8
'a'+x%3+5/2-'\24'=97+8%3+5/2-24=97+2+2-20=81
‘\24’为八进制，‘\24’=024（八进制）=2*8+4（十进制）=20（十进制）
'a'的 ASCII 码为 97

2.7 写出以下程序的运行结果。

```
#include <iostream.h>
void main(void)
{    int i,j,m,n;
     i=8;j=10;
     m=++i;n=j++;                    //i=9    m=9 ,n=10    j=11
     cout<<i<<'\t'<<j<<'\n';
     cout<<m<<'\t'<<n<<'\n';
}
```

答：
程序运行结果：

```
9    11
9    10
```

2.8 将下列数学表达式写成 C++中的算术表达式。

（1）$\frac{a+b}{x-y}$　　（2）$\sqrt{p(p-a)(p-b)(p-c)}$

（3）$\frac{\sin x}{2m}$　　（4）$\frac{a+b}{2}h$

解：算术表达式
（1） (a+b)/(x-y)
（2）sqrt(p*(p-a)*(p-b)*(p-c))
（3）sin(x)/2/m　或　sin(x)/（2*m）
（4）(a+b)/2*h

2.9 在 C++语言中如何表示“真”和“假”?系统又是如何判断一个量的“真”和“假”的?
答：
C++语言中用“1”表示“真”，“0”表示“假”。
系统判断“真”和“假”的方法为：一个量非零为“真”，等于零为“假”

2.10 设有变量说明：

```
int   a=3,b=2,c=1;
```

求出下列表达式的值：

（1）a>b　　（2）a<=b
（3）a!=b　　（4）(a>b)==c
（5）a-b==c
解：（1）1　（2） 0　（3）1　（4）1　（5） 1

2.11 设有变量说明：
int　a=3,b=1,x=2,y=0；
求出下列表达式的值：
（1）（a>b）&& (x>y)　　（2）a>b && x>y
（3）(y||b) && (y||a)　　（4）y||b && y||a
（5）!a || a>b
解：　（1）（a>b）&& (x>y)=(3>1) && (2>0) = 1 && 1=1
（2）a>b && x>y=3>1 && 2>0 = 1 && 1=1
（3）(y||b) && (y||a)= (0||1) && (0||3)=1 && 1=1
（4）y||b && y||a = y|| (b && y) ||a =0 || (1 && 0) || 3=0 || 0 || 3= 0 || 3 =1
（5）!a || a>b =!3 || 3>1 =0 || 1 =1

2.12 设有变量说明：
int　w=3,x=10,z=7;
char　ch='D'；
求出下列表达式的值。
（1）w++||z++　　（2）!w>z
（3）w && z　　（4）x>10 || z<9
（5）ch>='A' && ch<='Z'
解：
（1）w++|| z++ = 3 || 7 =1　（先用后加，w=4,z=8）
（2）!w>z = ! 3 > 7　=　0 >7　= 0
（3）w && z　= 3 && 7 =1
（4）x>10 || z<9　=　10>10 || 7<9　= 0 || 1　=1
（5）ch>='A' && ch<='Z'　= 'D'>='A' && 'D'<='Z' =1 && 1 =1

2.13 设 a、b 的值分别为 6、7；指出分别运算下列表达式后 a、b、c、d 的值。
（1）c=d=a　　（2）b+=b
（3）c=b/=a　　（4）d=(c=a/b+15)
解：
（1）c=d=a　c=(d=6) =6　　a=6,b=7,c=6,d=6
（2）b+=b　b=b+b=14　　a=6,b=14,c,d 不定
（3）c=b/=a　c=(b=b/a)=(b=7/6)=1　　a=6,b=1,b=1,c=1,d 不定
（4）d=(c=a/b+15)　d=(c=6/7+15)=(c=0+15)=15　　a=6,b=7,c=15,d=15

2.14 设 a、b、c 的值分别为 5、8、9；指出分别运算下列表达式后 x、y 的值。
（1）y=(a+b,b+c,c+a)　　（2）x=a,y=x+b
解：
（1）y=(a+b,b+c,c+a) =（5+8，8+9，9+5）=（13，17，14）=14　　y=14
（2）x=a, y=x+b = (x=5,y=x+b)=(5,y=5+8)=(5,y=13)=13　　x=5, y=13

2.15 设计一个程序，从键盘输入一个圆的半径，求其周长和面积。

解：

```
# include <iostream.h>
void  main()
{  float r,L,S;
   cout<<"请输入园的半径：";
   cin>>r;
   L=2*3.14*r;
   S=r*r*3.14;
   cout<<"r="<<r<<'\n';
   cout<<"L="<<L<<endl;
   cout<<"S="<<S<<endl;
}
```

2.16 设计一个程序，从键盘输入一个小写字母，将它转换成大写字母。

解：

```
# include <iostream.h>
void  main()
{  char c,C;
   cout<<"请输入小写字母：";
   cin>>c;
   C=c-0x20;      //此为十六进制，也可以用十进制 C=c-32;
   cout<<"C="<<C<<'\n';
}
```

2.17 从键盘输入一个三位数 abc，从左到右用 a、b、c 表示各位的数字，现要求依次输出从右到左的各位数字，即输出另一个三位数 cba，例如：输入 123，输出 321，试设计程序。（算法提示：a=n/100,b=(n-a*100)/10,c=(n-a*100)%10,m=c*100+b*10+a）

解：

```
# include <iostream.h>
# include <math.h>
main()
{  int a,b,c,n,m;
   cout<<"请输入三位十进制整数:";
   cin>>n;
   a=n/100;
   b=n%100/10;
   c=n%100%10;
   m=c*100+b*10+a;
   cout<<"m="<<m<<'\n';
}
```

习题 3

3.1 程序的三种基本控制结构是什么？

答：

程序的三种基本控制结构是：顺序程序结构、分支程序结构、循环结构程序结构。

3.2 C++语言中的语句分哪几类?

答：

C++语言中的语句分为 6 类，即：说明语句、控制语句、函数调用语句、表达式语句、空语句、复合语句。

3.3 怎样区分表达式和语句?

答：

表达式与语句的区别是：表达式不需要用分号结尾，而语句必须以分号结尾。

3.4 程序的多路分支可通过哪二种语句来实现？说出用这两种语句实现多路分支的区别。

答：

程序的多路分支可用 if 语句嵌套实现与 switch 语句来实现。switch 语句只能用于条件表达式的值为整型或字符型的场合，而 if 语句可用于条件表达式为任意类型值的场合。

3.5 使用 switch 开关语句时应注意哪些问题？

答：

使用 switch 开关语句应注意：

（1）必须用 break 才能退出当前 case 语句；

（2）表达式值与常量值只能是整型或字符型；

（3）每个 case 后面的常量表达式的值必须互不相同；

（4）case 后可有多个语句，而不必用花括号。

3.6 用于实现循环结构的循环语句有哪三种？ 分别用于实现哪二种循环结构？这三种循环语句在使用上有何区别？

答：

实现循环结构的循环语句有 while 、do while、for 三种语句。while、for 语句分别用于实现当型循环结构。do while 用于实现直到型循环结构。while、for 语句用于先判断循结束后执行循环体场合，而 do while 用于先执行循环体后判断循环结束的场合。

3.7 分支程序与循环程序常用于解决哪些实际问题？

答 ：

能够用分支与循环结构程序解决的实际问题有：累加和、连乘积、求一批数的和及最大值与最小值、求数列的前 n 项、判素数、求两个整数的最大公约数和最小公倍数、用迭代法求平方根、用穷举法求不定方程组的整数解、打印图形等等。

3.8 continue 语句与 break 语句均用于循环结构，在使用上有何区别？

答：

break 语句只能用在循环语句和 switch 语句中，其功能是终止循环语句和 switch 语句的执行。

continue 语句只能用在循环语句中，其功能是结束本次循环，重新开始下一次循环。

3.9 程序的正常终止与异常终止有何区别，分别用什么函数来实现？使用这些函数时应包含什么头文件？

答：

程序的正常终止用 exit()函数来完成，此时系统要做终止程序执行前的收尾工作，如关闭该程序打开的文件。释放变量所占用的存储空间（不包括动态分配的存储空间）等。

程序的异常终止用 abort ()函数来完成，系统不做结束程序前的收尾工作，直接终止程序的执行。

使用这两个函数时应包含 stdlib.h 头文件？

3.10 写出下列程序的运行结果：

```
#include <iostream.h>
void main(void)
{    int a=2,b=－1,c=2;
     if (a<b)
        if (b<0) c=0;
        else c=c+1;
     cout<<c<<endl;
}
```

解：运行结果：2

3.11 写出下列程序的运行结果：

```
#include <iostream.h>
void main(void)
{    int i=10;
     switch (i)
       {   case     9:i=i+1;
           case    10:i=i+1;
           case    11:i=i+1;
           default :i=i+1;
       }
     cout<<i<<endl;
}
```

解：运行结果：13

3.12 设计一个程序，判断从键盘输入的整数的正负性和奇偶性。

解：

```
# include <iostream.h>
void main()
{   int    i;
    cout<<"请输入一个整数：";
    cin>>i;
    if (i>0)
       if (i % 2==0)
          cout<<i<<" is positive even number"<<endl;
       else
```

```
            cout<<i<<" is positive odd number"<<endl;
        else
          if (i % 2==0)
            cout<<i<<" is negative even number"<<endl;
          else
              cout<<i<<" is negative odd number"<<endl;
    }
```

3.13 有下列函数：

$$y=\begin{cases} -x+2.5 & (x<2) \\ 2-1.5(x-3)^2 & (2\leqslant x<4) \\ \frac{x}{2}-1.5 & (x\geqslant 4) \end{cases}$$

设计一个程序，从键盘输入 x 的值，输出 y 的值。

解：

```
# include <iostream.h>
void main()
{   float   x,y;
    cout<<"please Input x：";
    cin>>x;
    if (x<2)
        y=-x+2.5;
    else if (x>=2 && x<4)
        y=2-1.5*(x-3)*(x-3);
    else
        y=x/2-1.5;
    cout<<"y="<<y<<endl;
}
```

3.14 设计一个程序，从键盘输入 a、b、c 三个整数，将它们按照从大到小的次序输出。

解：

```
# include <iostream.h>
void main()
{   int a,b,c,temp;
    cout<<"please Input a,b,c：";
    cin>>a>>b>>c;
    if (a<b)
        {temp=a;a=b;b=temp;}          //a 与 b 交换
    if (a<c)
          {temp=a;a=c;c=temp;}        //a 与 c 交换
    if (b<c)
          {temp=b;b=c;c=temp;}        //b 与 c 交换
    cout<<a<<'\t'<<b<<'\t'<<c<<endl;
}
```

3.15 输入平面直角坐标系中一点的坐标值（x,y），判断该点是在那一个象限中或那一条坐标轴上。

解：

```
# include <iostream.h>
void main()
{   float x,y;
    cout<<"please Input palne coordinate (x,y)：";
    cin>>x>>y;
    if (x>0 && y>0)
        cout<<"("<<x<<","<<y<<")"<<" is in first quadrant"<<endl;
  else if (x<0 && y>0)
        cout<<"("<<x<<","<<y<<")"<<" is in Second quadrant"<<endl;
  else if (x<0 && y<0)
        cout<<"("<<x<<","<<y<<")"<<" is in third quadrant"<<endl;
  else if (x>0 && y<0)
        cout<<"("<<x<<","<<y<<")"<<" is in four quadrant"<<endl;
  else if (x==0 && y==0)
        cout<<"("<<x<<","<<y<<")"<<" is on origin"<<endl;
  else if (x==0 && y!=0)
        cout<<"("<<x<<","<<y<<")"<<" is on y axis"<<endl;
  else if (x!=0 && y==0)
        cout<<"("<<x<<","<<y<<")"<<" is on x axis"<<endl;
}
```

3.16 简单计算器。设计一个程序计算表达式：data1 op data2 的值，其中 data1、data2 为两个实数，op 为运算符（+、－、*、/），并且都由键盘输入。

解：

```
# include <iostream.h>
void main()
{   float x,y;
    char   op;
    cout<<"please Input x op y：";
    cin>>x>>op>>y;
    switch(op)
    {  case '+':   cout<<x<<op<<y<<"="<<x+y<<endl;break;
       case '-':   cout<<x<<op<<y<<"="<<x-y<<endl;break;
       case '*':   cout<<x<<op<<y<<"="<<x*y<<endl;break;
       case '/':   cout<<x<<op<<y<<"="<<x/y<<endl;break;
       default :   cout<<"Input op error!"<<endl;
    }
}
```

3.17 奖金税率如下：（a 代表奖金，r 代表税率）

a<500(元)	r=0%
500≤a<1000	r=3%

1000≤a<2000　　r=5%
2000≤a<5000　　r=8%
a≥5000　　r=12%

输入一个奖金数，求税率、应交税款及实得奖金数。

解：

```
# include <iostream.h>
void main(void)
{   int a,b;
    float r,tax,prise;
    cout<<"please Input prize :";
    cin>>a;
    if (a>=5000)
        b=10;
    else
        b=a/500;
     switch(b)
     {   case 0:   r=0;break;
         case 1:   r=3;break;
         case 2:
         case 3:   r=5;break;
         case 4:
         case 5:
         case 6:
         case 7:
         case 8:
         case 9:   r=8;break;
         case 10: r=12;break;
         default :cout<<"Input error!";
        }
      tax= a*r/100;
      prise=a*(1-r/100);
      cout<<"Tax rate:"<<r<<"%"<<endl;
      cout<<"Tax :"<<tax<<endl;
      cout<<"Prise:"<<prise<<endl;
}
```

3.18 写出下列程序的运行结果：

```
#include <iostream.h>
void main(void)
{    int n=4;
     while (--n)
        cout<<n<<'\t';
     cout<<endl;
}
```

答：　3　2　1

3.19 写出下列程序的运行结果：

```
#include <iostream.h>
void main(void)
{    int x=3;
     do
       cout<<x<<'\t';
     while (!(x--));
     cout<<endl;
}
```

答：3

3.20 写出下列程序的运行结果：

```
#include <iostream.h>
void main(void)
{    int i=0,j=0,k=0,m;
     for (m=0;m<4;m++)
     switch (m)
     {   case 0:i=m++;
         case 1:j=m++;
         case 2:k=m++;
         case 3:m++;
     }
     cout<<i<<'\t'<<j<<'\t'<<k<<'\t'<<m<<endl;
}
```

答：0　1　2　5

3.21 求 $\sum_{n=1}^{100}\frac{1}{n}$ 的值，即求 $1+\frac{1}{2}+\frac{1}{3}+\frac{1}{4}+\cdots\cdots+\frac{1}{100}$ 的值。

解：

```
# include <iostream.h>
void main()
{    int i,n;
     float s;
     cout<<"Please Input n:";
     cin>>n;
     s=0;
     i=1;
     while (i<=n)
     {    s=s+1.0/i;
          i++;
     }
     cout<<"s="<<s<<endl;
}
```

另解(for 语句)

```
# include <iostream.h>
void main()
{    int i,n;
     float s;
     cout<<"Please Input n:";
     cin>>n;
     for(s=0,i=1;i<=n;i++) s=s+1.0/i;
     cout<<"s="<<s<<endl;
}
```

3.22 编程计算 $y=1+\frac{1}{x}+\frac{1}{x^2}+\frac{1}{x^3}+\cdots\cdots$ 的值（x>1），直到最后一项小于 10^{-4} 为止。

解：用当型语句求解

方法一：

```
# include <iostream.h>
# include <math.h>
void main(void)
{    int i;
     float x,y,z;
     cout<<"Please Input x:";
     cin>>x;
     if (x>1)
     {   i=1; y=1;
         z=1.0/pow(x,i);
         while (z>=0.0001)
         {   y+=z;
             i++;
             z=1.0/pow(x,i);
         }
          cout<<"y="<<y<<'\t'<<"1/xi="<<z<<'\t'<<"i="<<i<<endl;
     }
     else
          cout<<"x<=1    error!";
}
```

方法二：

```
# include <iostream.h>
void main(void)
{    float x,y,z;
     cout<<"Please Input x:";
     cin>>x;
     if (x>1)
     {   y=1;z=1.0/x;
         while (z>=0.0001)
         {   y=y+z;
             z=z/x;
```

```
        }
         cout<<"y="<<y<<endl;
    }
    else
        cout<<"x<=1    error!";
}
```

方法三：

```
# include <iostream.h>
void main(void)
{   float x,y,t;
    cout<<"Please Input x:";
    cin>>x;
    if (x>1)
    {   for(y=1,t=1.0/x;t>=0.0001;t=t/x)
          y=y+t;
        cout<<"y="<<y<<endl;
    }
    else
        cout<<"x<=1    error!";
}
```

方法四(直到型)

```
# include <iostream.h>
# include <math.h>
void main()
{   int i;
    float x,y,z;
    cout<<"Please Input x:";
    cin>>x;
    if (x>1)
    {   i=1;   y=1;
        z=1.0/pow(x,i);
        do
        {   y+=z;
            i++;
            z=1.0/pow(x,i);
        }while (z>=0.0001);
        cout<<"y="<<y<<'\t'<<"1/xi="<<z<<'\t'<<"i="<<i<<endl;
    }
    else
        cout<<"x<=1    error!";
}
```

3.23 输入两个正整数 m 和 n，求其最大公约数和最小公倍数。

解：求两个自然数 m 与 n 的最大公约数采用辗转相除法，

设最小公倍数 (Lease common multiple)为 lcm=m*n；

最大公约数(Gretest common divisor)为 gcd。

可采用如下循环实现：

r=m%n；m←n；n←r；直到 r=0 为止，此时最大公约数为 gcd=n，最小公倍数为 lcm=lcm/gcd

例如：求 m=4 与 n=6 的最大公约数：

r=4%6=4; m←6；n←4；

r=6%4=2; m←4；n←2；

r=4%2=0;

则最大公约数 gcd=n=2

最小公倍数 lcm=4*6/2=12

事实上：4 的公约数为 1、2、4，6 的公约数为 1、2、3、6，所以两数公约数为 1，2，最大公约数为 2。4 的公倍数为 4、8、12、16 等，6 的公倍数为 6、12、18 等，所以两数最小公倍 数为 12。

```
# include <iostream.h>
void main()
{   int m,n,lcm,gcd,r;
    cout<<"please input m,n:";
    cin>>m>>n;
    lcm=m*n;
    while (m%n!=0)
    {    r=m%n;
         m=n;
         n=r;
    }
    gcd=n;
    lcm=lcm/gcd;
    cout<<"Lease common multiple="<<lcm<<endl;
    cout<<"Greatest common divisor="<<gcd<<endl;
}
```

3.24 某月十天内的气温为：-5、3、4、0、2、7、0、5、-1、2（℃），编程统计出气温在 0℃以上、0℃和 0℃以下各多少天？并计算出这十天的平均气温值。

解：

```
# include <iostream.h>
void main(void)
{   int i,positive=0,zero=0,negative=0;
    float f,sum=0,ave;
    cout<<"请输入 10 天的气温:";
    for (i=1;i<=10;i++)
    {    cin>>f;
         if (f>0) positive++;
         else if (f==0) zero++;
         else negative++;
         sum+=f;
    }
    ave=f/10;
```

```
    cout<<"大于零度:"<<positive<<"天\n 等于零度:"
        <<zero<<"天\n 小于零度:"<<negative<<"天"<<endl;
    cout<<"平均温度："<<ave<<endl;
}
```

3.25 求 $\sum_{n=1}^{10} n!$，即求 1!+2!+3!+4!+……+10!。

解:

```
# include <iostream.h>
void main(void)
{   int i,n,k=1;
    long s=0;
    cout<<"Please Input n:";
    cin>>n;
    for(i=1;i<=n;i++)
    {   k=k*i;
        s=s+k;
    }
    cout<<"s="<<s<<endl;
}
```

3.26 设用 100 元钱买 100 支笔，其中钢笔每支 3 元，圆珠笔每支 2 元，铅笔每支 0.5 元，问钢笔、圆珠笔和铅笔可以各买多少支（每种笔至少买 1 支）？

解:

设钢笔、圆珠笔和铅笔各买 i、j、k 支，则应有下列式子成立。

3*i+2*j+0.5*k=100

i+j+k=100

用穷举法将购买钢笔、圆珠笔的数量用二重循环遍历一遍。从中找出符合上述条件的购买方法。

```
# include <iostream.h>
void main(void)
{   int i,j,k;
    for (i=1;i<=33;i++)
        for (j=1;j<=49;j++)
        {   k=100-i-j;
            if (k%2==0 && 3*i+2*j+k/2==100)
                cout<<"pen="<<i<<"\t ball pen="<<j<<"\t pencil="<<k<<endl;
        }
}
```

3.27 编程显示如下图形：

```
    *
   ***
  *****
 *******
*********
 *******
  *****
   ***
    *
```

解:

```
# include <iostream.h>
void main(void)
{   int i,j,n;
```

```
    cout<<"请输入棱形的对角线半长度 n:";
    cin>>n;
    for (i=1;i<=n;i++)   //画上半个棱形
    {   for(j=1;j<=40-i;j++) cout<<" ";
        for(j=1;j<=2*i-1;j++) cout<<"*";
        cout<<endl;
    }
    for (i=n-1;i>0;i--)   //画下半个棱形
    {   for(j=1;j<=40-i;j++) cout<<" ";
        for(j=1;j<=2*i-1;j++) cout<<"*";
        cout<<endl;
    }
}
```

习题 4

4.1 什么是数组？

答：

数组是若干个同类型数据元素的集合。

4.2 一维数组与二维数组在内存中是如何存储的？

答：

一维数组 a[n]定义后，系统分配 n*k 个连续存储单元，用于存放数组元素值，其中 k=元素占用字节数。

二维数组 a[m][n]定义后，系统分配 m*n*k 个连续存储单元，用于存放数组元素值，其中 k=元素占用字节数。

存放方式为按行存放，即先存第一行元素，再存第二行元素，依次把各行元素存入一串连续的存储单元中。

4.3 有如下数组定义：

int　a[20];

指出该数组的数组名、数组元素类型、数组元素个数、第一个数组元素的下标值和最后一个数组元素的下标值。

答：

数组名为 a，数组元素类型为整型，数组元素个数为 20，第一个数组元素的下标值为 0 和最后一个数组元素的下标值为 19。

4.4 写出下列程序的运行结果：

```
#include <iostream.h>
void main(void)
{    int i,k,a[10],p[3];
     k=5;
     for (i=0;i<10;i++)          // a[0]=0，…，a[9]=9
        a[i]=i;
     for (i=0;i<3;i++)
        p[i]=a[i*(i+1)];         // p[0]=a[0]=0，p[1]=a[2]=2，p[2]=a[6]=6，
     for (i=0;i<3;i++)
        k=k+p[i]*2;              // k=5+0*2+2*2+6*2=21
     cout<<k<<endl;
}
```

答：　21

4.5 写出下列程序的运行结果：

```
#include <iostream.h>
void main(void)
{    int a[6][6],i,j;
     for (i=1;i<6;i++)
        for (j=1;j<6;j++)
           a[i][j]=(i/j)*(j/i);      //当 i<j 时 i/j=0  当 j<i 时 j/i=0  所以只有当 i=j 时 a[i][j]=1
```

```
    for (i=1;i<6;i++)
      {   for (j=1;j<6;j++)
            cout<<a[i][j]<<'\t';
          cout<<endl;
      }
}
```

答：

a[i][j]=(i/j)*(j/i);　　//当 i<j 时 i/j=0 当 j<i 时 j/i=0 所以

a[i][j]=1, i=j

a[i][j]=0, i≠j

即矩阵对角线上元素为 1，其它元素为 0。输出结果为对角线矩阵：

1 0 0 0 0 0
0 1 0 0 0 0
0 0 1 0 0 0
0 0 0 1 0 0
0 0 0 0 1 0
0 0 0 0 0 1

4.6 写出下列程序的运行结果：

```
#include <iostream.h>
#include <string.h>
void main(void)
{   char str[80];
    int i,j,k;
    cout<<"Input string:";
    cin>>str;
    for (i=0,j=strlen(str)-1;i<j;i++,j--)
      {   k=str[i];
          str[i]=str[j];
          str[j]=k;
      }
    cout<<str<<endl;
}
```

运行时输入：abcdef(回车)

答：fedcba

4.7 某班有 30 个学生，进行了数学考试，编写程序将考试成绩输入一维数组，并求数学的平均成绩及不及格学生的人数。

解：

```
#include <iostream.h>
#define N 10
void main(void)
{   float math[N],sum,ave;
    int i,count;
    cout<<"Input math score：";
```

```
    sum=0;
    count=0;
    for(i=0; i<N;i++)
        cin>>math[i];
    for(i=0;i<N;i++)
    {   sum=sum+math[i];
        if (math[i]<60)     count++;
    }
    ave=sum/N;
    cout<<"ave="<<ave<<'\t'<<"count="<<count<<endl;
}
```

4.8 设有一个数列，它的前四项为 0、0、2、5，以后每项分别是其前四项之和，编程求此数列的前 20 项。用一维数组完成此操作。(提示：a[i]=a[i-1]+a[i-2]+a[i-3]+a[i-4];)

解：

```
#include <iostream.h>
#include <iomanip.h>
#define N 20
void main(void)
{   long i,a[N+1];
    a[1]=a[2]=0;
    a[3]=2;
    a[4]=5;
    for (i=5;i<=N;i++)
       a[i]=a[i-1]+a[i-2]+a[i-3]+a[i-4];
    for (i=1;i<=N;i++)
    {   cout<<setw(8)<<a[i];
        if (i%5==0)
        cout<<endl;
    }
}
```

4.9 某班有 30 个学生，进行了数学考试，编写程序将考试成绩输入一维数组，并将数学成绩用冒泡法、选择法与擂台法三种排序算法，由高到低的顺序排序后输出。

解：方法一：冒泡法

```
#include <iostream.h>
#include <iomanip.h>
#define N 30
void main(void)
{   float math[N];
    int i,j,temp;
    cout<<"Input math score："<<endl;
    for(i=0;i<N;i++)
        cin>>math[i] ;
    for(i=0;i<N-1;i++)
```

```
        for(j=0;j<=N-1-i;j++)
        {   if   (math[j]<math[j+1])
            {   temp=math[j];
                math[j]=math[j+1];
                math[j+1]=temp;
            }
        }
        for (i=0;i<N;i++)
        {   cout<<setw(6)<<math[i]<<endl;
        }
}
```

方法二：选择法

```
#include <iostream.h>
#include <iomanip.h>
#define N 30
void main(void)
{   float math[N];
    int i,j,temp;
    cout<<"Input math score："<<endl;
    for(i=0; i<N;i++)
        cin>>math[i];
    for(i=0;i<N-1;i++)
       for(j=i+1; j<N;j++)
       {   if   (math[i]<math[j])
           {    temp=math[i];
                math[i]=math[j];
                math[j]=temp;
           }
       }
       for (i=0;i<N;i++)
          cout<<setw(6)<<math[i]<<endl;
}
```

方法三：擂台法

```
#include <iostream.h>
#include <iomanip.h>
#define N 30
void main(void)
{   float math[N];
    int i,j,temp,k;
    cout<<"Input math score："<<endl;
    for(i=0; i<N;i++)
        cin>>math[i];
    for(i=0;i<N-1;i++)
      { k=i;
        for(j=i+1; j<N;j++)
```

```
            if    (math[k]<math[j]) k=j;
        if (k>i)
        { temp=math[i]; math[i]=math[k]; math[k]=temp;}
       }
    for (i=0;i<N;i++)
        cout<<setw(6)<<math[i]<<endl;
}
```

4.10.已有一按从小到大次序排序好的数组，现输入一数，要求按原来排序的规律将它插入到数组中。

解：先定位、向后移，再插入

```
#include <iostream.h>
#include <iomanip.h>
#define N 10
void main(void)
{   float a[N];
    int i,b,j;
    cout<<"Input sort array a[9]："<<endl;
    for(i=0;i<N-1;i++)
        cin>>a[i];
    cout<<"Input number b:";
    cin>>b;
    i=0;
    while (a[i]<b) i++;
    for(j=N-1;j>i;j--)    a[j]=a[j-1];
    a[i]=b;
    for (i=0;i<N;i++)
       cout<<setw(6)<<a[i]<<endl;
}
```

4.11 定义一个二维数组，用编程的方法形成如下矩阵（数据不能通过键盘输入），并按下列格式输出矩阵。（提示：当 i>=j 时，a[i][j]=1；当 i<j 时，a[i][j]=j-i+1）

$$A = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 \\ 1 & 1 & 2 & 3 & 4 \\ 1 & 1 & 1 & 2 & 3 \\ 1 & 1 & 1 & 1 & 2 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

解：

```
#include <iostream.h>
#include <iomanip.h>
#define N 5
void main(void)
{   float a[N][N];
    int i,j;
```

```
    for(i=0; i<N;i++)
    for(j=0; j<N;j++)
      {   if (i>=j)   a[i][j]=1;
          else   a[i][j]=j-i+1;
      }
    for(i=0; i<N;i++)
    {   for(j=0; j<N;j++)
          cout<<setw(6)<<a[i][j];
        cout<<endl;
    }
}
```

4.12 设计一个程序，打印杨辉三角形。

```
                    1
                  1   1
                1   2   1
              1   3   3   1
            1   4   6   4   1
          1   5  10  10   5   1
        1   6  15  20  15   6   1
      1   7  21  35  35  21   7   1
    1   8  28  56  70  56  28  8   1
  1   9  36  84 126 126  84  36  9   1
```

解：方法一

杨辉三角形第 n 行第 m 个元素值为多项式$(1+x)^n$的系数$C_n^m=\frac{n!}{m!(n-m)!}$值。

如第 4 行各元素值为：

$C_4^0=\frac{4!}{0!(4-0)!}=1$　$C_4^1=\frac{4!}{1!(4-1)!}=4$　$C_4^2=\frac{4!}{2!(4-2)!}=6$　$C_4^3=\frac{4!}{3!(4-3)!}=4$　$C_4^4=\frac{4!}{4!(4-4)!}=1$

所以只要计算出组合数$C_n^m=\frac{n!}{m!(n-m)!}$即可得到第 n 行第 m 个元素值。

用二重循环即可打印出杨辉三角形。在输出时每行控制输出的空格，以达到图示的效果。

```
#include <iostream.h>
#include <iomanip.h>
#define N 11
void main(void)
{   int c[N][N],m,n,k,m1,n1,n_m;
    for(n=0; n<N;n++)
      for(m=0; m<=n;m++)
      {   for(m1=1,k=1;k<=m;k++)   m1*=k;
          for(n1=1,k=1;k<=n;k++)   n1*=k;
          for(n_m=1,k=1;k<=n-m;k++)   n_m*=k;
          c[n][m]=n1/(m1*n_m);
```

```
        }
      for(n=0;n<N;n++)
      { if（n%2==0）cout<<"     " ;
        for (m=0;m<(N-n)/2;m++)
           cout<<setw(6)<< ' ';
        for(m=0; m<=n;m++)
            cout<<setw(6)<<c[n][m];
        cout<<endl;
      }
}
```

方法二

从杨辉三角形各元素数值可以看出如下规律：

（1）第一列和对角线元素值为 1

（2）其它其元素值为 c[n][m]=c[n-1][m-1]+c[n-1][m]

由此可编写程序如下：

```
#include <iostream.h>
#include <iomanip.h>
#define N 11
void main(void)
{   int c[N][N],m,n,k,m1,n1,nm1;
    for(n=1;n<N;n++)
    {   c[n][n]=1;
        c[n][1]=1;
    }
    for(n=3; n<N;n++)
      for(m=2; m<=n-1;m++)
        c[n][m]=c[n-1][m-1]+c[n-1][m];
    for(n=1; n<N;n++)
    {   if（n%2==0）cout<<"     " ;
        for (m=0;m<(N-n)/2;m++)
          cout<<setw(6)<< ' ';
        for(m=1;m<=n;m++)
          cout<<setw(6)<<c[n][m];
        cout<<endl;
    }
}
```

4.13 输入一个字符串，求出字符串长度（不能用 strlen 函数），输出字符串及其长度。

解：

```
#include <iostream.h>
#include <iomanip.h>
#define N 30
void main(void)
{   char str[N];
    int k,l,len;
```

```
    cout<<"Input a String"<<endl;
    cin>>str;
    len=0;
    k=0;
    while(str[k]!=0)
    {   k++;
        len++;
    }
    cout<<str<<endl;
    cout<<" Length of String："<<len<<endl;
}
```

4.14 输入一行字符，分别统计出其中英文字母、空格、数字字符和其它字符的个数。

解：

```
#include <iostream.h>
#include <iomanip.h>
#define N 30
void main(void)
{    char str[N];
     int f,i,k1,k2,k3,k4;
     cout<<"Input a String"<<endl;
     cin.getline(str,N);
     k1=0;
     k2=0;
     k3=0;
     k4=0;
     i=0;
     while(str[i]!=0)
     {   if ((str[i]>=0x41) && (str[i]<=0x5a) || (str[i]>=0x61) && (str[i]<=0x7a)) k1++;
         else if ((str[i]>=0x30) && (str[i]<=0x39)) k2++;
         else if (str[i]==0x20) k3++;
         else k4++;
         i++;
     }
     cout<<"字母:"<<k1<<"个\t 数字:"<<k2<<"个\t 空格:"<<k3<<"个\t 其它:"<<k4<<"个"<<endl;
}
```

4.15 编一程序，对从键盘输入的两个字符串进行比较，然后输出两个字符串中第一个不同字符的 ASCII 码之差。例如：输入的两个字符串分别为“abcdef”和“abceef”，则第一个不同字符为“d”和“e”，输出为-1。

解：

```
#include <iostream.h>
#include <string.h>
#define N 30
void main(void)
```

```
{   char s1[N],s2[N];
    int i,len;
    cout<<"Input   first String"<<endl;
    cin>>s1;
    cout<<"Input   second   String"<<endl;
    cin>>s2;
    if (strlen(s1)>strlen(s2))   len=strlen(s2);
    else len=strlen(s1);
    i=0;
    while (i<len && s1[i]==s2[i]) i++;
      if (i==len)
         cout<<"s1=s2";
      else
         cout<<"s1["<<i<<"]-s2["<<i<<"]="<<s1[i]-s2[i]<<endl;
}
```

4.16 从键盘输入三个字符串，将其合并成一个字符串，并求出合并后字符串的长度。

解：

```
#include <iostream.h>
#include <string.h>
#define N 30
void main(void)
{    char s1[N],s2[N],s3[N],s[3*N];
     cout<<"Input   first String"<<endl;
     cin>>s1;
     cout<<"Input   second   String"<<endl;
     cin>>s2;
     cout<<"Input   third   String"<<endl;
     cin>>s3;
     strcpy(s,s1);
     strcat(s,s2);
     strcat(s,s3);
     cout<<"s="<<s<<endl;
 }
```

4.17 已知某运动会上男子百米赛跑决赛成绩。要求编写程序，按成绩排序并按名次输出排序结果，包括输出名次、运动员号码和成绩三项内容。

解：

用 M 行 3 列数组存放运动员号码、成绩与名次，对决赛成绩降序排序，最后按排序后的位置输入名次。

```
# include <iostream.h>
#include <iomanip.h>
# define M 5
void main(void)
{    int a[M][3],i,j,k,temp;
```

```
    cout<<"Input No and final grade：";
    for (i=0;i<M;i++)
       cin>>a[i][0]>>a[i][1];
    for (i=0;i<M-1;i++)
    {   k=i;
        for(j=i+1;j<M;j++)
        if (a[k][1]<a[j][1]) k=j;
        if (k!=i)
        {   temp=a[i][0];a[i][0]=a[k][0];a[k][0]=temp;
            temp=a[i][1];a[i][1]=a[k][1];a[k][1]=temp;
        }
    }
    for (i=0;i<M;i++)   a[i][2]=i+1;
    cout<<setw(6)<<"No." <<setw(6)<<"Grade" <<setw(6)<<"Order"<<endl;
    cout<<"-----------------------------------------------------------------"<<endl;
    for (i=0;i<M;i++)
    { for (j=0;j<3;j++)
        cout<<setw(6)<<a[i][j];
      cout<<endl;
    }
    cout<<"-----------------------------------------------------------------"<<endl;
}
```

4.18 某小组有 5 个学生，考了 3 门课程，他们的学号及成绩如表 4.2 所示，试编程求每个学生的总成绩及每门课的平均成绩，并按表格形式输出每个学生的学号、3 门课程成绩、总成绩及各门课程的平均成绩。要求用一个 6 行 5 列的数组完成上述操作。

表 4.3　学生成绩情况表

学 号	数 学	语 文	外 语	总成绩
1001	90	80	85	
1002	70	75	80	
1003	65	70	75	
1004	85	50	60	
1005	80	90	70	
最高分				

```
#include <iostream.h>
#include <iomanip.h>
#define M 6
#define N 5
void main(void)
{   float s[M][N],sum,max;
    int i,j;
    cout<<"Input data:\n";                  //输入数据
    for (i=0;i<M-1;i++)                     //输入 5 个学生的学号与 3 门课成绩
```

```
    {   for (j=0;j<N-1;j++)
            cin>>s[i][j];
    }
    for (i=0;i<M-1;i++)                          //处理数据
    {   sum=0.0;
        for (j=1;j<N-1;j++)                      //计算每个学生的总成绩
            sum=sum+s[i][j];
        s[i][N-1]=sum;                           //计算每个学生的总成绩
    }
    for (j=1;j<N;j++)                            //处理数据
    {   max=s[0][j];
        for (i=0;i<M-1;i++)                      //处理计算每门课程
            if (max<s[i][j])
                max=s[i][j];
         s[M-1][j]=max;                            //计算每门课程的最高分
    }
    cout<<setw(5)<<" Num. "<<"    Math.    Chin.    Engl.    Sum."<<endl;    //输出数据
    cout<<"-------------------------------------------------------------\n";
    for (i=0;i<M;i++)
    {   for (j=0;j<N;j++)                        //输出学号、3 门课程的成绩与总分
            if (i==M-1 && j==0) cout<<setw(6)<<"  最高分  ";
            else cout<<setw(6)<<s[i][j];
        cout<<endl;
    }
    cout<<"-------------------------------------------------------------\n";
}
```

4.19 对 4.18 题中学生成绩表用擂台法按总成绩排序后输出，并输出每门课程不及格学生的成绩信息。

```
#include <iostream.h>
#include <iomanip.h>
#define M 6
#define N 5
void main(void)
{   float s[M][N],sum,ave,temp;
    int i,j,k;
    cout<<"Input data:\n";                       //输入数据
    for (i=0;i<M-1;i++)                          //输入 5 个学生的学号与 3 门课成绩
    {   for (j=0;j<N-1;j++)
            cin>>s[i][j];
    }
    for (i=0;i<M;i++)                            //处理数据
    {   sum=0.0;
        for (j=1;j<N-1;j++)                      //计算每个学生的总成绩
            sum=sum+s[i][j];
```

```
        s[i][N-1]=sum;                          //计算每个学生的总分
    }
    for (j=1;j<N;j++)                           //处理数据
    {   max=s[0][j];
        for (i=0;i<M-1;i++)                     //处理计算每门课程
            if (max<s[i][j])
                max=s[i][j];
         s[M-1][j]=max;                           //计算每门课程的最高分
    }
    for (i=0;i<M-2;i++)
    {   k=i;
        for(j=i+1;j<M-1;j++)
          if (s[k][4]<s[j][4]) k=j;
        if (k!=i)
          for (j=0;j<N;j++)
          {   temp=s[i][j];s[i][j]=s[k][j];s[k][j]=temp;}
    }
    cout<<setw(5)<<" Num. "<<"   Math.   Chin.   Engl.   Sum."<<endl;    //输出数据
    cout<<"-----------------------------\n";
    for (i=0;i<M;i++)
    {   for (j=0;j<N;j++)                       //输出学号、3 门课程的成绩与总分
            if (i==M-1 && j==0) cout<<setw(6)<<"最高分";
            else cout<<setw(6)<<s[i][j];
        cout<<endl;
    }
    cout<<"-----------------------------\n";
}
```

4.20 设 A 为 m 行 n 列矩阵，B 为 n 行 k 列矩阵，C 为 m 行 k 列矩阵。设计矩阵乘法程序，能完成 C=A*B 的操作。m、n 与 k 用 define 定义为常量，其值由用户自定义。

解：

```
#include <iostream>
#include <iomanip.h>
#define M 2
#define N 3
#define P 4
void main(void)
{   int   a[M][N],b[N][P],c[M][P],i,j,k;
    cout<<"Input a["<<M<<"]["<<N<<"]:"<<endl;
    for (i=0;i<M;i++)
      for (j=0;j<N;j++)
        cin>>a[i][j];
    cout<<"Input b["<<N<<"]["<<P<<"]:"<<endl;
    for (i=0;i<N;i++)
      for (j=0;j<P;j++)
```

```
            cin>>b[i][j];
        for (i=0;i<M;i++)
           for (j=0;j<P;j++)
           {   c[i][j]=0;
               for (k=0;k<N;k++)
               c[i][j]=c[i][j]+a[i][k]*b[k][j];
           }
        for (i=0;i<M;i++)
        {     for (j=0;j<P;j++)
                  cout<<setw(4)<<c[i][j];
              cout<<endl;
        }
}
```

习题 5

5.1　什么是函数？函数是如何分类的？

答：

函数是实现特定功能的相对独立的程序段。

函数分成：标准库函数和用户自定义函数，主调函数和被调函数，无参函数和有参函数。库函数可以直接使用，而用户自定义函数通常是先定义，然后使用。

5.2　自定义函数如何定义？有哪三种调用方式？

答：

自义函数定义一般格式为：

〔类型〕〈函数名〉(〈形式参数表〉)

{〈语句序列〉}

自定义函数三种调用形式为：函数调用语句、函数表达式、函数参数。

5.3　在函数的调用过程中，实参传送给形参有哪两种方式？这两种传送方式有什么区别？

答：

在函数的调用过程中，实参传送给形参有值传送与传地址两种形式。

“值传送”方式：调用函数时，系统给形参分配存储单元，使用形参与实占用不同的内存空间，形参值的改变不影响实参，实参保持原来的值。

“传地址”方式：调用函数时，系统将实参地址传递给形参，实与形参占用相同的内存空间。形参的变化会影响实参的变化，使形参与实参具有相同的值。

5.4　什么情况下必须使用函数的原型说明？在函数的原型说明的形参表中，形参名是否是必须的？为什么？

答：

在 C++程序中，当函数调用在前、函数定义在后时，则应在主调函数中，在调用前增加对被调函数的原型说明。因为在函数的原型说明的形参表中，只须说明形参的类型与个数，所以形参名不是必须的。

5.5　什么是递归？递归方法解决问题时，必须分析清楚哪三个问题？

答：

函数的递归调用指的是在一个函数定义的函数体中又出现直接或间接地调用该函数本身。

利用递归方法解决问题时，必须注意三点：递归的公式、递归的结束条件、递归的限制条件。

5.6　什么是作用域？在 C++中，作用域分成哪 5 类？在前 4 类作用域中，变量有效作用区域是什么？

答：

作用域是变量在程序中可引用的区域。在 C++中，作用域共分成五种：块作用域、文件作用域、函数原型作用域、函数作用域和类作用域。

1. 块作用域：从块内变量定义处到块的结束处

2. 文件作用域:从函数外变量定义开始到文件结束（可用 extern 进行扩展）;

3. 函数原型作用域：从函数原型变量定义开始到函数原型说明结束;

4. 函数作用域：从函数开始到函数结束；

5.7　什么是局部变量与全局变量？局部变量与全局变量的作用域是什么？什么是静态变量与动态变量？静态变量与动态变量分别存放在什么存储区？静态变量与动态变量各自生存期是什么？

答：

在一个函数内部定义的变量或在一个块内定义的变量称为局部变量。局部变量都具有块作用域。在函数外定义的变量称为全局变量。全局变量具有文件作用域。

在程序的执行过程中，为其分配存储空间的变量称为动态变量，存储在内存中的动态存储区，生存期为从变量定义开始到作用域结束。

在程序开始执行时系统就为变量分配存储空间，直到程序执行结束时，才收回为变量分配的存储空间，这种变量称为静态变量，存储在内存中的静态存储区，生存期为程序执行的全过程。

5.8　变量的存储类型有哪四种？按存储类型，变量可分为哪五种？叙述每类存储变量的存储特性。

答：

变量的存储类型分为四种：自动类型（auto）、静态类型(static)、寄存器类型(register)、外部类型(extern)。按存储类型，变量可分为自动变量、寄存器变量、局部静态变量、全局静态变量、外部变量。各类变量的存储特性如下：

变量类型	全局、局部变量	作用域	静、动态变量	存储区
自动变量	局部变量	块作用域	动态变量	动态存储区
寄存器变量	局部变量	块作用域	动态变量	CPU 寄存器
局部静态变量	局部变量	块作用域	静态变量	静态存储区
全局静态变量	全局变量	文件作用域	静态变量	静态存储区
外部变量	全局变量	文件作用域	静态变量	静态存储区

5.9　如何定义内联函数？使用内联函数的实质与目的是什么？

答：

在类型前加关键字 inline 定义的函数称为所谓内联函数，定义格式为：

inline　<类型>　<函数名>（<形式参数表>）

{函数体}

内联函数的实质就是在编译时把函数的函数体直接插入到调用处。其目的是降低系统的开销，提高程序的执行效率。

5.10　什么是函数的重载？调用重载函数时，通过什么来区分不同的重载函数？

函数的重载是指用重名函数完成不同功能的函数运算。调用重载函数时，通过重载函数形参（参数个数不同或者数据类型不同）来区分不同的重载函数。

5.11 写出下列程序的运行结果：

```
#include <iostream.h>
void modify(int x,int y)
{   cout<<"x="<<x<<'\t'<<"y="<<y<<endl;
    x=x+3; y=y+4;
    cout<<"x="<<x<<'\t'<<"y="<<y<<endl;
}
```

```
void main(void)
{   int a,b;
    a=1;b=2;
    cout<<"a="<<a<<'\t'<<"b="<<b<<endl;
    modify(a,b);
    cout<<"a="<<a<<'\t'<<"b="<<b<<endl;
}
```

解：

运行结果：

```
a=1       b=2
x=1       y=2
x=4       y=6
a=1       b=2
```

5.12 写出下列程序运行后的输出图形：

```
#include <iostream.h>
void main(void)
{   int i=4,j;
    void printchar(int,char);
    printchar(25,' ');
    j=i;
    printchar(i+2*j-2,'*');
    cout<<endl;
    for (j=2;j>=0;j--)
    {   printchar(28-j,' ');
        printchar(i+2*j,'*');
        cout<<endl;
    }
}
void printchar(int len,char c)
{   int k;
    for (k=1;k<=len;k++)
    cout<<c;
}
```

解：

```
          **********
           ********
            ******
             ****
```

5.13 写出下列递归程序的递归公式、限制条件、结束条件及运行结果：

```
#include <iostream.h>
int fac(int n)
{   int z;
    if (n>0)
```

```
        z=n*fac(n-2);
    else
        z=1;
    return z;
}
void main(void)
{   int x=7,y;
    y=fac(x);
    cout<<y<<endl;
}
```

解：

递归公式：　　fac(n)=1;　　当 n=0,n=1;

　　fac(n)=n*fac(n-2);　　当 n>0

限制条件：　n>=0

结束条件：　n=0

运行结果：　105

5.14 写出下列以数组为参数的函数调用运行结果：

```
#include <iostream.h>
#include <string.h>
void main(void)
{   char a[]="abcdef";
    int n;
    void fun(char s[],int k);
    n=strlen(a);
    fun(a,n);
    cout<<a<<endl;
}
void fun(char s[],int k)
{   int x,y;
    char c;
    x=0;
    for (y=k-1;x<y;y--)
    {   c=s[y]; s[y]=s[x]; s[x]=c;
        x++;
    }
}
```

解：

运行结果：　fedcba

5.15 指出下列程序各函数中的全局变量与局部变量，静态变量与动态变量，各变量的存储类型、作用域与生存期，并写出下列程序的运行结果：

```
#include <iostream.h>
extern int x;
void change(void)
```

```
{   register int y=0,z=3;
    cout<<x<<'\t'<<y<<'\t'<<z<<endl;
    x=2;y=2;
    cout<<x<<'\t'<<y<<'\t'<<z<<endl;
}
int x=3,y=4;
void main(void)
{   auto int x,z= -3;
    x=1;
    cout<<x<<'\t'<<y<<'\t'<<z<<endl;
    change();
    cout<<x<<'\t'<<y<<'\t'<<z<<endl;
    cout<<::x<<'\t'<<::y<<'\t'<<z<<endl;
}
```

解：

```
1    4    -3
3    0    3
2    2    3
1    4    -3
2    4    -3
```

5.16 指出下列各函数中各变量的存储类型、作用域与生存期，写出下列程序的运行结果：

```
#include <iostream.h>
void main(void)
{   int i;
    void add1(void),add2(void);
    for (i=0;i<3;i++)
    {   add1();
        add2();
        cout<<endl;
    }
}
void add1(void)
{   int x=0;
    x++;
    cout<<x<<'\t';
}
void add2(void)
{   static int x=0;
    x++;
    cout<<x<<'\t';
}
```

解：

```
1    1
1    2
```

1　3

5.17 指出下列各文件中变量的存储类型、作用域与生存期，写出下列程序的运行结果：

```
//文件名：exercise5_17.cpp
#include <iostream.h>
int x=1,y=2;
static int z=3;
extern void add(void);
void main(void)
{   add();
    cout<<"x="<<x<<'\t'<<"y="<<y <<'\t'<<"z="<<z<<endl;
}
//文件名：exercise5_171.cpp
#include <iostream.h>
extern int x,y;
void add(void)
{   x+=3;
    y+=4;
    cout<<"x="<<x<<'\t'<<"y="<<y<<endl;
}
```

在第二个文件中能否将第一个文件中的变量 z 定义为外部变量？

解：

x=4　　y=6

x=4　　y=6　　z=3

在第二个文件中可以将第一个文件中的变量 z 定义为外部变量。

5.18 编写一个函数把华氏温度转换成摄氏温度，温度转换公式为：c=(f-32)*5/9。在主函数中输入华氏温度值，转换后输出相应的摄氏温度值。

解：

```
#include <iostream.h>
float fc(float f)
{   float c;
    c=(f-32)*5/9;
    return c;
}
void main(void)
{   float c,f;
    cout<<"请输入华氏温度值:";
    cin>>f;
    c=fc(f);
    cout<<"摄氏温度值:"<<c<<endl;
}
```

程序运行后

请输入华氏温度值: 41

摄氏温度值:5

5.19 编写一个函数判断一个整数是否为素数。在主函数中输入一个整数，输出该整数是否为素数的信息。（提示：参考例 3.24）

解：

```
#include <iostream.h>
#include <math.h>
#include <iomanip.h>
int prime(int a)
{    int k,i;
     k=sqrt(a);
     for (i=2;i<=k;i++)                    //判断 a 是否是素数
        if (a%i==0)
            break;
        if (i>k)                           //若 i>k，则 i 为素数
            return 1;
        else
            return 0;
}
void main(void)
{   int a,b;
    cout<<"请输入一个整数:";
    cin>>a;
    b=prime(a);
    if (b==1)
        cout<<a<<"是素数!"<<endl;
    else
        cout<<a<<"不是素数!"<<endl;
}
```

运行程序后

请输入输入一个整数: 7

7 是素数

5.20 编写一个函数 power(float x,int n)，用于计算 x 的 n 次幂。在主函数中实现输入输出。

解：

```
#include <iostream.h>
# include <math.h>
float power(float x,int n)
{   float y=1;
    int i;
    for(i=1;i<=n;i++)
         y=y*x;
    return y;
}
void main(void)
{   float x,y;
```

```
    int n;
    cout<<"请输入自变量 x 与指数 n:";
    cin>>x>>n;
    y=power(x,n);
    cout<<"幂函数值:"<<y<<endl;
}
```

运行程序后

请输入自变量 x 与指数 n：2　3

幂函数值:8

5.21 编写一个计算 1 至 n 的平方和的函数，并调用此函数计算：

$$(1^2+2^2+\cdots+12^2)+(1^2+2^2+\cdots+15^2)^2$$

解：

```
#include <iostream.h>
#include <math.h>
long square_sum(int n)
{   long sum=0;
    int i;
    for(i=1;i<=n;i++)
        sum=sum+i*i;
    return sum;
}
void main(void)
{   long s;
    s=square_sum(12)+square_sum(15)*square_sum(15);
    cout<<"S="<<s<<endl;
}
```

程序运行后输出：

S=1538250

5.22 编写两个函数，分别求两个整数 m、n 的最大公约数和最小公倍数。在主函数中输入两个整数，分别调用这两个函数求得结果并输出。求两个整数 m、n 的最大公约数和最小公倍数的算法提示如下：

（1）将 m、n 中的最大数赋给变量 a，最小数赋给变量 b。

（2）用大数 a 除以小数 b，若余数 c 为 0，则余数 c 为最大公约数，否则进行（3）。

（3）将小数 b 赋给 a，余数 c 赋给 b，再进行（2），直到余数等于 0 为止。

（4）最小公倍数=（m*n）/最大公约数。

例如：求 20 与 14 的最大公约数方法：20%14=6，14%6=2，6%2=0，则 2 为 20 与 14 的最大公约数。最小公倍数=20*14/2=140。

解:

```
# include <iostream.h>
int gcd(int m,int n)
{   int r;
    while (m%n!=0)
```

```
    {    r=m%n;
         m=n;
         n=r;
    }
    return n;
}
int lcm(int m,int n)
{    int lcm;
     lcm=m*n;
     lcm=lcm/gcd(m,n);
     return lcm;
}
void main(void)
{   int m,n;
    cout<<"please input m,n:";
    cin>>m>>n;
    cout<<"Lease common multiple="<<lcm(m,n)<<endl;
    cout<<"Gretest common divisor="<<gcd(m,n)<<endl;
}
```

程序运行后

```
please input m,n: 6   4
Lease common multiple=12
Gretest common divisor=2
```

5.23 编写一个函数用递归的方法求 1+2+3+4+…+n 的值。在主函数中进行输入输出。

解:

（1）递归公式为：

$$sum(n)=\begin{cases}1 & ;\ n=1\\ sum(n-1)+n & ;\ n>1\end{cases}$$

（2）递归结束条件：　n=1

（3）递归约束条件：　n>1

```
# include <iostream.h>
void main(void)
{   int n;
    int sum(int n);
    cout<<"please input n:";
    cin>>n;
    cout<<"sum="<<sum(n)<<endl;
}
int sum(int n)
{   int s;
     if (n==1)
          s=1;
    else
```

```
        s=sum(n-1)+n;
    return s;
}
```

程序运行后:

```
please input n:6
sum=21
```

5.24 编写一个函数 power(float x,int n)用递归的方法计算 x 的 n 次幂。在主函数中实现输入输出。

（1）递归公式为：

$$power(n)=\begin{cases} x & ;\ n=1 \\ power(n-1)*x & ;\ n>1 \end{cases}$$

（2）递归结束条件： n=1

（3）递归约束条件： n>1

```
# include <iostream.h>
void main(void)
{   float x;
    int n;
    float power(float,int);
    cout<<"please input x,n:";
    cin>>x>>n;
    cout<<"power="<<power(x,n)<<endl;
}
float power(float x,int n)
{   float p;
     if (n==1)
         p=x;
    else
         p=power(x,n-1)*x;
    return p;
}
```

程序执行后:

```
please input x,n:2 4
power=16
```

5.25 编写一个排序函数用选择法对一批整数按从大到小的次序进行排序。在主函数内输入数据，调用排序函数对数据排序，输出排序结果。

解：

```
# include <iostream.h>
# define N 5
void sort1(int a[])
{   int i,j,k,temp;
    for (i=0;i<N-1;i++)
    {   k=i;
```

```
        for (j=i+1;j<N;j++)
            if (a[k]<a[j])
                k=j;
        if (i!=k)
        { temp=a[i];a[i]=a[k];a[k]=temp;}
    }
}
void main(void)
{   int a[N],i;
    cout<<"Please input a["<<N<<"]:";
    for (i=0;i<N;i++)
        cin>>a[i];
    sort1(a);
    for (i=0;i<N;i++)
        cout<<a[i]<<'\t';
    cout<<endl;
}
```

程序运行后：

```
Please input a[5]: 5 8 4 1 3
8    5    4    3    1
```

5.26 用输入 input(int s[6][5])、计算 calculate(int s[][5],int n)、输出 output(int s[6][5])三个函数完成习题 4.21 所要求功能。在主函数中定义二维数组 s[6][5]，调用 input()、calculate()、output()完成学号与成绩的输入、总成绩与最高分的计算、结果的输出。

解：

```
#include <iostream.h>
#include <iomanip.h>
#define M 6
#define N 5

void input(float x[M][N])
{    int i,j;
     cout<<"Input data:\n";                    //输入数据
     for (i=0;i<M-1;i++)                       //输入 5 个学生的学号与 3 门课成绩
       for (j=0;j<N-1;j++)
           cin>>x[i][j];
}
void calculate(float x[][N],int n)
{   int i,j;
    float sum,max;
    for (i=0;i<M-1;i++)                        //处理数据
      {   sum=0.0;
          for (j=1;j<N-1;j++)                  //计算每个学生的总成绩
              sum=sum+x[i][j];
          x[i][N-1]=sum;                       //计算每个学生的总成绩
```

```
        }
        for (j=1;j<N;j++)                          //处理数据
        {   max=x[0][j];
            for (i=0;i<M-1;i++)                    //处理计算每门课程
                if (max<x[i][j])
                    max=x[i][j];
             x[M-1][j]=max;                        //计算每门课程的最高分
        }
    }
    void output(float x[M][N])
    {   int i,j;
        cout<<setw(5)<<" Num. "<<"   Math.   Chin.   Engl.   Sum."<<endl;    //输出数据
        cout<<"---------------------------------------------------------------\n";
        for (i=0;i<M;i++)
        {   for (j=0;j<N;j++)                      //输出学号、3 门课程的成绩与总分
                if (i==M-1 && j==0) cout<<setw(6)<<"  最高分  ";
                else cout<<setw(6)<<x[i][j];
            cout<<endl;
        }
        cout<<"---------------------------------------------------------------\n";
    }
    void main(void)
    {   float s[M][N];
        input(s[M][N]);
        calculate(s[][N],6);
        output(s[M][N]);
    }
```

5.27 用内联函数实现求出一维数组的最大值。在主函数中输入数组元素值，调用求最大值函数，并输出数组最大值。

解：

```
    # include <iostream.h>
    # define N 5
    inline int maxf(int a[])
    {   int i,max;
        max=a[0];
        for (i=1;i<N;i++)
          if (max<a[i])
              max=a[i];
        return max;
    }
    void main(void)
    {   int a[N],i;
        cout<<"Please input a["<<N<<"]:";
        for (i=0;i<N;i++)
```

```
        cin>>a[i];
    cout<<maxf(a)<<endl;
}
```

程序运行后：

```
Please input a[5]: 5 8 4 1 3
Max=8
```

5.28 编写一个函数求长方体的体积，长方体的长、宽、高的默认值分别为 30、20、10。在主函数中进行输入输出。

解：

```
# include <iostream.h>
float cuboid(float a,float b,float c)
{   float v;
    v=a*b*c;
    return v;
}
void main(void)
{   float l,w,h,v;
    cout<<"Please l、w、h:";
    cin>>l>>w>>h;
    cout<<"volume:"<<cuboid(l,w,h)<<endl;
}
```

5.29 编写两个名为 max 的重载函数，分别实现求两个整数和两个实数中的大数。

解：

```
# include <iostream.h>
float max(float a,float b)
{   return a>b?a:b;
}
int max(int a,int b)
{   return a>b?a:b;
}
void main(void)
{   int a,b;
    float   x,y ;
    cout<<"Please a、b:";
    cin>>a>>b;
    cout<<"max="<<max(a,b)<<endl;
    cout<<"Please x、y:";
    cin>>x>>y;
    cout<<"max="<<max(x,y)<<endl;
}
```

将光标停留在程序的第 3 行，按 ctrl+F10 键，运行程序到

```
Please x、y:1.2 3.4
```

后，转移到第 3 行，再按 F5 键执行结束。这说明当输入数据为实型时，系统会自动调用实型的 max 函数求最大值。而当输入数据为整型时，系统会自动调用整型的 max 函数求最大值。

习题 6

6.1 什么是编译预处理？C++具有哪几种编译处理功能？

答：

在编译前，对源程序中的预处理命令作处理的过程称为编译预处理。C++具有：

（1）文件包含

（2）宏定义

（3）条件编译

三种编译处理功能。

6.2 什么是“文件包含”处理？在什么情况下需要使用“文件包含”处理？

答：

将另一个头文件(.h)内容包含到本源文件中来称为文件包含处理。

当要调用标准库函数时，或使用用户自定义头文件时，需要使用文件包含处理。

6.3 什么是宏定义？宏定义有哪两种形式？

答：

用指定标识符（宏名）来代表一个字符串称为宏定义。

宏定义分为无参的宏定义和带参的宏定义两种形式。

6.4 带参数的宏定义和函数有什么区别？

答：

（1）两者的定义形式不一样。宏定义中只给出形式参数，而不要指明每一个形式参数的类型；而在函数定义时，必须指定每一个形式参数的类型；

（2）函数调用是在程序运行时进行的，分配临时的内存单元；而宏调用则是在编译前进行的，并不分配内存单元，不进行值的传递处理。

（3）函数调用时，先求实参表达式的值，然后将值代入形参；而宏调用时只是用实参简单地替换形参；

（4）函数调用时，要求实参和形参的类型一致；而宏调用时不存在类型问题；

（5）使用宏次数多时，宏展开后源程序变长，因为每一次宏展开都使源程序增长；而函数调用不使源程序变长；

6.5 设计一个程序，将求两个实数中的较大数函数放在头文件中，在源程序文件中包含该头文件，并实现输入二个实数，求出其最大值。

exercise6_5.h

```
float maxf(float a,float b)
{   return    (a>b)?a:b; }
```

exercise6_5.cpp

```
# include <iostream.h>
# include “exercise6_5.h”
void main(void)
{    float x,y;
     cout<<”Input x,y:”;
     cin>>x>>y;
     cout<<”max=”<<maxf(x,y)<<endl;
}
```

6.6 设计一个程序，定义一个带参数的宏以求两个数中的较大数。在主函数中输入三个数，求出其中的最大数。

解：

```
# include <iostream.h>
# define MAX(a,b)    (a)>(b)?(a):(b)
void main(void)
{    float x,y,z,m;
     cout<<"Input x,y,z,m:";
     cin>>x>>y>>z;
     m=MAX(x,y);
     if (m>z)
        cout<<"max="<<m<<endl;
     else
        cout<<"max="<<z<<endl;
}
```

6.7 设计一个程序，分别用带参数的宏和函数求矩形的面积、周长。矩形边长用键盘输入。

解法一：用带参宏定义

```
# include <iostream.h>
# define AREA(a,b)    a*b
# define PREIMETER(a,b) 2*(a+b)
void main(void)
{    float x,y;
     cout<<"Input x,y:";
     cin>>x>>y;
     cout<<"Area="<<AREA(x,y)<<endl;;
     cout<<"Preimete="<<PREIMETER(x,y)<<endl;
}
```

解法二：用函数

```
# include <iostream.h>
float area(float a,float b)
{    return a*b;}
float preimeter(float a,float b)
{ return    2*(a+b);}
void main(void)
{    float x,y;
     cout<<”Input x,y:”
     cin>>x>>y;
     cout<<”Area=”<<area(x,y)<<endl;;
     cout<<”Preimete=”<<preimeter(x,y)<<endl;
}
```

6.8 设计一个程序，三角形三条边用键盘输入，用带参数的宏求三角形的面积、周长。最后输出三角形的面积、周长。在调试程序阶段，定义标识符 DEBUG，使用条件编译命令输出调试信息(如：三角形的三条边的边长)。程序调成功后，删除定义标识符 DEBUG 的命令，不输出调试信息。

```
//# define DEBUG
# include <math.h>
# include <iostream.h>
# define L(a,b,c) (a+b+c)/2
# define AREA(a,b,c)    sqrt(L(a,b,c)*( L(a,b,c)-a)*( L(a,b,c)-b)*( L(a,b,c)-c))
# define PREIMETER(a,b,c) a+b+c
void main(void)
{    float a,b,c;
     cout<<"Input a,b,c:";
     cin>>a>>b>>c;
     # ifdef DEBUG
     cout<<"a="<<a<<'\t'<<"b="<<b<<'\t'<<"c="<<c<<'\t'<<endl;
     #endif
     cout<<"Area="<<AREA(a,b,c)<<endl;;
     cout<<"Preimete="<<PREIMETER(a,b,c)<<endl;
}
```

习题 7

7.1 什么叫指针？什么叫指针变量？有哪些类型的指针？

答：

指针是变量、数组、字符串、函数等在内存的首地址，指针变量是存放指针的变量。指针变量按定义格式大致可分为五种：指针变量、指针数组、指向一维数组的指针变量、返回指针值的函数、函数指针变量。

7.2 指针变量按定义格式分为哪 5 类？，每类指针变量如何定义？

答：

指针变量按定义格式大致可分为五种：指针变量、指针数组、指向一维数组的指针变量、返回指针值的函数、函数指针变量。

指针变量：〔存储类型〕 <类型>　*<指针变量名>

指针数组：〔存储类型〕 <类型>　*<指针变量名>[长度]

指向一维数组的指针变量：〔存储类型〕 (<类型> *)<指针变量名>[长度]

返回指针值的函数：〔存储类型〕 <类型>　*<指针变量名>（形参表）

函数指针变量：〔存储类型〕 <类型>　（*<指针变量名>）（形参表）；

7.3 定义一个整型指针变量 pi，用什么方法，才能使 pi 指向整型变量 i，指向整型一维数组 a 的首地址，指向整型二维数组 b 的首地址。

答：

pi 指向整型变量 i：　pi=&i;

指向整型一维数组 a 的首地址：pi=a;

指向整型二维数组 b 的首地址：pi=&b[0][0];

7.4 叙述二维数组 a 的行地址、行首地址、元素地址的概念、作用及表示方法，写出元素 a[i][j]值的表示方法。

答：

（1）二维数组 a 的第 i 行首地址是第 i 行第 0 列元素地址&a[i][0]。

有三种表示方式：&a[i][0]、a[i]、*(a+i)、&a[i][0]。

（2）二维数组 a 的第 i 行的行地址是用于指向一维数组指针变量的地址。

有二种表示方式：a+i、&a[i]

（3）二维数组 a 的元素 a[i][j]的地址为该元素在内存中的地址，

有四种表示方式：a[i]+j 、*(a+i)+j、&a[i][0]+j、&a[i][j]

（4）元素 a[i][j]的值的表示方式：*(a[i]+j) 、 *(*(a+i)+j)、*(&a[i][0]+j)、a[i][j]

7.5 引用类型变量与其相关变量有何关系？引用类型变量主要用于什么场合？

答：

引用类型变量是其相关变量的别名，引用类型变量与相关变量使用相同的内存空间，所以用引用类型变量作为函数形参时，形参与实参变量使用相同的内存，在函数内对形参的运算就是对实参的运算，因此可通过参数返回函数运算结果。所以引用类型变量主要用于函数的形参，用于函数之间传送数据。

7.6 对字符串指针变量，能否用 cin、cout 对其输入/输出字符串？能否对字符串指针变量进行赋值运算？字符串指针变量能否作为字符串处理函数的参数？

答：
对字符串指针变量，能用 cout 对其输出字符串，但不能用 cin 对其输入字符串；
能对字符串指针变量进行赋值运算；
字符串指针变量能作为字符串处理函数的参数。

7.7 读下列程序，并写出程序运行结果。
（1）

```
# include <iostream.h>
void main (void)
{   float   x=1.5,y=2.5,z;
    float   *px,*py;
    px=&x;
    py=&y;
    z= * px + *py;
    cout<<"x="<<*px<<'\t'<<"y="<<*py<<"z="<<z<<'\n';
}
```

解：
x=1.5　　y=2.5　　z=4.0

（2）

```
# include <iostream.h>
void main( void)
{   int   a[5]={10,20,30,40,50};
    int   *p=&a[0];                  //p 指向 a[0]
    p++;                             //p 指向 a[1]
    cout<< *p<<'\t';                 //输出 a[1]=20
    p+=3;                            //p 指向 a[4]
    cout<< *p<<'\t';                 //输出 a[4]=50
    cout<< *p--<<'\t';               //先输出 a[4]=50，后 p 指向 a[3]
    cout<<++ *p<<'\n';               //先对 a[3]内容加 1，后输出 a[3]=41
}
```

解：
20　50　50　41

（3）

```
# include<iostream.h>
  void f(int *a,int b)
  {   int t=*a;*a=b;b=t;}
  void main(void)
  {   int x=10,y=20;
      cout<<x<<'\t'<<y<<'\n';
          f(&x,y);
           cout<<x<<'\t'<<y<<'\n';
  }
```

解：
10　20

20　20

7.8 编写程序，用 4 种方式求整型一维数组的和。4 种方式是指 4 种不同的数组元素表达方式。

解：

方法一 （指针方式，改变 p，用 *p 访问元素）

```
#include <iostream.h>
#define N 10
void main(void)
{    float a[N],sum,*p;
     cout<<"Input data:";
     for (p=a;p<a+N;p++)
         cin>>*p;
     for (p=a,sum=0;p<a+N;p++)
          sum=sum+*p;
     cout<<"sum="<<sum<<endl;
}
```

方法二（指针方式，首地址+位移，用 *（p+i）访问元素）

```
#include <iostream.h>
#define N 10
void main(void)
{   float a[N],sum,*p=a;
    int i;
    cout<<"Input integers:";
    for (i=0;i<N;i++)
        cin>>*(p+i);
     sum=0;
     for (i=0;i<N;i++)
         sum=sum+*(p+i);
     cout<<"sum="<<sum<<endl;
}
```

方法三（数组方式，指针变量的数组形式，用 p[i]访问元素）

```
#include <iostream.h>
#define N 10
void main(void)
{    float a[N],sum,*p=a;
     int i;
     cout<<"Input data:";
     for (i=0;i<N;i++)
        cin>>p[i];
     sum=0;
     for (i=0;i<N;i++)
         sum=sum+p[i];
     cout<<"sum="<<sum<<endl;
}
```

方法四（指针方式，数组名+位移，用 *（a+i）访问元素）

```
#include <iostream.h>
#define N 10
void main(void)
{    float a[N],sum;
     int i;
     cout<<"Input data:";
     for (i=0;i<N;i++)
        cin>>*(a+i);
     sum=0;
     for (i=0;i<N;i++)
          sum=sum+*(a+i);
     cout<<"sum="<<sum<<endl;
}
```

7.9 编写程序，用以下二种方法求实型二维数组 a[3][4]的最大值。

（1）用指针变量 （2）用表 7.1 中的数组元素表示法中的第三种

解：（1）

```
#include <iostream.h>
#define N 10
void main(void)
{    float a[3][4],max,*p;
     int i;
     cout<<"Input data:";
     for (p=&a[0][0];p<&a[0][0]+12;p++)
        cin>>*p;
     p=&a[0][0];
     max=*p++;
     for (;p<&a[0][0]+12;p++)
          if (max<*p)
               max=*p;
     cout<<"max="<<max<<endl;
}
```

（2）

```
#include <iostream.h>
void main(void)
{    float a[3][4],max;
     int i,j;
     cout<<"Input data:";
     for (i=0;i<3;i++)
          for (j=0;j<4;j++)
             cin>>*(&a[i][0]+j);
     max=*(&a[0][0]+0);
     for (i=0;i<3;i++)
          for (j=0;j<4;j++)
```

```
            if (max<*(&a[i][0]+j))
                max=*(&a[i][0]+j);
        cout<<"max="<<max<<endl;
}
```

7.10　用指针变量编写下列字符串处理函数：

（1）字符串拷贝函数，void str_cpy(char *p1,char *p2){函数体}

（2）字符串比较函数，int str_cmp(char *p1,*char *p2) {函数体}

（3）取字符串长度函数，int str_len(char *p){函数体}

在主函数中输入两个字符串，对这两个字符串进行比较，并输出比较结果。然后将第一个字符串拷贝到第二个字符串，输出拷贝后的字符串及其长度。

解：

```
# include <iostream.h>
int str_cmp(char *p1,char *p2)
{   while(*p1==*p2)
      {p1++;
       p2++;
      }
    if (*p1>*p2)
          return 1;
    else if (*p1==*p2)
          return 0;
    else
          return -1;
}
void str_cpy(char *p1,char *p2)
{   while(*p1!=0)
       *p2++=*p1++;
    *p1=0;
}
int str_len(char *p)
{   int length=0;
    while(*p!=0)
    {   p++;
        length++;
    }
    return length;
}
void main(void)
{   char s1[20],s2[40];
    cout<<"Input String1:";
    cin>>s1;
    cout<<"Input String2:";
    cin>>s2;
    if (str_cmp(s1,s2)==1)
```

```
        cout<<"String1>String2"<<endl;
    else if (str_cmp(s1,s2)== -1)
        cout<<"String2>String1"<<endl;
    else
        cout<<"String1=String2"<<endl;
    str_cpy(s1,s2);
    cout<<"String2 ="<<s2<<endl;
    cout<<" String2 Length="<<str_len(s2)<<endl;
}
```

另解：

```
# include <iostream.h>
char * str_cmp(char *p1,char *p2)
{    char *q1=p1,*q2=p2;
     while(*p1==*p2)
     {p1++;   p2++;     }
    if (*p1>=*p2)
      return q1;                //返回指向目标串首地址的指针 q1。
    else
      return q2;                //返回指向目标串首地址的指针 q2。
}

char   *str_cpy(char * p1,char *p2)
{   char *p=p2;                     //将目标串首地址赋给指针变量 p。
    while (*p1);                    //  将源串 s1 中的字符依次复制到目标串 s2 中。
      *p2++=*p1++;
    *p1=0;
    return p;                       //返回指向目标串首地址的指针 p。
}

int str_len(char *p)
{   int l=0;
    while(*p!=0)
    {   p++;
        l++;
    }
    return l;
}

void main(void)
{   char s1[40],s2[20];
    cout<<"Input String1:";
    cin.getline(s1,20);
    cout<<"Input String2:";
    cin.getline(s2,20);
```

```
    cout<<str_cmp(s1,s2)<<endl;
    cout<<"String2+String1="<<str_cat(s2,s1)<<endl;
    cout<<"Length="<<slen(s1)<<endl;
}
```

7.11 定义一个二维字符数组 c[5][80]及指针数组 s[5]，用 cin.getline(s[i],80) 输入 5 个字符串到二维数组 5 行中，然后用指针数组 s 对字符串进行降序排列（要求用擂台法），最后用指针数组 s 输出排序后的结果。

解：

```
#include <iostream.h>
# include <string.h>
void main(void)
{   char c[5][80];
    char *s[5]={c[0],c[1],c[2],c[3],c[4]};
    char   *pc;
    int i,j,k;
    cout<<"Input 5 String:"<<endl;
    for (i=0;i<5;i++)
         cin>>s[i];
    for (i=0;i<4;i++)
    {   k=i;
        for (j=i+1;j<5;j++)
        if (strcmp (s[i],s[j])<0)
             k=j;
        if   (k!=i)
        {   pc=s[i];s[i]=s[k];s[k]=pc;
        }
    }
    for ( i=0;i<5;i++) cout<<s[i]<<endl;
}
```

7.12 输入一个二维数组 a[6][6]，设计一个函数，用指向一维数组的指针变量和二维数组的行数作为函数的参数，求第 0 行与最后一行两行元素的平均值，并输出。。

解：

方法一：用指向一维数组的指针变量和二维数组的行数作为函数的参数

```
#include <iostream.h>
float ave(float (*p)[6],int n)
{   float sum=0;
    int i,j;
     for(j=0;j<6;j++)
          {    sum+=(*p)[j];
               sum+=(*(p+n-1))[j];
          }
    return sum/n/2;
}
```

```
void main(void)
{   float a[6][6],average;
    int i,j;
    cout<<"Input Data:";
    for (i=0;i<6;i++)
        for (j=0;j<6;j++)
            cin>>a[i][j];
    average=ave(a,6);
    cout<<"ave="<<average<<'\n';
}
```

方法二：用数组名为函数参数，在函数内用指向一维数组的指针变量，求出平均值。

```
#include <iostream.h>
float fun(float a[6][6],int n)
{   float sum=0,(*p)[6];
    int i,j;
    {   p=a;
        for(j=0;j<n;j++)
          sum+=(*p)[j];
p=a+n-1;
        for(j=0;j<n;j++)
          sum+=(*p)[j];
    }
return sum/2/n;
}
void main(void)
{   float b[6][6],average;
    int i,j;
    cout<<"Input Data:";
    for (i=0;i<6;i++)
        for (j=0;j<6;j++)
            cin>>b[i][j];
    average=fun(b,6);
    cout<<"ave="<<average<<'\n';
}
```

7.13 用指针与数组作为函数参数，按如下四种情况用擂台法对一维实型数组进行降序排序。

（1）函数的实参为数组名，形参为数组（用擂台法）。

（2）函数的实参为数组名，形参为指针变量（用冒泡法）。

（3）函数的实参为指针变量，形参为数组（用选择法）。

（4）函数的实参为指针变量，形参为指针变量（用选择法）。

解：

```
#include <iostream.h>
void sort1( int a[ ],int n)                    //形参为数组名，擂台法
{   int i,j,k,temp;
```

```
    for (i=0;i<n-1;i++)
    {   k=i;
        for (j=i+1;j<n;j++)
        if   (a[k]<a[j]) k=j;
        if (k!=i)
        { temp=a[i]; a[i]=a[k]; a[k]=temp; }
    }
}
void sort2( int *p,int n)                    //形参为指针，冒泡法
{   int i,j,temp;
    for (i=0;i<n-1;i++)
    {   for (j=0;j<n-i-1;j++)
        if   (*(p+j)<*(p+j+1))
          { temp=*(p+j); *(p+j)=*(p+j+1); *(p+j+1)=temp; }
    }
}

void sort3( int a[ ],int n)                  //形参为数组名，擂台法
{   int i,j,k,temp;
    for (i=0;i<n-1;i++)
    {   k=i;
        for (j=i+1;j<n;j++)
        if   (*(a+k)<*(a+j)) k=j;
        if (k!=i)
        { temp=*(a+i); *(a+i)=*(a+k); *(a+k)=temp; }
    }
}
void sort4( int *p,int n)                    // 形参为指针，选择法
{   int i,j,k,temp;
    for (i=0;i<n-1;i++)
    {   for (j=i+1;j<n;j++)
        if   (p[i]<p[j])
        { temp=p[i]; p[i]=p[j]; p[j]=temp; }
    }
}
void main( void)
{   int a1[6]={1,3,2,5,4,6},*pi, i;
    int a2[6]={1,3,2,5,4,6};
    int a3[6]={1,3,2,5,4,6};
    int a4[6]={1,3,2,5,4,6};
    sort1(a1,6);                             //实参为数组名，形参为数组
    sort2(a2,6);                             //实参为数组名，形参为指针变量
    pi=a3;
    sort3(pi,6) ;                            //实参为指针变量，形参为数组
    pi=a4;
```

```
    sort4(pi,6) ;                          //实参为指针变量，形参指针变量
    for (i=0;i<6;i++) cout<<a1[i]<<'\t';
    cout<<endl;
    for (i=0;i<6;i++) cout<<a2[i]<<'\t';
    cout<<endl;
    for (i=0;i<6;i++) cout<<a3[i]<<'\t';
    cout<<endl;
    for (i=0;i<6;i++) cout<<a4[i]<<'\t';
    cout<<endl;
}
```

7.14 设计返回指针值的函数，将输入的一个字符串按逆向输出。

解：

```
# include <iostream.h>
# include <string.h>
char *reverse(char *p, int n)
{   int i=0,j=n-1;
    char temp;
    while(i<j)
    {   temp=*(p+i);
        *(p+i)=*(p+j);
        *(p+j)=temp;
        i++;
        j--;
    }
    return p;
}
void main(void)
{   char s[30];
    int l;
    cout<<"Input String:";
    cin>>s;
    l=strlen(s);
    cout<<reverse(s,l)<<endl;
}
```

7.15 将字符串存入字符数组 s，然后将 s 中的每一个字符按例【7.21】中程序加密，设计对应的解密程序，解密函数为：

$$s[i]=\begin{cases} s[i]+16 & ; \text{ 当 } 32\leqslant s[i]\leqslant 41 \\ s[i]+23 & ; \text{ 当 } 42\leqslant s[i]\leqslant 67 \\ s[i]-5 & ; \text{ 当 } 102\leqslant s[i]\leqslant 127 \end{cases}$$

其中 s[i]为字符串中第 i 个字符的 ASCII 码。输入的字符串只能由字符或数字组成。

解：

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

```
char *encrypt(char *pstr)                    //定义加密函数为返回指针值的函数
{   char *p=pstr;                            //保存字符串首地址到指针变量 p
    while (*pstr)
    { if (*pstr>=48 && *pstr<=57) *pstr-=16;          //用指针变量*pstr 表示 s[i]，
      else if (*pstr>=65 && *pstr<=90) *pstr-=23;     //进行加密处理
      else if (*pstr>=97 && *pstr<=122) *pstr+=5;
      pstr++;                                //指针变量加 1，指向下个单元
    }
    return p;                                //返回字符串指针
}

char *revert(char *pstr)                     //定义加密函数为返回指针值的函数
{   char *p=pstr;                            //保存字符串首地址到指针变量 p
    while (*pstr)
    { if (*pstr>=32 && *pstr<=41) *pstr+=16;          //用指针变量*pstr 表示 s[i]，
      else if (*pstr>=42 && *pstr<=67) *pstr+=23;     //进行加密处理
      else if (*pstr>=102 && *pstr<=127) *pstr-=5;
      pstr++;                                //指针变量加 1，指向下个单元
    }
    return p;                                //返回字符串指针
}
void main(void)
{   char s[80];
    cin.getline(s,80);                       //输入字符串到数组 s
    cout<< encrypt(s)<<endl;     // 将字符串加密后输出
    cout<<revert(s)<<endl;
}
```

7.16 设计程序，编写加法函数 add()与减法函数 sub()。在主函数中定义函数指针变量，用函数指针变量完成两个操作数的加、减运算。

解：

```
# include <iostream.h>
float add(float x,float y)
{   return x+y;}
float sub(float x,float y)
{   return x-y;}
void main(void)
{   float x,y;
    char operate;
    float (*f)(float,float);
    cout<<"Input x operate y ="<<endl;
    cin>>x>>operate>>y ;
    switch (operate)
    {
       case '+':f=add;break;
```

```
        case '-':f=sub;break;
    }
    cout<<"="<<f(x,y)<<endl;
}
```

7.17 设计一个用矩形法求定积分的通用函数，被积函数的指针、积分的上限、积分的下限和积分的区间等分数作为函数的参数。分别求出下列定积分的值。

$$s1=\int_1^2(1+\ln(x)+x^3)dx$$

$$s2=\int_{-1}^4(\frac{1}{1+x^2})dx$$

$$s3=\int_1^3\frac{x+e^x}{1+\sin(x)+x^2}dx$$

解：

分析：由高等数学可知，$\int_a^b f(x)dx$ 的定积分值等于由曲线 y=f(x)、直线 x=a 、x=b、 y=0 所围曲边梯形的面积 S，如图 7.15 所示。现将曲边梯形划分成 n 个小曲边梯形△S0、△S1、△S2、…、△Sn-1。每个曲边梯形的高均为 h=(b－a)/n，用矩形近似曲边梯形后各曲边梯形的面积近似为：

△S0=y_0*h

△S1=y_1*h

△S2 =y_2*h

…

△Sn-1=y_{n-1}*h

S =△S0+△S1+△S2+…+△Sn-1=(y_0+y_1+y_2+…+y_{n-1})*h

=((f (x_0)+(f (x_1)+f (x_2)+…+f (x_{n-1}))*h

∵x_0=a ，x_n=b，x_i=a+i*h

∴用梯形法求定积分面积的公式为：$s=[\sum_{i=0}^{i=n-1}f(a+i\times h)]\times h$

其中：a、b 分别为积分的下、上限，n 为积分区间的分隔数，h=(b－a)/n，h 为积分步长；f(x)为被积函数。

程序编写如下：

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

```
# include <iostream.h>
float f1(float x)
{   return (1+log(x)+x*x*x);}
float f2(float x)
{   return (1/(1+x*x));}
float f3(float x)
{   return (x+exp(x))/(1+sin(x)+x*x);}
float integral(float (*f)(float),float a,float b,int n)
{
   float y,h;
   int i;
   y=0;
   h=(b-a)/n;
   for (i=0;i<n;i++) y+=f(a+i*h);
   return (y*h);
}
void main (void )
{
   cout<<"s1="<<integral(f1,1,2,1000)<<endl;
   cout<<"s2="<<integral(f2,-1,4,1000)<<endl;
   cout<<"s3="<<integral(f3,1,3,1000)<<endl;
}
```

7.18 定义一个指向字符串的指针数组，用一个函数完成 n 个不等长字符串的输入，根据实际输入的字符串长度用 new 运算符分配存储空间，依次使指针数组中的元素指向每一个输入的字符串。设计一个完成 n 个字符串按升序排序的函数（在排序过程中，要求只交换指向字符串的指针值，不交换字符串）。在主函数中实现将排序后的字符串输出。

解：

```
# include <iostream.h>
# include <string.h>
const int N=5;
void input(char *p[],int n)
{   char s[50];
    int i;
    for (i=0;i<n;i++)
    {   cout<<"Input String["<<i<<"]:";
        cin>>s;
        p[i]=new char[strlen(s)+1];
        strcpy(p[i],s);
    }
}
void sort(char *p[],int n)
{   int i,j,k;
    char *pt;
    for (i=0;i<n-1;i++)
```

```
    {   k=i;
        for(j=i+1;j<n;j++)
        if (strcmp(p[k],p[j])>0) k=j;
        if (i!=k)
        {   pt=p[k];p[k]=p[i];p[i]=pt;}
    }
}
void main(void)
{   char *p[N];
    int i;
    input(p,N);
    sort(p,N);
    for (i=0;i<N;i++)
    {   cout<<p[i]<<endl;
        delete []p[i];
    }
}
```

7.19 编写求一维数组平均值的函数，用引用类型变量作为函数参数返回平均值。在主函数中输入数组元素值，并输出平均值。

解：

```
# include <iostream.h>
void ave(float *p,float &average,int n)
{   int i;
    for(i=0;i<n;i++)
          average+=*p++;
     average=average/n;
}
const int N=10;
void main(void)
{   int i;
    float a[N],aver=0;
    cout<<"Input Arrary:";
    for (i=0;i<N;i++)
          cin>>a[i];
    ave(a,aver,N);
    cout<<"ave="<<aver<<endl;
}
```

7.20 用引用类型变量作为函数参数，编写对三个变量进行按升序排序的函数，在主函数中输入三个变量的值，输出排序后三个变量的值。

解：

```
# include <iostream.h>
void sort(float &a,float &b,float &c)
{       float temp;
```

```
        if (a>b)
        {    temp=a;a=b;b=temp;}
        if (a>c)
        {    temp=a;a=c;c=temp;}
        if (b>c)
        {    temp=b;b=c;c=temp;}
    }
    void main(void)
    {   float x,y,z;
        cout<<"Input x,y,z:";
        cin>>x>>y>>z;
        sort(x,y,z);
        cout<<x<<'\t'<<y<<'\t'<<z<<'\n';
    }
```

7.21 有哪三种方法来定义 const 型指针？这三种 const 指针限制哪些变量的值不能改变？

答：

（3）const 型指针

有三种方法来定义 const 型指针：

const <类型> *<指针变量名>；

定义指针变量所指数据值不能改变，但指针变量值可以改变。

<类型> * const <指针变量名>；

定义指针变量值不能改变，但指针变量所指数据值可以改变。

const <类型> * const <指针变量名>；

指针变量值与指针变量所指数据值都不能改变。

习题 8

8.1 什么是枚举类型？如何定义枚举类型？枚举类型中元素的默认序号值是如何确定的？序号值是否一定要惟一？用枚举类型定义的枚举变量只能进行哪二种运算？枚举变量能否用 cin 输入元素值？能否用 cout 输出枚举变量的值？若能则输出的内容是什么？

答：

（1）枚举类型是某类数据可能取值的集合；

（2）定义枚举类型

enum <枚举类型名>

{ <枚举元素表> };

（3）默认序号从 0 开始依次加 1；

（4）序号值不要求惟一；

（5）用枚举类型可定义枚举变量或枚举数组，枚举变量可进行赋值运算与比较运算。

（6）枚举变量不能用 cin 输入枚举元素值或序号值，只能用赋值运算符将枚举元素值赋给枚举变量。

（7）用 cout 可以输出枚举变量，但输出的是其序号而不是元素值。

8.2 定义一个描述学生成绩等级的枚举类型{A，B，C，D，E}，成绩等级与分数段的对应关系为 A：90~100 ，B：80~89 ，C：70~79 ，D：60~69 ，E：0~59。 在主函数中定义全班学生成绩枚举类型数组，输入全班学生的分数，并转换成等级赋给枚举数组元素。最后输出全班学生的成绩等级。

解：

```
# include <iostream.h>
# define N 5
enum grade
     {A,B,C,D,E};
void main(void)
{    grade g[N];
     int i,score;
     for(i=0;i<N;i++)
     {   cout<<"Input score["<<i<<"]=";
         cin>>score;
         if (score>100 || score<0 )
         {     cout<<"enter score error!"<<endl;
               i--;
         }
         else
         switch (score/10)
         {   case 9:;
             case 10: g[i]=A;break;
             case 8: g[i]=B;break;
             case 7: g[i]=C;break;
             case 6: g[i]=D;break;
             default:g[i]=E;
         }
```

```
        }
        cout<<"Student score:"<<endl;
        for(i=0;i<N;i++)
        {   switch (g[i])
            {   case   A : cout <<"G["<<i<<"]=A" ; break;
                case   B : cout <<"G["<<i<<"]=B" ; break;
                case   C : cout <<"G["<<i<<"]=C" ; break;
                case   D : cout <<"G["<<i<<"]=D" ; break;
                case   E : cout <<"G["<<i<<"]=E" ; break;
            }
            cout<<'\n';
        }
    }
```

8.3 从 A、B、C、D 四个字母中任取 3 个不同的字母，共有多少种取法？编写程序，输出所有取法中字母排列。

解：

```
# include <iostream.h>
enum letter
{A,B,C,D};
void show(letter c)
{   switch(c)
    {   case   A    : cout<<"A";break;
        case   B    : cout<<"B";break;
        case   C    : cout<<"C";break;
        case   D    : cout<<"D";break;
    }
    cout<<'\t';
}
void main(void)
{   letter c1,c2,c3;
    for(c1=A ;c1<=D;c1=letter(int (c1) +1))
      for(c2=A ;c2<=D;c2=letter(int (c2) +1))
        for(c3=A ;c3<=D;c3=letter(int (c3) +1))
          { if (c1!=c2 && c1!=c3 && c2!=c3)
              {   show(c1);
                  show(c2);
                  show(c3);
                  cout<<'\n';
              }
          }
}
```

8.4 定义学生档案结构体，描述一个学生的档案信息有：学号、姓名、性别、年龄、家庭地址、邮政编码、电话号码。

解：

```
struct student
{   int no ;
    char name[8];
    char sex;
    int age;
    char addr[20];
    int post;
    char photo[15];
};
```

8.5 定义学生成绩结构体，描述学生成绩信息有：学号、英语、数学、物理、总分、名次。

```
struct grade
{   int no ;
    float eng,math,phy;
    float total;
    int   order;
};
```

8.6 用习题 8.4 中的学生档案结构体定义一个学生的档案结构变量。用初始化方式输入学生档案内容，并输出学生档案。

解：

```
# include <iostream.h>
struct student
{   int no ;
    char name[8];
    char sex;
    int age;
    char addr[20];
    int post;
    char photo[15];
};
void main(void)
{   student s={101,"Zhou",'M',23,"wuxi",214073,"0510-5414727"};
    cout<<"学号："<<s.no<<'\n';
    cout<<"姓名："<<s.name<<'\n';
    cout<<"性别："<<s.sex<<'\n';
    cout<<"年龄："<<s.age<<'\n';
    cout<<"地址："<<s.addr<<'\n';
    cout<<"邮编："<<s.post<<'\n';
    cout<<"电话："<<s.photo<<'\n';
}
```

8.7 用习题 8.5 中的学生成绩结构体定义班级成绩结构体数组。编写四个函数分别用于：（1）输入全班学生的学号、英语、数学、物理成绩。

（2）计算每一个学生的总分。
（3）按总分降序排序，并给每个学生填入名次。
（4）输出全班学生的学号、三门课成绩、总分、名次。
在主函数中定义学生成绩数组，调用四个函数完成输入、计算总分、排名、输出工作。
解：

```
# include <iostream.h>
# include <iomanip.h>
# define N 5
struct grade
{   int no ;
    float eng,math,phy;
    float total;
    int   order;
};
void input(grade s[],int n)
{   int i;
    cout<<"Input Score:"<<endl;
    for(i=0;i<n;i++)
       cin>>s[i].no>>s[i].eng>>s[i].math>>s[i].phy;
}
void sum(grade s[],int n)
{   int i;
    for(i=0;i<n;i++)
       s[i].total=s[i].eng+s[i].math+s[i].phy;
}
void sort(grade s[],int n)
{   int i,j,k;
    grade temp;
    for(i=0;i<n-1;i++)
    {   k=i;
        for (j=i+1;j<n;j++)
           if (s[k].total<s[j].total) k=j;
        if (k!=i)
        { temp=s[i];s[i]=s[k];s[k]=temp;}
    }
    for(i=0;i<n;i++)
       s[i].order=i+1;
}
void output(grade s[],int n)
{ int i;
  cout<<setw(6)<<"No"<<setw(6)<<"Eng"<<setw(6)<<"Math"<<setw(6)<<"Phy"
      <<setw(6)<<"Total"<<setw(6)<<"Order"<<'\n';
  for(i=0;i<n;i++)
      cout<<setw(6)<<s[i].no<<setw(6)<<s[i].eng<<setw(6)<<s[i].math<<
      setw(6)<<s[i].phy<<setw(6)<<s[i].total<<setw(6)<<s[i].order<<'\n';
```

```
}
void main(void)
{   grade stu[N];
    input(stu,N);
    sum(stu,N);
    sort(stu,N);
    output(stu,N);
}
```

8.8 定义描述复数类型的结构体变量，编写减法函数 Sub()与乘法函数 Mul()分别完成复数的减法与乘法运算。在主函数中定义四个复数类型变量 c1、c2、c3、c4，输入 c1、c2 的复数值，调用 Sub()完成 c3=c1－c2 操作，调用 Mul( )完成 c4=c1*c2 操作。 最后输出 c3、c4 复数值。

解：

```
# include <iostream.h>
# include <iomanip.h>
struct complex
{   float r;
    float i;
};
complex Sub(complex c1,complex c2)
{   complex c;
    c.r=c1.r-c2.r;
    c.i=c1.i-c2.i;
    return c;
}
complex Mul(complex c1,complex c2)
{   complex c;
    c.r=c1.r*c2.r-c1.i*c2.i;
    c.i=c1.r*c2.i+c1.i*c2.r;
    return c;
}
void main(void)
{   complex c1,c2,c3,c4;
    cout<<"c1=";
    cin>>c1.r>>c1.i;
    cout<<"c2=";
    cin>>c2.r>>c2.i;
    c3=Sub(c1,c2);
    c4=Mul(c1,c2);
    cout<<"c3="<<c3.r<<"+("<<c3.i<<")i"<<'\n';
    cout<<"c4="<<c4.r<<"+("<<c4.i<<")i"<<'\n';
}
```

8.9 定义描述矩形的结构体类型，该结构体类型的数据成员为矩形的左上角坐标(x1,y1)与右下角(x2,y2)。编写函数 Area()计算出矩形的面积。在主函数中定义矩形结构体变量，输入矩形的左上

角坐标与右下角坐标，调用 Area()计算出矩形面积，并输出矩形面积。

解：

```
# include <iostream.h>
# include <math.h>
struct Rectangle
{   float x1,y1;
    float x2,y2;
};
float Area(Rectangle a)
{    float area;
     area=abs((a.x1-a.x2)*(a.y1-a.y2));
     return area;
}
void main(void)
{   Rectangle Rect;
    cout<<"Ipunt Rect(x1,y1):";
    cin>>Rect.x1>>Rect.y1;
    cout<<"Ipunt Rect(x2,y2):";
    cin>>Rect.x2>>Rect.y2;
    cout<<"Area="<<Area(Rect)<<endl;
}
```

8.10 什么是链表？链表的基本操作是什么？在链表的建立、插入、删除、输出函数中的关键语句是什么？

答：

（1）链表由若干个结构类型的结点用指针链接而成，每个结点由数据与指针两部分组成，其中指针用于链接下一个结点。

（2）链表的主要操作有链表的建立、插入、删除、输出等。

（3）建立链表的核心语句：

```
pt->next=pn;          //新结点加入到链尾
pt=pn;                //使 pt 指向新的链尾
```

（4）插入结点的核心语句

```
pn->next=pc;          //将插入点前一个结点的地址赋给新结点的指针
pa->next=pn;          //将新结点的地址赋给插入点后一个结点的指针
```

（5）删除结点的核心语句

```
pa->next=pc->next;    //要删除结点前一个结点的地址赋给其后一个结点的指针
delete pc;            //动态回收结点占用空间
```

8.11 建立一个描述学生成绩的无序链表，各结点内容如表 8.3 所示。计算出各学生的总分成绩，并输出链表中各学生结点的信息内容。最后删除链表回收链表占用空间。建立无序链表、计算总分、输出链表、删除链表各用一个函数实现。在主函数中调用四个函数完成上述操作。

表 8.3　学生成绩表

no(学号)	name[8] (姓名)	math(数学)	eng(英语)	score(总分成绩)
1001	Zhang	90	85	
1002	Wang	85	80	

1003	Li	75	70	
1004	Zhou	95	90	

解:

```
# include <iostream.h>
# include <string.h>
struct node
{   int no;
    char   name[8];
    float math,eng;
    float score;
    node   *next;
};
node * Create(void )
{    int no;                    //定义输入学生成绩的临时变量 score
     node *head,*pn,*pt;        //定义链表头指针、新结点指针、尾指针 head、pn、pt。
     head=0;                        //链表头指针赋 0，表示链表为空。
     cout<<"产生无序链表，请输入学号、姓名、数学与英语，以学号为-1 结束："<<endl;
     cin>>no;                       //输入学生学号
     while (no!= -1)                //成绩为-1 时结束输入
     {   pn= new node;          //动态分配新结点内存空间，并将结点地址赋给 pn。
         pn->no=no;                 //将学生成绩输入新结点
         cin>>pn->name ;        //学生姓名输入新结点
         cin>>pn->math ;        //数学成绩输入新结点
         cin>>pn->eng   ;       //英语成绩输入新结点
         if (head==0)           //若链表为空
         {   head=pn;               //则将新结点地址由 pn 赋给头指针 head 与尾指针 pt
             pt=pn;                 //使新结点加入到链首
         }
         else                       //否则链表非空
         {   pt->next=pn;           //将新结点地址由 pn 赋给链尾的 next 指针与尾指针 pt
             pt=pn;                 //使新结点加入到链尾
         }
         cin >>no;                  //输入学生成绩
     }
     pt->next=0;                //链尾指针变量赋 0
     return (head);             //返回链表的头指针
}

void Total(node *head)
{   node *p;
    p=head;
    while (p!=0 )
    {   p->score=p->math+p->eng;
        p=p->next;
```

```
    }
}

void Print(const node *head)
{   const node *p;
    p=head;

    while (p!=0 )   cout<<"输出链表中各结点值:"<<endl;
    {   cout<<p->no<<'\t'<<p->name<<'\t'<<p->math
        <<'\t'<<p->eng<<'\t'<<p->score<<'\t'<<endl;
        p=p->next;
    }
}
void Delchain(node * head)
{   node * p;
    p=head;              //链表头指针赋给 p
    while (head)         //当链表非空时删除结点
    { head=p->next;      //将链表下一个结点指针赋给 head
      delete p;          //删除链表第一个结点
      p=head;            //再将头指针赋给 p
   }
}

void main(void)                    //主函数
{   node * head;
    head=Create();                 //产生无序链表
    Total(head);
    Print(head);                   //输出无序链表
    Delchain(head);                //删除整个链表
}
```

8.12 在习题 8.11 的基础上，再编写能删除指定学号结点的函数，能在指定学号结点前插入新学生结点的函数。在主函数中输入要删除与插入结点的学号，并调用删除与插入函数删除与插入指定结点。插入新学生的信息在插入函数内输入。

解:

```
# include <iostream.h>
# include <string.h>
struct node
{   int no;
    char    name[8];
    float math,eng;
    float score;
    node    *next;
};
node * Create(void )
```

```
{    int no;                    //定义输入学生学号的临时变量 no
     node *head,*pn,*pt;        //定义链表头指针、新结点指针、尾指针 head、pn、pt。
     head=0;                    //链表头指针赋 0，表示链表为空。
     cout<<"产生无序链表，请输入学号、姓名、数学与英语，以学号为-1 结束："<<endl;
     cin>>no;                   //输入学生学号
     while (no!= -1)            //成绩为-1 时结束输入
     {   pn= new node;          //动态分配新结点内存空间，并将结点地址赋给 pn。
         pn->no=no;             //将学生成绩输入新结点
         cin>>pn->name ;        //学生姓名输入新结点
         cin>>pn->math ;        //数学成绩输入新结点
         cin>>pn->eng   ;       //英语成绩输入新结点
         if (head==0)           //若链表为空
         {   head=pn;           //则将新结点地址由 pn 赋给头指针 head 与尾指针 pt
             pt=pn;             //使新结点加入到链首
         }
         else                   //否则链表非空
         {   pt->next=pn;       //将新结点地址由 pn 赋给链尾的 next 指针与尾指针 pt
             pt=pn;             //使新结点加入到链尾
          }
         cin >>no;              //输入学生学号
     }
     pt->next=0;                //链尾指针变量赋 0
     return (head);             //返回链表的头指针
}

void Total(node *head)
{   node *p;
    p=head;
    while (p!=0 )
    {   p->score=p->math+p->eng;
        p=p->next;
    }
}

void Print(const node *head)
{   const node *p;
    p=head;
    cout<<"输出链表中各结点值:"<<endl;
    while (p!=0 )
    {   cout<<p->no<<'\t'<<p->name<<'\t'<<p->math
        <<'\t'<<p->eng<<'\t'<<p->score<<'\t'<<endl;
        p=p->next;
    }
}
void Delchain(node * head)
```

```
{   node * p;
    p=head;              //链表头指针赋给 p
    while (head)         //当链表非空时删除结点
    { head=p->next;      //将链表下一个结点指针赋给 head
      delete p;          //删除链表第一个结点
      p=head;            //再将头指针赋给 p
   }
}

node * Del( node *head, int no)
{   node *pc,*pa,*headtemp;
    headtemp=pc=pa=head;
    if (head=NULL)                          //链表为空的情况
    {   cout << "链表为空，无结点可删！\t";
        return   NULL;
    }
    if (pc->no==no)                         //  第一个结点为要删除结点的情况
    {   head=pc->next;                      //将第二个结点的地址赋给 head，
                                            //使首结点从链表中分离出来
        delete pc;                          //删除首结点
        cout<<"删除了一个结点！\n";
    }
    else                                    //第一个结点不是要删除的结点
    {   while (pc->no!=no && pc->next!=0 )  //查找要删除的结点
        {   pa=pc;                          //当前结点地址由 pc 赋给 pa
            pc=pc->next;                    //pc 指向下一个结点
        }
        if (pc==NULL)                       //若 pc 为空表示链表中无要删除的结点
        cout << "链表中没有要删除的结点！\n";
        else
        {   pa->next=pc->next;//将下结点地址赋给上结点，使删除结点从链表分离出来
            delete pc;                      //删除指定结点
            cout<<"删除一个结点！\n";
        }
        head=headtemp;
    }
    return head;                            //返回链表头指针
}

node * Insert(node * head, int no)
{   node *pc,*pa,*pn;                       //定义指向插入点前、后的指针 pc 与 pa
    pn=new node;
    cout<<"请输入学号、姓名、数学与英语："<<endl;
    cin>>pn->no>>pn->name>>pn->math>>pn->eng;
    pn->score=pn->math+pn->eng;
```

```
    pc=pa=head;
    if (head==0)                                  //若链表为空，则新结点插入到链表首
    {   head=pn;
        pn->next=0;
        return    head;
    }
    if (pc->no==no)                               // 第一个结点为要插入结点的情况
    {   pn->next=head;
        head=pn;
        return head;
    }
    else                                          //第一个结点不是要插入的结点
    {   while (pc->no!=no && pc->next!=0 )        //查找要插入的结点位置
        {   pa=pc;                                //当前结点地址由 pc 赋给 pa
            pc=pc->next;                          //pc 指向下一个结点
        }
        if (pc==NULL)                             //新结点插入到链尾
           {    pc->next=pn;
                pn->next=0;
           }
        else                                      //新结点插入到链表中间
           {    pn->next=pc;
                pa->next=pn;
           }
    }
 return head;                                     //返回链表头指针
}

void main(void)                                   //主函数
{   node * head;
    int no;
    head=Create();                                //产生无序链表
    Total(head);
    Print(head);                                  //输出无序链表
    cout<<"输入要删除结点上学生学号:\n";
    cin>>no;
    head=Del(head,no);                            //删除指定学号的结点
    Print(head);                                  //输出显示删除后的链表。
    cout<<"输入要插入结点学生学号:\n";
    cin>>no;
    head=Insert(head,no);                         //插入指定学号的结点
    Print(head);                                  //输出显示更新后的链表。
}
```

8.13 建立一个描述学生成绩的有序链表，各结点内容如表 8.3 所示，输入学生成绩时自动计

算总分，链表按总分升序排列。输出有序链表各结点内容，最后删除链表。

```
# include <iostream.h>
# include <string.h>
struct node
{   int no;
    char    name[8];
    float math,eng;
    float score;
    node    *next;
};

void Print(const node *head)
{   const node *p;
    p=head;
    cout<<"输出链表中各结点值:"<<endl;
    while (p!=0 )
    {   cout<<p->no<<'\t'<<p->name<<'\t'<<p->math
        <<'\t'<<p->eng<<'\t'<<p->score<<'\t'<<endl;
        p=p->next;
    }
}

void Delchain(node * head)
{   node * p;
    p=head;                 //链表头指针赋给 p
    while (head)            //当链表非空时删除结点
    { head=p->next;         //将链表下一个结点指针赋给 head
      delete p;             //删除链表第一个结点
      p=head;               //再将头指针赋给 p
   }
}

node * Insert(node * head, node *pn)
{   node *pc,*pa;                       //定义指向插入点前、后的指针 pc 与 pa
    pc=pa=head;
    if (head==0)                        //若链表为空，则新结点插入到链表首
    {   head=pn;
        pn->next=0;
        return    head;
    }
    if (pn->score<=head->score)         //若新结点成绩≥首结点成绩则新结点插在链首
    {   pn->next=head;
        head=pn;
        return head;
    }
```

```
    while (pc->next!=0 && pn->score>=pc->score) //若链表非空，则按成绩查找插入点
    {   pa=pc;                                  // pc、pa 移到插入点前、后结点处
        pc=pc->next;
    }
    if (pn->score >= pc->score)    //新结点插入到链尾
    {   pc->next=pn;
        pn->next=0;
    }
    else                           //新结点插入到链表中间
    {   pn->next=pc;
        pa->next=pn;
    }
 return head;                      //返回链表头指针
}

node *Create_sort( void)
{   node *pn,*head=0;              //定义指向新结点的指针变量 pn 及链表头指针 head
    int   no;                      //定义输入学生学号临时变量 no
    cout<<"产生一条有序链表，请输入数据，以-1 结束!\n";
    cin>>no;                       //输入学生学号
    while (no!= -1)                //学号不等于-1 则循环
    {   pn=new node;               //动态分配 node 类型结点空间，并将其地址赋给 pn
        pn->no=no;
        cin>>pn->name;             //输入学生姓名
        cin>>pn->math;
        cin>>pn->eng;
        pn->score=pn->math+pn->eng;
        head=Insert(head,pn);      //调用结点插入函数，将新结点按成绩降序插入链表
        cin>>no;                   //输入学生学号
    }
    return head;                   //返回链表头指针
}

void main(void)                    //主函数
{   node * head;
    head=Create_sort();            //产生一个有序链表
    Print (head);                  //输出显示有序链表
    Delchain(head);                //删除整个链表
 }
```

习题 9

9.1 什么叫类？什么叫对象？举三个可用类描述的事例。

答：

类由描述某类事物的数据(数据成员)及处理数据的函数(成员函数)组成的导出数据类型，用类定义的变量称为对象。职工工资、复数、圆柱体均可用类来描述。

职工工资：

描述职工工资的数据成员有：工号、姓名、性别、基本工资、津贴工资、公积金、实发工资等；

处理职工工资的成员函数有：输入职工信息、输出职工信息、显示职工工资、统计职工工资等成员函数。

复数：

描述复数的数据成员有：实部、虚部

处理复数的成员函数有：输入复数、输出复数、显示复数、复数运算等成员函数。

圆柱体：

描述圆柱体的数据成员有：半径、高；

处理圆柱体的成员函数有：输入半径与高、显示半径与高、计算圆的体积等成员函数。

9.2 叙述公有、私有、保护成员在类内、外的访问权限。

答：

成员的访问权限：

- public（公有成员）：公有数据成员允许类内或类外函数访问，公有成员函数允许在类内或类外调用。
- private（私有成员）：私有数据成员只允许类内函数访问，私有成员函数只允许在类内调用。
- protected(保护成员)：保护数据成员只允许类或其子类中函数访问，保护成员函数允许在类内或其子类中调用。

9.3 构造函数的作用是什么？构造函数的名称与类型有何特点？何时调用构造函数？

答：

构造函数用于对象数据成员的初始化。

构造函数名必须与类名相同，且无返回类型。

用类定义对象时调用构造函数对数据成员进行初始化。

9.4 构造函数可以分为哪三类？其形参有何区别？分别初始化何种对象？

答：

构造函数可以分为：

（1）有参构造函数：形参可以是任意类型的变量，用于初始化有实参的对象；

（2）无参构造函数：没有形参，用于初始化无实参的对象；

（3）拷贝构造函数：形参必须为类的对象的引用, 初始化时用于将一个已存在对象的数据拷贝到新建对象中，即用于初始化实参为已存在对象的对象。

9.5 叙述定义对象时，调用构造函数的过程及参数的传送过程？

答：

用类定义一个对象时，系统先为其分配内存空间，然后调用构造函数对数据成员进行初始化。系统调用构造函数前，首先判断定义对象是否有实参。

若无实参，则调用无参构造函数，对新建对象的数据成员进行初始化。

若有实参，且实参不是对象，则调用带参构造函数，先将实参传送给形参，然后通过构造函数体实现对新建对象数据成员的初始化。

若实参为已存在的对象，则调用拷贝构造函数，由于拷贝构造函数的形参为对象的引用，所以实参对象与形参对象占用相同的内存空间，具有相同的数据内容，因此通过拷贝构造函数可实现新建对象数据成员的初始化。

9.6 析构函数的作用是什么？析构函数的名称与类型有何特点？何时调用析构函数？

答：

（1）析构函数的作用是回收对象所占用的内存空间。

（2）析构函数的名称必须与类名相同，且析构函数名前必须加"~"。析构函数无参数无返回类型，析构函数不允许重载，即析构函数是唯一的；

（3）当对象结束其生命期，系统调用析构函数，回收对象所占用的内存空间。

9.7 用 new 运算符动态建立对象与用 delete 运算符撤消动态建立对象时，系统如何调用构造函数与析构函数？

答：

当用 new 运算符动态建立对象时，系统

（1）为对象分配内存空间

（2）调用构造函数初始化对象的数据成员

先将实参传送给形参，然后由构造函数体完成对数据成员的初始化工作。

（3）将对象起始地址返回给指针变量

用 new 动态分配的内存空间必须用 delete 回收，在执行 delete 语句时，将调用析构函数回收对象占用的内存空间。

9.8 什么是 this 指针？

答：

用类定义一个对象时，系统会自动建立一个指向该对象的指针变量，该指针变量称 this 指针，this 指针变量指向所定义的对象。

9.9 定义一个学生成绩类 Score，描述学生成绩的私有数据成员为学号(No)、姓名（Name[8]）、数学(Math)、物理(Phi)、数据结构(Data)、总分(Sum)。定义能输入学生成绩的公有成员函数 Input()，能计算学生总分的成员函数 Sum()，能显示学生成绩的成员函数 Show()。在主函数中用 Score 类定义学生成绩对象数组 s[5]。用 Input()输入学生成绩，用 Sum()计算每个学生的总分，最后用 Show()显示每个学生的成绩。

解：

```
# include <iostream.h>
# include <string.h>
class Score
{    private:
        int No;
        char Name[8];
        float Math,Phi,Data,Sum;
     public:
       void Input(int no,char name[],float math,float phi,float data)
```

```
    {   No=no;
        strcpy(Name,name);
        Math=math;
        Phi=phi;
        Data=data;
    }
    void Output(int &no,char name[],float &math,float &phi,float &data,float &sum)
    {   no=No;
        strcpy(name,Name);
        math=Math;
        phi=Phi;
        data=Data;
        sum=Sum;
    }
    void Summation(void)
    {   Sum=Math+Phi+Data;}

     void Show()
     {    cout<<No<<'\t'<<Name<<'\t'<<Math<<'\t';
          cout<<Phi<<'\t'<<Data<<'\t'<<Sum<<'\n';
     }
};

void main(void)
{   int i,no;
    char name[8];
    float math,phi,data;
    Score   s[5];
    cout<<"Input 5 student data"<<'\n';
    for (i=0;i<5;i++)
    {
        cin>>no>>name>>math>>phi>>data;
        s[i].Input(no,name,math,phi,data);
        s[i].Summation();
    }
    cout<<"学号    姓名    数学    物理    数据结构    总分\n";
    for (i=0;i<5;i++)
        s[i].Show();
}
```

9.10 定义一个复数类 Complex，复数的实部 Real 与虚部 Image 定义为私有数据成员。用复数类定义两个复数对象 c1、c2，用　　构造函数将 c1 初始化为

c1=10+20i ，将 c2 初始化为 c2=0+0i。

然后将 c1 的值赋给 c2。最后用公有成员函数 Dispaly()显示复数 c1 与 c2 的内容。

解：

```
# include <iostream.h>
class Complex
{    private:
        float Real,Image;
     public:
        Complex(float r,float i)        //定义有参构造函数
        {   Real=r;
            Image=i;
        }
        Complex(Complex &c)         //定义拷贝构造函数
        {   Real=c.Real;
            Image=c.Image;
        }
       Complex()                    //定义无参构造函数
        {   Real=0;
            Image=0;
        }
        void Display()
          {    cout<<Real<<"+"<<Image<<"i"<<'\n';}
} ;
void main(void)
{   Complex c1(10,20),c2;
    c1.Display();
    c2.Display();
    c2=c1;
    c2.Display();
}
```

9.11 定义一个矩形类 Rectangle，矩形的左上角（Left,Top）与右下角坐标(Right,Bottom)定义为保护数据成员。用公有成员函数 Diagonal()计算出矩形对角线的长度，公有成员函数 Show()显示矩形左上角与右下角坐标。在主函数中用矩形类定义对象 r1 与 r2，r1 的初值为（10，10，20，20）。r2 右下角坐标的初值用拷贝构造函数将 r1 右下角坐标值拷贝到 r2 中，左上角坐标初值为（0，0）。显示矩形 r1、r2 的左上角与右下角坐标及对角线长度。

解：

```
# include <iostream.h>
# include <math.h>
class Rectangle
{    protected:
        float Left,Top;
        float Right,Bottom;
     public:
        Rectangle(float l,float t, float r,float b)
        {   Left=l;Top=t;
            Right=r;Bottom=b;
```

```
        }
        Rectangle(Rectangle & R)
        {   Left=0;Top=0;
            Right=R.Right;Bottom=R.Bottom;
        }
        float Diagonal()
        {   return sqrt((Left-Right)* (Left-Right)+(Top-Bottom)*(Top-Bottom));}
        void Display()
          {     cout<<"(Left,Top)=("<<Left<<","<<Top<<")"<<'\n';
                cout<<"(Right,Bottom)=("<<Right<<","<<Bottom<<")"<<'\n';
                cout<<" Diagonal="<< Diagonal()<<'\n';
        }
} ;
void main(void)
{   Rectangle r1(10,10,20,20),r2(r1);
    r1.Display();
    r2.Display();
}
```

9.12 定义一个描述圆柱体的类 Cylinder，定义圆柱体的底面半径 Radius 与高 High 为私有数据成员。用公有成员函数 Volume()计算出圆柱体的体积，公有成员函数 Show()显示圆柱体的半径、高与体积。在主函数中用 new 运算符动态建立圆柱体对象，初值为（10，10）。然后调用 Show()显示圆柱体的半径、高与体积。最后用 delete 运算符回收为圆柱体动态分配的存储空间。

解：

```
# include <iostream.h>
class Cylinder
{   private:
       float Radius,High;
    public:
       Cylinder(float r,float h)
       {   Radius=r;
           High=h;
       }
       float Volume()
       {   return Radius*Radius*3.1415*High;}
        void Show()
        { cout<<"Radius="<<Radius<<'\t'<<"High="<<High<<'\n';
          cout<<"Volume="<<Volume()<<'\n';
        }
};
void main(void)
{   Cylinder *pc=new Cylinder(10,10);
    pc->Volume();
    pc->Show();
    delete pc;
```

```
}
```

9.13 先定义一个能描述平面上一条直线的类 Beeline，其私有数据成员为直线两个端点的坐标(X1,Y1,X2,Y2)。在类中定义形参缺省值为 0 的构造函数，及计算直线长度的公有成员函数 Length()，显示直线两个端点坐标的公有成员函数 Show()。然后再定义一个能描述平面上三角形的类 Triangle，其数据成员为用 Beeline 定义的对象 line1、line2、line3 与三角形三条边长 l1、l2、l3。在类中定义的构造函数要能对对象成员与边长进行初始化。再定义计算三角形面积的函数 Area()，及显示三条边端点坐标及面积的函数 Print()，Print()函数中可调用 Show()函数显示三条边两端点坐标。

在主函数中定义三角形对象 tri(10,10,20,10,20,20)，调用 Print()函数显示三角形三条边端点坐标及面积。

解：

```
# include <iostream.h>
# include <math.h>
class Beeline
{   private:
       float X1,Y1,X2,Y2;
    public:
       Beeline(float x1=0,float y1=0,float x2=0,float y2=0)
       {   X1=x1;Y1=y1;X2=x2;Y2=y2;}
       float Length()
       {   return sqrt((X1-X2)*(X1-X2)+(Y1-Y2)*(Y1-Y2));}
       void Show()
       {   cout<<"("<<X1<<","<<Y1<<")"<<'\t';
           cout<<"("<<X2<<","<<Y2<<")"<<'\n';
       }
};
class Triangle
{   private:
       float L1,L2,L3;
       Beeline Line1,Line2,Line3;
    public:
       Triangle(float x1,float y1,float x2,float y2,float x3,float y3):Line1(x1,y1,x2,y2),
       Line2(x2,y2,x3,y3),Line3(x3,y3,x1,y1)
       {   L1=Line1.Length();
           L2=Line2.Length();
           L3=Line3.Length();
       }
       float Area()
       {   float s;
           s=(L1+L2+L3)/2;
           return sqrt(s*(s-L1)*(s-L2)*(s-L3));
       }
```

```
    void Print()
    {   Line1.Show();
        Line2.Show();
        Line3.Show();
        cout<<"Area="<<Area()<<'\n' ;
    }
};
void main(void)
{   Triangle tri(10,10,20,10,20,20) ;
    tri.Print();
}
```

9.14 写出下列程序的运行结果

```
# include <iostream.h>
class Point
{   private:
        float X,Y;
    public:
        void SetXY(float x,float y)
        {   X=x;
            Y=y;
        }
        unsigned Address(void)
        {   return (unsigned) this;}
        void Show(void)
        {   cout<<"X="<<X<<'\t'<<"Y="<<Y<<endl;}
 };
void main(void)
{   Point a,b;
    a.SetXY(10.5,10.5);
    b.SetXY(1.2,5.4);
    a.Show( );
    b.Show( );
    cout<<"a address:"<<a.Address( )<<endl;
    cout<<"b address:"<<b.Address( )<<endl;
}
```

答：

```
X=10.5   Y=10.5
X=1.2    Y=5.4
a   address:6749680
b   address:6749672
```

9.15 写出下列程序的执行结果。

```
# include <iostream.h>
```

```
class Strucf
{   private:
        float a,b,max;
    public:
        Strucf(int x,int y)
        {   a=++x;
            b=y++;
            if (a>b) max=a;
            else   max=b;
            cout<<"Call Strucf(int ,int)"<<endl;
        }
        Strucf(float x)
        {   a=x;
            b=10;
            if (a>b) max=a;
            else   max=b;
            cout<<"Call Strucf(float)"<<endl;
        }
        Strucf()
        {   max=a=b=0;}
        void Print()
        {   cout<<"a="<<a<<'\t'<<"b="<<b<<endl;
             cout<<"max="<<max<<endl;
        }
        ~Strucf()
        {   cout<<"Call ~Strucf()"<<endl;}
};
void main (void)
{    Strucf s1(2,3),s2(2.75),s3;
     s1.Print();
     s2.Print();
     s3.Print();
}
```

解：

```
Call Strucf(int, int)
Call Strucf(float)
a=3     b=3
max=3
a=2.75   b=10
max=10
a=0     b=0
max=0
Call ~Strucf()
Call ~Strucf()
Call ~Strucf()
```

9.16 阅读下列程序，说出初始化对象成员 c1 过程，并写出程序运行结果。

```
# include <iostream.h>
class A
{   private:
        int X,Y;
    public:
        A(int a,int b)
        {   X=a;Y=b;}
        void Show( )
        {   cout<<"X="<<X<<'\t'<<"Y="<<Y<<'\n';}
 };
class B
{   private:
        int Length,Width;
    public:
        B(int a,int b)
        {   Length=a; Width=b;}
        void Show( )
        {   cout<<"Length="<<Length<<'\t'<<"Width="<<Width<<endl;}
};
class C
{   private:
        int R,High;
        A a1;
        B b1;
    public:
        C(int a,int b,int c,int d,int e,int f):a1(e,f),b1(c,d)
        {   R=a;High=b;}
        void Show( )
        {   cout<<"R="<<R<<'\t'<<"High="<<High<<endl;
            a1.Show( );
            b1.Show( );
        }
};
void main(void)
{   C c1(25,35,45,55,65,100);
    c1.Show( );
}
```

解：

R=25　　High=35

X=65　　Y=100

Legnth=45　Width=55

习题 10

10.1 叙述继承与派生的定义。什么叫单一派生？什么叫多重派生？

答：

从已有类出发建立新的类，使新类部分或全部地继承已有类的成员称为继承。通过继承已有的一个或多个类产生一个新类称为派生。

派生类的定义格式如下：

class <派生类名>:<access><基类名 1>〔,…,<access><基类名 n>〕{派生类体}；

当 n=1 时为单一继承，当 n>1 时为多重继承。

10.2 叙述基类成员经公有派生后，在派生类中访问权限的变化。叙述基类成员经私有派生后，在派生类中访问权限的变化。

答：

经公有派生或私有派生后，基类成员在派生类中访问权限将发生变化，如表所示。

基类成员访问权限	公有派生后的访问权限	私有派生后的访问权限
public	public	private
private	不可直接访问	不可直接访问
protected	private	private

10.3 叙述用派生类定义对象时，构造函数的调用执行过程。

答：

派生类构造函数说明部分由两部分组成：派生类构造函数(形参表)：基类构造函数(实参表)。按先基类后派生类的原则调用构造函数。具体调用过程如下：

程序执行到用派生类定义对象的语句时，首先将对象后括号内的实参传送给派生类构造函数中的形参，并将所得到形参值传送给基类构造函数的实参。然后按从左到右的顺序依次调用冒号后基类构造函数，将调用函数中的实参传送给基类构造函数的形参，并执行基类构造函数完成基类数据成员的初始化工作。最后再调用派生类构造函数，完成派生类数据成员的初始化工作。

10.4 叙述冲突、支配规则、赋值兼容性的定义。

答：

（1）冲突：派生类使用基类中同名成员时出现不唯一称为冲突。

冲突的解决方法：

<基类名>::<成员名>

（2）支配规则：使用派生类中与基类同名成员时，派生类成员优于基类同名成员的规则称为支配规则。

（3）赋值兼容性：派生类对象可赋值给基类对象，基类对象不能赋给派生类对象称为赋值的兼容性。

10.5 定义描述职工档案的类 Archives，数据成员为职工号(No)、姓名(Name[8])、性别(Sex)、年龄(Age)。成员函数有：构造函数、显示职工信息的函数 Show()。再由职工档案类派生出职工工资类 Laborage，在职工工资类 Laborage 中新增数据成员：应发工资(SSalary)、社保金(Security)、实发工资(Fsalary)，其成员函数有：构造函数、显示职工档案及工资的函数。在主函数中用 Laborage 类定义职工对象 lab，并赋初始值，然后显示职工档案与工资。

解：

```
# include <iostream.h>
# include <string.h>
class Archives
{   private:
        int No;
        char Name[8];
        char Sex;
        int Age;
    public:
        Archives(int n,char name[],char s,int a)
        {   No=n;
            strcpy(Name,name);
            Sex=s;
            Age=a;
        }
        void Show(void)
        {   cout<<"No="<<No<<'\t'<<"Name="<<Name<<'\t'
            <<"Sex="<<Sex<<'\t'<<"Age="<<Age<<'\n';
        }
};
class Laborage:public Archives
{   private:
        float SSalary,Security,Fsalary;
    public:
         Laborage(int n,char name[],char s,int a,float ss,float se,float fs):
          Archives(n,name,s,a)
          {   SSalary=ss;
              Security=se;
              Fsalary=fs;
          }
          void Display(void)
          {   Show();
              cout<<"SSalary="<<SSalary<<'\t'<<"Security="<<Security
                    <<'\t'<<"Fsalary="<<Fsalary<<'\n';
          }
};
void main(void)
{   Laborage lab(1001,"Zhou",'M',52,2000,200,1800);
    lab.Display();
}
```

10.6 定义描述矩形的类 Rectangle，其数据成员为矩形的长(Length)与宽(Width)。成员函数为计算矩形面积的函数 Area()与构造函数。再由矩形类派生出长方体类 Cuboid，其数据成员为长方体的高 High 与体积 Volume。成员函数为：构造函数、计算体积的函数 Vol()、显示长、宽、高与体积的函数 Show()。主函数中用长方体类定义长方体对象 cub，并赋初始值(10,20,30)，最后显示

长方体的长、宽、高与体积。

解：

```
# include <iostream.h>
class Rectangle
{ protected:
      float Length,Width;
   public:
      Rectangle(float l,float w)
      {   Length=l;
          Width=w;
      }
      float Area(void)
      {   return Length*Width;}
};
class Cuboid:public Rectangle
{   private:
       float High,Volume;
    public:
       Cuboid(float l,float w,float h):Rectangle(l,w)
       {     High=h;}
       void Vol(void)
       {     Volume=Area()*High;}
       void Show(void)
       { cout<<"Length="<<Length<<'\t'<<"Width="<<Width<<'\t'
          <<"High="<<High<<'\n';
          Vol();
          cout<<"Volume="<<Volume<<'\n';
       }
};
void main (void)
{   Cuboid cub(10,20,30);
    cub.Show();
}
```

10.7 定义描述矩形的类 Rectangle，其数据成员为矩形的长(Length)与宽(Width)。成员函数为计算矩形面积的函数 Area()与构造函数。再定义描述长方体高的类 High，其数据成员为长方体高度 H，其成员函数为构造函数。再由矩形类与高类多重派生出长方体类 Cuboid，其数据成员为体积 Volume。成员函数为：构造函数、计算体积的函数 Vol()、显示长、宽、高与体积的函数 Show()。主函数中用长方体类定义长方体对象 cub，并赋初始值(10,20,30)，最后显示长方体的长、宽、高与体积。

解：

```
# include <iostream.h>
class Rectangle
{ protected:
      float Length,Width;
```

```
    public:
      Rectangle(float l,float w)
      {   Length=l;
          Width=w;
      }
      float Area(void)
      {   return Length*Width;}
};
class High
{   protected:
        float H;
    public:
        High(float h)
        {   H=h;}
};
class Cuboid:public Rectangle,public High
{   private:
        float Volume;
    public:
      Cuboid(float l,float w,float h):Rectangle(l,w),High(h)
      {     Vol();}
      void Vol(void)
      {     Volume=Area()*H;}
      void Show(void)
      { cout<<"Length="<<Length<<'\t'<<"Width="<<Width<<'\t'
        <<"High="<<H<<'\n';
        cout<<"Volume="<<Volume<<'\n';
      }
};
void main (void)
{   Cuboid cub(10,20,30);
    cub.Show();
}
```

10.8 将定义直角坐标系上一个点的类作为基类，派生出描述一条直线的类（两点坐标确定一直线），再派生出三角形类（三点坐标确定一个三角形）。要求成员函数能求出两个点间的距离、三角形的周长和面积。设计一个测试程序，并构成完整的程序。

解：

```
# include <iostream.h>
# include <math.h>
class Point
{   protected:
        float X1,Y1;
    public:
```

```
        Point(float x,float y)
        {   X1=x; Y1=y;}
};
class Beeline: public Point
{   protected:
        float X2,Y2;
    public:
        Beeline(float x1=0,float y1=0,float x2=0,float y2=0):Point(x1,y1)
        {   X2=x2;Y2=y2;}
};
class Triangle:public Beeline
{   private:
        float X3,Y3,L1,L2,L3;
    public:
        Triangle(float x1,float y1,float x2,float y2,float x3,float y3):Beeline(x1,y1,x2,y2)
        {   X3=x3;
            Y3=y3;
            L1=Length(X1,Y1,X2,Y2);
            L2=Length(X2,Y2,X3,Y3);
            L3=Length(X3,Y3,X1,Y1);
        }
        float Length(float x1,float y1,float x2,float y2)
        {   return sqrt((x1-x2)*(x1-x2)+(y1-y2)*(y1-y2));
        }
        float Perimeter(void)
        {   return L1+L2+L3;
        }
        float Area()
        {   float s;
            s=(L1+L2+L3)/2;
            return sqrt(s*(s-L1)*(s-L2)*(s-L3));
        }
        void Show(void)
        {   cout<<"(X1,Y1)=("<<X1<<","<<Y1<<")"<<'\n';
            cout<<"(X2,Y2)=("<<X2<<","<<Y2<<")"<<'\n';
            cout<<"(X3,Y3)=("<<X3<<","<<Y3<<")"<<'\n';
            cout<<"Length1="<<L1<<'\n';
            cout<<"Length2="<<L2<<'\n';
            cout<<"Length3="<<L3<<'\n';
        }
};
void main(void)
{   Triangle tri(10,10,20,10,20,20) ;
    tri.Show();
    cout<<"Perimeter="<<tri.Perimeter()<<'\n' ;
```

```
    cout<<"Area="<<tri.Area()<<'\n' ;

}
```

10.9 阅读下列程序，写出执行结果：

```
# include <iostream.h>
# include <string.h>
class Father
{   protected:
        char Face[20],Eye[20],Mouth[20];
        float High;
    public:
        Father(char *face,char *eye,char *mouth)
        {   strcpy(Face,face);
            strcpy(Eye,eye);
            strcpy(Mouth,mouth);
        }
        void Print()
        {   cout<<"Father:"<<'\t'<<'\t'<<Face<<'\t'<<Eye<<'\t'<<Mouth<<endl;}
};
class Son:public Father
{   private:
        float High;
    public:
        Son(char *face,char *eye,char *mouth,float high):Father(face,eye,mouth)
        {   High=high;}
        void Print()
        {   cout<<"Son:"<<'\t'<<'\t'<<Face<<'\t'<<Eye<<'\t'<<Mouth<<'\t'<<High<<endl;}
};
class Daughter:public Father
{   private:
        float High;
    public:
        Daughter(char *face,char *eye,char *mouth,float high):Father(face,eye,mouth)
        {   High=high;}
        void Print()
        {   cout<<"Daugther:"<<'\t'<<Face<<'\t'<<Eye<<'\t'<<Mouth<<'\t'<<High<<endl;}
};
void main(void)
{   Father ba("square face","big eye","samll mouth");
    Father *pba=&ba;
    pba->Print();
    Son    son("square face","big eye","samll mouth",1.75);
    son.Print();
    Daughter    daug("square face","big eye","samll mouth",1.60);
```

```
    daug.Print();
}
```

解：

Father：　square face big eye small mouth

Son：　　square face big eye small mouth　1.75

Daugther：square face big eye small mouth 1.6

10.10 阅读下列程序，写出执行结果：

```
# include <iostream.h>
class A
{   private:
       int i,j;
    public:
        A(int a,int b)
        {i=a;j=b;}
        void add(int x,int y)
        {i+=x;j+=y;}
        void print()
        { cout<<"i="<<i<<'\t'<<"j="<<j<<endl;}
 };
  class B:public A
  {   private:
         int x,y;
      public:
         B(int a,int b,int c,int d) :A(a,b)
         {   x=c;y=d;}
         void ad(int a,int b)
         {   x+=a;y+=b;add(-a,-b);}
         void p() {A::print();}
         void print()
         { cout<<"x="<<x<<'\t'<<"y="<<y<<endl;}
    };
    void main(void)
{   A a(100,200);
      a.print();
      B b(200,300,400,500);
      b.ad(50,60);
      b.A::print();
      b.print();
      b.p();
  }
```

解：

i=100　　j=200

i=150　j=240
x=450　y=560
i=150　j=240

10.11 静态数据成员如何定义？如何初始化？如何引用？静态成员函数如何定义？如何引用？

答：

（1）定义静态数据成员必须分二步：在类内作引用性说明，在类外作定义性说明。

类内作引用性说明的格式：static <类型> 数据成员；

类外作定义性说明的格式：<类型> <类名>::<静态数据成员>〔=初值〕；

（2）静态数据成员的初始化是通过类外定义性说明时，给静态数据成员赋初值实现的。

（3）静态数据成员的引用格式：<类名>::<静态数据成员>

（4）静态成员函数定义方法：

类内定义：static <类型> <静态成员函数>(形参) {函数体}

类外定义：<类型> <类名>::<静态成员函数>(形参) {函数体}

（5）引用方式：<类名>::<静态成员函数>(实参)；

10.12 静态数据成员与非静态数据成员有何本质区别？使用静态成员函数与使用非静态成员函数有何不同之处？

答：

（1）同类不同对象的静态数据成员占用相同的内存空间，其使用方式只与类有关即：<类名>::<静态数据成员>，而与对象无关。

同类不同对象的动态数据成员占用不同的内存空间，所以其使用方式与类的对象有关即：<对象名>.<静态数据成员>。

静态数据成员必须在类内作引用性说明，而在类外定义性说明并分配内存空间，赋初值（默认初值为 0）。

动态数据成员与对象有关，只有定义对象后才能为对象的动态数据成员分配内存空间。

（2）静态成员函数只能使用类内的静态数据成员或静态成员函数，不能使用非静态成员。动态成员函数可使用本类中所有的数据成员。

10.13 定义描述小汽车的类 Car，将汽车生产厂(Factory)定义为静态数据成员，汽车颜色（Color）、重量(Weight)定义为私有数据成员。用构造函数对私有数据成员进行初始化，再定义能显示汽车信息成员函数 Show()。汽车生产厂 Factory 初始化为”别克”。在主函数中，定义二辆汽车 car1(”red”，1000kg)、car2(“black”,800kg)。输出这二辆汽车的生产厂、颜色与重量。

解:

```
# include <iostream.h>
# include <string.h>
class Car
{   private:
      char Color[10];
      float Weight;
      static char Factory[40];        //引用性说明时，并不为 Factory 分配内存空间
    public:
      Car( char cl[],float wt)
      {   strcpy(Factory,"别克");
```

```
        strcpy(Color,cl);
        Weight=wt;
    }
    Car()
    {   strcpy(Factory,"别克");
        strcpy(Color,"red");
        Weight=1000;
     }

    void Show()
    {   cout<<"生产厂："<<Factory<<'\t';
        cout<<"颜色："<<Color<<'\t';
        cout<<"重量："<<Weight<<" kg\n";
    }
};
char   Car::Factory[40]="别克";                    //定义性说明分配内存
void main (void)
{   Car c1("red",1000),c2("black",800);
    c1.Show();
    c2.Show();
}
```

习题 11

11.1 为何要使用友元？使用友元有哪些利与蔽？友元有哪三类？如何定义？

答：

（1）由于类的封装性与安全性，类的私有数据成员只能在类中直接使用，而在类外必须要通过公有接口函数才能使用，这给用户带来许多不便。使用友元能使用户在类外直接使用类的私有成员或保护成员。使用友元虽然方便了用户，但却破坏了数据的安全性，因此友元的使用要适度。

（2）友元有三类：普通函数、成员函数、类，这三类友元的定义方式如下：

将普通函数定义为某类的友元函数的方法是，在该类中增加语句：friend <类型><普通函数名>(形参)；

类 C 的成员函数定义为类 D 的友元函数的方法见 11.1.2 节。

类 C 定义为类 D 的友元的方法是在类 D 中增加语句：freind class C;

11.2 定义学生成绩类 Score，其私有数据成员有学号、姓名、物理、数学、外语、平均成绩。再定义一个能计算学生平均成绩的普通函数 Average()，并将该普通函数定义为 Score 类友元函数。在主函数中定义学生成绩对象，通过构造函数输入除平均成绩外的其它信息，然后调用 Average() 函数计算平均成绩，并输出学生成绩的所有信息。

答：

```
# include <iostream.h>
# include <string.h>
class Score
{    private:
          int No;
          char Name[8];
          float Math,Phi,Eng,Ave;
     public:
       Score(int no,char name[],float math,float phi,float eng)
       {   No=no;
           strcpy(Name,name);
           Math=math;
           Phi=phi;
           Eng=eng;
       }
       friend void Average(Score &s);
       void Show(void)
       {    cout<<No<<'\t'<<Name<<'\t'<<Math<<'\t';
            cout<<Phi<<'\t'<<Eng<<'\t'<<Ave<<'\n';
       }
};
void Average(Score &s)
{   s.Ave=(s.Math+s.Phi+s.Eng)/3;
}

void main(void)
{    Score   s(1001,"Zhou",90,80,70);
```

```
    Average(s);
    cout<<"学号    姓名    数学    物理    英语    平均分\n";
    s.Show();
}
```

11.3 定义描述圆的类 Circle，其数据成员为圆心坐标(X,Y)与半径 R。再定义一个描述圆柱体的类 Cylinder，其私有数据成员为圆柱体的高 H。定义计算圆柱体体积的成员函数 Volume()，并将 Volume()定义为圆类 Circle 的友元函数，该函数使用圆类对象的半径 R 来计算圆柱体的体积。在主函数中定义圆的对象 ci，圆心坐标为(12,15)，半径为 10。再定义圆柱体对象 cy，其底面半径与圆半径相同，高为 10。调用 Volume()函数计算圆柱体的体积，并显示体积值。

解：

```
# include <iostream.h>
class Circle;
class Cylinder
{   private:
        float H;
    public:
        Cylinder(float h)   {   H=h;}
        float Volume(Circle &);
};
class Circle
{   private:
        float X,Y,R;
    public:
        Circle(float x,float y,float r)
        {   X=x;Y=y;R=r;}
        friend float Cylinder::Volume(Circle &);
 };
 float Cylinder::Volume(Circle &c)
 {   return c.R*c.R*H*3.1415;}
 void main(void)
  {   Circle    ci(10,10,10);
      Cylinder    cy(10);
      cout<<"圆柱体体积="<<cy.Volume(ci)<<endl;
  }
```

在 11.3 题中，将圆柱体类定义为圆类的友元，求圆柱体体积。

解：

```
# include <iostream.h>
class Circle;
class Cylinder
{   private:
        float H;
    public:
        Cylinder(float h)   {   H=h;}
```

```
        float Volume(Circle &);
};
class Circle
{   private:
        float X,Y,R;
    public:
        Circle(float x,float y,float r)
        {   X=x;Y=y;R=r;}
        friend class Cylinder;
 };
 float Cylinder::Volume(Circle &c)
 {   return c.R*c.R*H*3.1415;}
 void main(void)
  {   Circle   ci(10,10,10);
      Cylinder   cy(10);
      cout<<"圆柱体体积="<<cy.Volume(ci)<<endl;
  }
```

11.4 何为运算符重载？如何实现运算符重载？运算符重载函数通常为类的哪二种函数？

解：

（1）运算符重载是指用同一运算符完成不同的运算操作。

（2）运算符重载是通过运算符重载函数来实现的。

（3）运算符重载函数通常为类的成员函数或友元函数。

11.5 设 c1、c2、c3 为复数对象，分别就“+”运算符重载函数为成员函数与友元函数两种情况，写出编译器对表达式 c1=c2+c3 的解释结果及对重载函数的调用过程。

答：

（1）运算符重载函数为成员函数

编译器对表达式 c1=c2+c3 的解释结果为：c1=c2.operator +(c3);

调用过程：先由对象 c2 调用重载函数 operator +(c3)，在函数体内，定义复数对象 c，

执行语句：c.Real=Real+c3.Real; c.Image=Image+c3.Image;

最后用 return c; 将复数和返回给 c1。

（2）运算符重载函数为友元函数

编译器对表达式 c1=c2+c3 的解释结果为：c1=operator +(c2,c3);

调用过程：先调用重载函数 operator +(c2,c3)，在函数体内，定义复数对象 c，

执行语句：c.Real=c2.Real+c3.Real; c.Image=c2.Image+c3.Image;

最后用 return c; 将复数和返回给 c1。

11.6 定义一个复数类，重载“+=”运算符，使这个运算符能直接完成复数的“+=”运算。分别用成员函数与友元函数编写运算符重载函数。在主函数中定义复数对象 c1(10,20)、c2(15,30)，进行 c2+=c1 的复数运算，并输出 c1、c2 的复数值。

解：

（1）成员函数

```
# include <iostream.h>
```

```
class Complex
{   private:
        float Real,Image;
    public:
        Complex(float r=0,float i=0)
        {   Real=r;Image=i;}
        void Show(int i)
        {   cout<<"c"<<i<<"=" <<Real<<"+"<<Image<<"i"<<endl;}
        void operator +=(Complex &c)
        {   Real=Real+c.Real;
            Image=Image+c.Image;
        }
};
void main(void)
{   Complex c1(10,20),c2(15,30);
    c1.Show(1);
    c2.Show(2);
    c2+=c1;
    c2.Show(2);
}
```

（2）友元函数

```
# include <iostream.h>
class Complex
{   private:
        float Real,Image;
    public:
        Complex(float r=0,float i=0)
        {   Real=r;Image=i;}
        void Show(int i)
        {   cout<<"c"<<i<<"=" <<Real<<"+"<<Image<<"i"<<endl;}
        friend void operator +=(Complex &c1,Complex &c2);
};
void operator +=(Complex &c1,Complex &c2)
        {   c1.Real=c1.Real+c2.Real;
            c1.Image=c1.Image+c2.Image;
        }
void main(void)
{   Complex c1(10,20),c2(15,30);
    c1.Show(1);
    c2.Show(2);
    c2+=c1;
    c2.Show(2);
}
```

11.7 定义一个描述矩阵的类 Array，其数据成员为 3×3 实数矩阵，用 Put()成员函数输入矩阵元素值，重载“+”运算符完成二个矩阵的加法。分别用成员函数与友元函数编写运算符重载函数。在主函数中定义矩阵对象 a1、a2、a3，进行矩阵加法 a3=a1+a2 运算，并输出矩阵 a1、a2、a3 的全部元素值。

解：

（1）成员函数

```
#include <iostream>
#include <iomanip.h>
#define M 3
#define N 3
class Array
{   private:
       float A[M][N];
    public:
       void Put(float a[M][N])
       {   for(int i=0;i<M;i++)
              for(int j=0;j<N;j++)
                 A[i][j]=a[i][j];
       }
       Array operator +(Array &ac)
       {    int i,j;
            Array   aa;
            for (i=0;i<M;i++)
               for (j=0;j<N;j++)
                  aa.A[i][j]=A[i][j]+ac.A[i][j];
             return aa;
       }
       void Show(int k)
       {   int i,j;
           for (i=0;i<M;i++)
           {   for (j=0;j<N;j++)
               cout<<setw(4)<<A[i][j];
               cout<<endl;
           }
       }
};
void main(void)
{    Array a1,a2,a3;
     int   i,j;
     float a[M][N];
     cout<<"Input a1[M][N]:"<<endl;
     for (i=0;i<M;i++)
        for (j=0;j<N;j++)
           cin>>a[i][j];
     a1.Put(a);
```

```
    cout<<"Input a2[M][N]:"<<endl;
    for (i=0;i<M;i++)
       for (j=0;j<N;j++)
          cin>>a[i][j];
    a2.Put(a);
    a3=a1+a2;
    a3.Show(3);
}
```

（2）友员函数

```
#include <iostream>
#include <iomanip.h>
#define M 3
#define N 3
class Array
{   private:
      float A[M][N];
    public:
      void Put(float a[M][N])
      {   for(int i=0;i<M;i++)
            for(int j=0;j<N;j++)
              A[i][j]=a[i][j];
      }
      friend Array operator +(Array &ab,Array &ac);
      void Show(int k)
      {   int i,j;
          for (i=0;i<M;i++)
          {   for (j=0;j<N;j++)
              cout<<setw(4)<<A[i][j];
              cout<<endl;
          }
      }
};
Array operator +(Array &ab,Array &ac)
{   int i,j;
    Array   aa;
    for (i=0;i<M;i++)
        for (j=0;j<N;j++)
            aa.A[i][j]=ab.A[i][j]+ac.A[i][j];
     return aa;
}

void main(void)
{   Array a1,a2,a3;
    int   i,j;
    float a[M][N];
```

```
    cout<<"Input a1[M][N]:"<<endl;
    for (i=0;i<M;i++)
       for (j=0;j<N;j++)
          cin>>a[i][j];
    a1.Put(a);
    cout<<"Input a2[M][N]:"<<endl;
    for (i=0;i<M;i++)
       for (j=0;j<N;j++)
          cin>>a[i][j];
    a2.Put(a);
    a3=a1+a2;
    a3.Show(3);
}
```

11.8 定义描述平面任意点坐标(X,Y)的类 Point，编写“－”运算符重载函数，使该函数能求出平面上任意二点的距离。在主函数中用 Point 类定义二个平面点对象 p1(1,1)、p2（4，5），再定义一个实数 d 用于存放二点间的距离，用表达式 d=p1－p2；计算出二点间的距离，并显示二点 p1、p2 的坐标值与二点距离 d。用成员函数与友元函数两种方法实现上述要求。

解：

（1）成员函数

```
# include <iostream.h>
# include <math.h>
class Point
{   private:
        float X,Y;
    public:
        Point(float x,float y)
        {   X=x;
            Y=y;
        }
        float operator -(Point & p)
        {   float d;
            d=sqrt((X-p.X)*(X-p.X)+(Y-p.Y)*(Y-p.Y));
            return d;
        }
        void Show(int i)
        {   cout<<"P"<<i<<"(X,Y)=("<<X<<","<<Y<<")"<<endl;}
};
void main(void)
{   Point p1(1,1),p2(4,5);
    float d;
    d=p1-p2;
    p1.Show(1);
    p2.Show(2);
    cout<<"d="<<d<<endl;
```

```
}
```

（2）友元函数

```
# include <iostream.h>
# include <math.h>
class Point
{   private:
        float X,Y;
    public:
        Point(float x,float y)
        {   X=x;
            Y=y;
        }
        friend float operator -(Point & p1,Point & p2);
        void Show(int i)
        {   cout<<"P"<<i<<"(X,Y)=("<<X<<","<<Y<<")"<<endl;}
};
float operator -(Point & p1,Point & p2)
{   float d;
    d=sqrt((p1.X-p2.X)*(p1.X-p2.X)+(p1.Y-p2.Y)*(p1.Y-p2.Y));
    return d;
}
void main(void)
{   Point p1(1,1),p2(4,5);
    float d;
    d=p1-p2;
    p1.Show(1);
    p2.Show(2);
    cout<<"d="<<d<<endl;
}
```

11.9 如何区分前置“++”与后置“++”的运算符重载函数？为何前置“++”重载成员函数中，必须要用 this 指针返回运算结果？

答：

（1）当形参中的无 int 时表示前置++重载函数，有 int 时表示后置++重载函数。

（2）假设要对对象 t 进行自加操作，当执行++t 时，对象 t 要调用前置“++”重载成员函数：t.opreator++()；在函数体内对 t 的数据成员进行自加操作，由于前置“++”是先加后用，所以必须将加 1 后的 t 值返回。由于调用函数中无实参，因此，重载函数内不能直接使用 t，即不能 return t 返回 t 值。但在定义 t 对象时系统要建立一个指向 t 的 this 指针。因为 t=*this，所以可用 return *this 语句可将自加后的 t 值返回给调用函数。

11.10 对类的对象进行自加时，为什么前置“++”运算符重载友元函数的形参必须为该类的引用？为什么前置“++”运算符重载友元函数返回类型必须为该类的引用？

答：

为了实现对类的对象（实参）自加操作，必须将前置“++”运算符重载友元函数的形参定义

实参对象的别名，使实参对象与形参对象占用相同的内存空间，从而使对形参对象的加 1 操作变为对实参对象的加 1 操作。因此，前置“++”运算符重载友元函数的形参必须为该类的引用。

11.11 定义一个人民币类 Money，类中数据成员为元、角、分。用成员函数重载“++”运算符，实现人民币对象的加 1 运算。在主函数中定义人民币对象 m1=10 元 8 角 5 分及对象 m2、m3。对 m1 作前置“++”并赋给 m2。对 m1 作后置“++”并赋给 m3。显示 m1、m2、m3 的结果。

解：

```
# include <iostream.h>
# include <math.h>
class Money
{   private:
       float Dollars,Jiaos,Cents;
    public:
       Money()                                  //定义默认的构造函数
       {   Dollars=Jiaos=Cents=0;}
        Money(float d,float j,float c)
       {   Dollars=d;
           Jiaos=j;
           Cents=c;
       }
       Money operator ++( );
       Money operator ++( int );
       void Show(void)
       {   cout<<Dollars<<"元"<<Jiaos<<"角"<<Cents<<"分"<<endl;}
};

Money Money::operator ++ ()
{    Cents++;
     if (Cents==10)
     {    Jiaos++;
          Cents=0;
          if (Jiaos==10)
          { Dollars++;
            Jiaos=0;
          }
     }
     return *this;                              //返回自加后的人民币对象值
}
Money Money::operator++ (int )
{    Money temp=*this;                          //将自加前人民币对象值存入临时对象 temp
     Cents++;
     if (Cents==10)
     {    Jiaos++;
          Cents=0;
          if (Jiaos==10)
```

```
            { Dollars++;
              Jiaos=0;
            }
        }
        return temp;
}
void main( void)
{   Money m1(10,8,5),m2,m3;
    m1.Show();
    m2= ++m1;
    m3= m1++;
    m2.Show();
    m3.Show();
}
```

11.12 用友元函数重载“++”运算符，实现 11.11 题中对人民币对象的加 1 运算。

```
# include <iostream.h>
# include <math.h>
class Money
{   private:
      float Dollars,Jiaos,Cents;
    public:
      Money()                                  //定义默认的构造函数
      {   Dollars=Jiaos=Cents=0;}
       Money(float d,float j,float c)
      {   Dollars=d;
          Jiaos=j;
          Cents=c;
      }
      friend Money operator ++(Money &m );
      friend Money operator ++(Money &m,int );
      void Show(void)
      {   cout<<Dollars<<"元"<<Jiaos<<"角"<<Cents<<"分"<<endl;}
};

Money operator ++ (Money &m)
{    m.Cents++;
     if (m.Cents==10)
     {    m.Jiaos++;
          m.Cents=0;
          if (m.Jiaos==10)
          { m.Dollars++;
            m.Jiaos=0;
          }
     }
```

```
    return m;
}
Money operator++ (Money &m,int )
{   Money temp=m;
    m.Cents++;
    if (m.Cents==10)
    {   m.Jiaos++;
        m.Cents=0;
        if (m.Jiaos==10)
        { m.Dollars++;
          m.Jiaos=0;
        }
    }
    return temp;
}
void main( void)
{   Money m1(10,8,5),m2,m3;
    m1.Show();
    m2= ++m1;
    m3= m1++;
    m2.Show();
    m3.Show();
}
```

11.13 编写字符串运算符“=”、“+”“<”的重载函数，使运算符“=”、“+”“<”分别用于字符串的赋值、拼接、比较运算，实现字符串直接操作运算。其中“=”与“<”运算符重载函数为友元函数，而“+”运算符重载函数为成员函数。

答：（有错）

```
# include <iostream.h>
# include <string.h>
class String                                    //定义字符串类
{   protected :
     int Length;
     char *Sp;
    public:
     String()                                   //定义缺省的构造函数
     {Sp=0;Length=0;}
     String(const char *s)                      //定义初始化构造函数
     {   Length=strlen(s);
         Sp=new char[Length +1];
         strcpy(Sp,s);
     }
     ~String()                                  //定义析构函数
     { if (Sp) delete [] Sp;}
     void Show()                                //定义显示字符串函数
```

```
        { cout<<Sp<<endl;}
        String operator + ( String &);      //定义字符串拼接友元函数
        friend int operator<(String &,String &) ;    //定义字符串比较成员函数
};

String String::operator+(String &s)
{   String t;
    t.Length=Length+s.Length;
    t.Sp=new char [t.Length+1];
    strcpy(t.Sp,Sp);
    strcat(t.Sp,s.Sp);
    return t;
}

int operator <(String &s1,String &s2)
{   if (strcmp(s1.Sp,s2.Sp)<0 ) return 1;
    else 0;
}
void main (void)
{   String s1("software"),s2("hardware"),s3;
    s3=s1;
    s1.Show();
    s2.Show();
    s3=s1+s2;
    s3.Show();
    if (s1<s2) s1.Show();
    else    s2.Show();
}
```

11.14 何为多态性技术？编译时的多态性用什么技术实现？运行时的多态性用什么函数实现？

答：

（1）多态性技术是指调用同名函数完成不同的函数功能，或使用同名运算符完成不同的运算功能。它常用重载函数与虚函数来实现。

（2）编译时的多态性用函数重载或运算符重载实现，运行时的多态性用虚函数实现。

11.15 何为虚函数？如何用虚函数实现“运行时的多态性”，何为纯虚函数？纯虚函数与虚函数有何区别？含有纯虚函数的类叫什么类？能否用该类定义对象？

答：

（1）在基类中用关键字 virtual 修饰的成员函数称为虚函数。

（2）用虚函数实现“运行时的多态性”的方法是：在派生类中定义与基类虚函数同名同参数同返回类型的虚函数，用基类定义指针变量 p，将基类或派生类对象的地址赋给 p（即 p=&对象）后，用 p->虚函数，则可实现“运行时的多态性”。

（3）将函数名赋 0 值且无函数体的虚函数称为纯虚函数。

（4）纯虚函数与虚函数的区别：

纯虚函数的函数名赋 0 值，无函数体，不能定义对象。
虚函数的函数名不能赋值，有函数体，可以定义对象。
（5）含有纯虚函数的类称为抽象类，不能用抽象类定义对象。

11.16 定义描述计算机的基类 Computer，其数据成员为处理器(CPU)、硬盘(HDisk)、内存(Mem)，定义显示数据的成员函数 Show()为虚函数。然后再由 Computer 派生出台式机类 PC 与笔记本类 NoteBook。PC 类的数据成员为显示器(Display)、键盘(Keyboard)，NoteBook 类的数据成员为液晶显示屏(LCD)，在两个派生类中定义显示机器配置的成员函数 Show()为虚函数。在主函数中，定义 PC 机与笔记本机对象，并用构造函数初始化对象。用基类 Computer 定义指针变量 p，然后用指针 p 动态调用基类与派生类中虚函数 Show()，显示 PC 机与笔记本电脑的配置。

解：

```
# include <iostream.h>
# include <string.h>
class Computer
{   protected:
        char CPU[20],HDisk[20],Mem[20];
    public:
        Computer(char c[],char h[],char m[])
        {   strcpy(CPU,c);
            strcpy(HDisk,h);
            strcpy(Mem,m);
        }
        virtual void Show(void)
        {   cout<<"CPU:"<<CPU<<'\t'<<"HDisk:"<<HDisk<<'\t'<<"Mem:"<<Mem<<endl;}
};
class PC: public Computer
{   private:
        char Display[20],Keyboard[20];
    public:
         PC(char c[],char h[],char m[],char d[],char k[]):Computer(c,h,m)
        {   strcpy(Display,d);
            strcpy(Keyboard,k);
        }
        void Show(void)
        {   cout<<"CPU:"<<CPU<<'\t'<<"HDisk:"<<HDisk<<'\t'<<"Mem:"<<Mem<<endl;
            cout<<"Display:"<<Display<<'\t'<<"Keyboard:"<<Keyboard<<endl;
        }
};
class NoteBook: public Computer
{   private:
        char LCD[20];
    public:
        NoteBook(char c[],char h[],char m[],char l[]):Computer(c,h,m)
        {   strcpy(LCD,l);
        }
```

```
    void Show(void)
    {   cout<<"CPU:"<<CPU<<'\t'<<"HDisk:"<<HDisk<<'\t'<<"Mem:"<<Mem<<endl;
        cout<<"LCD:"<<LCD<<endl;
    }
};
void main(void)
{   PC pc("赛扬 1G","Seagate40G","HY256MSDRAM","AOC15","美上美");
    NoteBook nb("P4/2G","金钻/80G(7200)","DDR/256M","飞利普 107T/107F4");
    Computer *p;
    p=&pc;
    p->Show();
    p=&nb;
    p->Show();
}
```

11.17 将 11.16 题中基类的虚函数改为纯虚函数，重新编写实现上述要求的程序。

解：

```
# include <iostream.h>
# include <string.h>
class Computer
{   protected:
        char CPU[20],HDisk[20],Mem[20];
    public:
        Computer(char c[],char h[],char m[])
        {   strcpy(CPU,c);
            strcpy(HDisk,h);
            strcpy(Mem,m);
        }
        virtual void Show(void)=0;
};
class PC: public Computer
{   private:
        char Display[20],Keyboard[20];
    public:
        PC(char c[],char h[],char m[],char d[],char k[]):Computer(c,h,m)
        {   strcpy(Display,d);
            strcpy(Keyboard,k);
        }
        void Show(void)
        {   cout<<"CPU:"<<CPU<<'\t'<<"HDisk:"<<HDisk<<'\t'<<"Mem:"<<Mem<<endl;
            cout<<"Display:"<<Display<<'\t'<<"Keyboard:"<<Keyboard<<endl;
        }
};

class NoteBook: public Computer
```

```
{   private:
        char LCD[20];
    public:
        NoteBook(char c[],char h[],char m[],char l[]):Computer(c,h,m)
        {   strcpy(LCD,l);
        }
        void Show(void)
        {   cout<<"CPU:"<<CPU<<'\t'<<"HDisk:"<<HDisk<<'\t'<<"Mem:"<<Mem<<endl;
            cout<<"LCD:"<<LCD<<endl;
        }
};
void main(void)
{   PC pc("赛扬 1G","Seagate40G","HY256MSDRAM","AOC15","美上美");
    NoteBook nb("P4/2G","金钻/80G(7200)","DDR/256M","飞利普 107T/107F4");
    Computer *p;
    p=&pc;
    p->Show();
    p=&nb;
    p->Show();
}
```

11.18 读下列程序，写出程序执行后输出结果。

```
# include <iostream.h>
class Base
{       int x,y;
    public:
       Base(int a,int b){x=a;y=b;}
       void virtual f() { cout<<x+y<<'\n';}
       void virtual g() { cout<<x*y<<'\n';}
};
class Derive:public Base
{      int z;
    public:
       Derive(int a,int b,int c):Base(a,b)
       {z=c;}
       void f() { cout<<z+z<<'\n';}
void g(int a=0) { cout<<2*z<<'\n';}
};
void main(void)
{   Derive d(10,10,5);
    Base *p=&d;
    p->f();
    p->g();
}
```

习题 12

12.1 解释 C++中有关流、流类、流类库的概念及流类体系的组成，并指出流类体系中的派生关系。

答：

（1）用流类定义的对象称为流；

（2）流类是 C++中为执行输入输出操作而专门定义的类；

（3）流类库是 C++中所有流类的集合。

流类库由基类 ios、输入类 istream、输出类 ostream、输入/输出类 iostream 等组成。由基类 ios 派生出输入类 istream 与输出类 ostream，而由输入类 istream 与输出类 ostream 派生出输入输出类 iostream。

12.2 在数据的输入与输出过程中，为何要使用缓冲区？在 C++中有哪些标准输入输出流，其中哪些是缓冲流？哪些是非缓冲流？

答：

（1）由于内存与磁盘之间的 I/O 操作速度较慢，若每次读/写操作都直接对磁盘进行会降低系统的工作效率。为了提高数据的输入与输出效率，在内存区中设置缓冲区，每次写操作时，先将数据写入内存缓冲区，当缓冲区满后再写入磁盘。通过减少 I/O 次数来提高数据的传送速度。这就是使用缓冲区原因。

（2）在 C++中有标准输入输出流有：cin 、cout、clog、cerr；

cin 、cout、clog 为缓冲流，cerr 为非缓冲流。

12.3 C++中数据的输入输出有哪二种方式？控制数据输出格式的函数有哪二类？使用时有何区别？

答：

数据的输入/输出两种方式：

（1）标准的输入输出流 cin、cout 与提取运算符“>>”、插入运算符“<<”；

（2）输入/输出成员函数 get()与 put()。

控制数据输出格式的函数有：

（1）格式控制成员函数，使用时必须用输入输出流 cin、cout 来调用。

（2）预定义格式控制函数，使用时可直接调用。

12.4 使用 width 成员函数控制二维数组各元素的输出宽度为 10，前导字符为“#”。

解：

```
# include <iostream.h>
void main(void)
{   float a[3][3]={11,22.2,33.33,44.444,55.5555,6,7,8,9};
    for (int i=0;i<3;i++)
    {   for (int j=0;j<3;j++)
          {cout.width(10);                    //设置输出域宽为 10
           cout.fill('#');                   //设置填充字符"#"
           cout<<a[i][j]<<'\t';
          }
        cout<<endl;
    }
```

```
}
```

12.5 定义一个学生成绩类 Score，描述学生成绩的私有数据成员为学号(No)、姓名（Name[8]）、数学(Math)、物理(Phi)、数据结构(Date)。定义能输入学生成绩的公有成员函数 Input()，在 Input()函数用 getline 函数输入学生姓名。定义输出学生成绩的成员函数 Show()。在显示函数 Show()中，学号与姓名的输出域宽为 10、左对齐，其余数据的输出域宽为 8、右对齐、保留小数点后一位，输出格式均用预定义格式控制函数设置。在主函数中用 Score 类定义班级学生成绩对象数组 s[5]。用 Input()输入学生成绩，用 Show()显示每个学生的成绩。(提示：用 getline 输入姓名后，必须用回车结束姓名输入)

解：

```
# include <iostream.h>
# include <iomanip.h>
class Score
{    private:
          int No;
          char Name[8];
          float Math,Phi,Data;
     public:
        void Input(void)
        {   cin>>No;
            cin.getline(Name,8);
            cin>>Math>>Phi>>Data;
        }
         void Show()
         {     cout<<setiosflags(ios::left)
               <<setw(10)<<No
               <<setw(10)<<Name
               <<setiosflags(ios::right|ios::fixed)
               <<setprecision(1)
               <<setw(8)<<Math
               <<setw(8)<<Phi
               <<setw(8)<<Data<<'\n';
         }
};

void main(void)
{    int i,no;
     Score   s[5];
     cout<<"请输入学号、姓名、数学、物理、数据结构 \n";
     for (i=0;i<5;i++)
     {
         s[i].Input();

     }
     cout<<"学号     姓名     数学     物理     数据结构 \n";
```

```
    for (i=0;i<5;i++)
        s[i].Show();
}
```

12.6 何为文件？如何使用文件？

答：

（1）文件是由文件名标识的一组有序数据的集合，文件通常存放在磁盘上。

（2）文件的使用过程为：打开、读/写、关闭文件。文件的打开、读/写、关闭必须使用文件流类定义的成员函数来完成。

12.7 叙述文件流类体系的组成及各派生类的作用。

答：

（1）*文件流类体系由 filebuf 管理缓冲区类、ofstream 写文件类、ifstream 读文件类、fstream 读/写文件类等组成。*

（2）派生类 ofstream 负责定义写文件（即输出文件）流对象；

派生类 ifstream 负责定义读文件（即读出文件）流对象；

派生类 fstream 负责定义读/写文件（即输入/输出文件）流对象；

12.8 叙述文本文件的使用过程。

答：

文本文件的具体使用步骤如下：

(1) 用文件流类定义文件流对象，

(2) 打开文件

①用成员函数 open()打开文件

②用构造函数打开文件

(3) 读/写文件

①用提取运算符“>>”与插入运算符“<<”对文件进行读与写操作；

②用成员函数 get()、getline()与 put() 对文件读写。

(4)关闭文件

用成员函数 close 关闭文件。

12.9 将一个源文件复制为一个目的文件，源文件用成员函数打开，目的文件用构造函数打开，使用提取与插入运算符读写文件。 （提示：将当前目录下的.cpp 文件作为源文件）

解：

```
# include <iostream.h>
# include <fstream.h>
# include <stdlib.h>
void main(void)
{   char fname1[256],fname2[256];
    cout<<"输入源文件名:";
    cin>>fname1;
    cout<<"输入目的文件名:";
    cin>>fname2;
    ifstream infile;
    infile.open(fname1);
```

```
    ofstream outfile(fname2);
    if (!infile)
    {   cout<<"不能打开输入文件："<<fname1<<endl;
        exit(1);
    }
    if (!outfile)
    {   cout<<"不能打开目的文件："<<fname2<<endl;
        exit(1);
    }
    infile.unsetf(ios::skipws);       //设置为不要跳过文件中的空格。
    char ch;
    while (infile.get(ch))            //从源文件中提取一个字符到变量 ch 中；
        outfile.put(ch);              //将 ch 中的字符写入目的文件中。
    infile.close();                   //关闭源文件
    outfile.close();                  //关闭目的文件
}
```

12.10 使用成员函数打开文本文件，并将源文件中内容添加到目的文件的尾部。

解：

```
# include <iostream.h>
# include <fstream.h>
# include <stdlib.h>
void main(void)
{   char fname1[256],fname2[256];
    char buff[300];
    cout<<"输入源文件名:";
    cin>>fname1;
    cout<<"输入目的文件名:";
    cin>>fname2;
    fstream infile,outfile;
    infile.open(fname1,ios::in | ios::nocreate);
    outfile.open(fname2,ios::app);
    if (!infile)
    {   cout<<"源文件不存在，不能打开源文件！"<<endl;
        exit(1);
    }
     if (!outfile)
    {   cout<<"目标文件不存在，不能打开目标文件！"<<endl;
        exit(2);
    }
    while (infile.getline(buff,300))          //从源文件中读一行字符到缓冲区；
          outfile<<buff<<'\n';                //  将缓冲区中一行字符写入目的文件中。
    infile.close();                           //关闭源文件
    outfile.close();                          //关闭目的文件
 }
```

12.11 产生一个九九乘法表文件，文件名为 mul.txt。

解：

```
# include <fstream.h>
# include <iomanip.h>
# include <stdlib.h>
void main(void)
{   float a[10][10];
    int i,j;
    char fname[256];
    cout<<"输入文件名：";
    cin>>fname;
    ofstream outfile;
    outfile.open(fname);
    if (!outfile)
    {   cout<<"不能打开目的文件:"<<fname;
        exit(1);
    }
    for (j=0;j<10;j++)
          a[0][j]=j;
    for (i=1;i<10;i++)
          a[i][0]=i;
    for ( i=1 ;i<10;i++)
      for (j=1;j<10;j++)
        a[i][j]=i*j;
    for(i=0;i<10;i++)
    {   for(j=0;j<10;j++)
        outfile<<setw(4)<<a[i][j];
        outfile<<'\n';
    }
    outfile.close();
}
```

12.12 定义一维数组，用键盘输入数据。然后将一维数组元素值写入文本文件（文件名为 data.txt）。

解：

```
# include <stdlib.h>
# define N 5
void main(void)
{   float a[N];
    int i,j;
    char fname[256];
    cout<<"输入文件名：";
    cin>>fname;
```

```
    ofstream outfile;
    outfile.open(fname);
    if (!outfile)
    {   cout<<"不能打开目的文件:"<<fname;
        exit(1);
    }
    cout<<"请输入"<<N<<"个数据！"<<endl;
    for (i=0;i<N;i++)
          cin>>a[i];
    for(i=0;i<N;i++)
      outfile<<setw(4)<<a[i];
      outfile<<'\n';
    outfile.close();
}
```

12.13 从文本文件中读取数据输入一维数组，求出一维数组的平均值。再从屏幕上输出一维数组的元素值及平均值。（提示：程序执行前，先将 12.13 题执行后的文件 data.txt 复制到 12.14 题的目录）

解：

```
# include <fstream.h>
# include <iomanip.h>
# include <stdlib.h>
# define N 5
void main(void)
{   float a[N],s,ave;
    int i,j;
    char fname[256];
    cout<<"输入源文件名：";
    cin>>fname;
    ifstream infile;
    infile.open(fname);
    if (!infile)
    {   cout<<"不能打开目的文件:"<<fname;
        exit(1);
    }
    s=0;
    for(i=0;i<N;i++)
    {   infile>>a[i];
        s=s+a[i];
    }
    ave=s/N;
    cout<<"输出"<<N<<"个数据！"<<endl;
    for (i=0;i<N;i++)
          cout<<setw(4)<<a[i];
    cout<<endl;
```

```
    cout<<"ave="<<avedat<<endl;
    infile.close();
}
```

12.14 从文本文件中读取数据输入一维数组，对一维数组元素进行升序排列。再存入另一个文本文件中。（提示：程序执行前，先将 12.13 题执行后的文件 data.txt 复制到 12.14 题的目录）

解：

```
# include <fstream.h>
# include <iomanip.h>
# include <stdlib.h>
# define N 5
void main(void)
{   float a[N],temp;
    int i,j,k;
    char fname1[256],fname2[256];
    cout<<"输入源文件名：";
    cin>>fname1;
    ifstream infile;
    infile.open(fname1);
    if (!infile)
    {   cout<<"不能打开目的文件:"<<fname1;
        exit(1);
    }
    cout<<"输入目标文件名：";
    cin>>fname2;
    ofstream outfile;
    outfile.open(fname2);
    if (!outfile)
    {   cout<<"不能打开目的文件:"<<fname2;
        exit(1);
    }
    for(i=0;i<N;i++)
      infile>>a[i];
    for(i=0;i<N-1;i++)
    {  k=i;
       for(j=i+1;j<N;j++)
          if (a[k]>a[j]) k=j;
       if (k>i)
       {   temp=a[i];a[i]=a[k];a[k]=temp;}
    }
    for (i=0;i<N;i++)
         cout<<setw(4)<<a[i];
    cout<<endl;
    for(i=0;i<N;i++)
      outfile<<setw(4)<<a[i];
```

```
    infile.close();
    outfile.close();
}
```

12.15 使用二进制成员函数将源文件中内容添加到目的文件的尾部。

解：

```
# include <fstream.h>
# include <stdlib.h>
void main(void)
{   char fname1[256],fname2[256];
    char buff[4096];                                    //建立 4K 缓冲区
    cout<<"输入源文件名:";
    cin>>fname1;
    cout<<"输入目标件名:";
    cin>>fname2;
    fstream infile,outfile;
    infile.open(fname1,ios::in | ios::binary);
    outfile.open(fname2,ios::app| ios::binary);
    if (!infile)
    {   cout<<"不能打开输入文件："<<fname1<<endl;
        exit(1);
    }
     if (!outfile)
    {   cout<<"不能打开目的文件："<<fname2<<endl;
        exit(2);
    }
    int n;
    while (!infile.eof())                   //文件不结束就继续循环；
    {   infile.read(buff,4096);             //一次读 4096 个字节
        n=infile.gcount();                  //取实际读的字节数
        outfile.write(buff,n);              //按实际读的字节数写入文件
    }
    infile.close();                         //关闭源文件
    outfile.close();                        //关闭目的文件
}
```

12.16 产生一个二进制数据文件，将 1~50 之间的所有偶数写入文件 data.dat 中。

（提示：由于数组下标从 1 开始，所以应定义 26 个数组单元，保存时，也应保存 26 个数据。）

解：

```
# include <fstream.h>
# include <math.h>
# include <stdlib.h>
# include <iomanip.h>
void main (void)
{
```

```
    fstream outfile("even.bin",ios::out | ios::binary); //以只写方式打开二进制文件 even.bin
    int i;
    if (!outfile)
    {   cout<<"不能打开输出文件 even.bin \n";
        exit(1);
    }
    int s[26],j=1;
    for (i=1;i<=50;i++)
        if (i%2==0)
        {   s[j]=i; j++;}
    for (i=1 ;i<=25;i++)
        cout <<s[i]<<'\t';
    outfile.write((char * )s,sizeof(int) *26);              //一次写入 25 个整数
    outfile.close();
}
```

12.17 从 12.16 题中产生的数据文件中读取二进制数据，并在显示器上每行 5 个数的形式显示。（提示：由于数组下标从 1 开始，所以应定义 26 个数组单元，读取数据时，应读取 26 个数据。）

解：

```
# include <fstream.h>
# include <math.h>
# include <stdlib.h>
# include <iomanip.h>
void main (void)
{
    int s[26],i;
    fstream infile("even.bin",ios::in | ios::binary); //以只读方式打开二进制文件 even.bin
    if (!infile)
    {   cout<<"不能打开输入文件 even.bin \n";
        exit(1);
    }
    infile.read((char* )s,sizeof(int)*26);              //一次读出 25 个整数
    for (i=1;i<=25;i++)
    {   cout<<setw(10)<<s[i]<<'\t';
        if ((i)%5==0) cout<<endl;
    }
    infile.close();
}
```

12.18 从 12.16 题的数据文件 data.dat 文件中，读出文件中第 n 个偶数并显示在屏幕上。再将文件指针移动 m 个偶数单元，在该单元写入新的数据 a，最后将数据文件中的全部数据以每行 5 个的形式在屏幕上显示。n、m 与 a 的值用键盘输入。

解：

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

```
# include <string.h>
# include <iomanip.h>
void main (void)
{
    int c,n,m,a,s[26],i;
    cout<<"Input n,m,a:";
    cin>>n,m,a;
    ifstream infile("even.bin",ios::binary |ios::in );
    if (!infile)
    {   cout<<"不能打开输入文件 even.bin \n";
        exit(1);
    }
    infile.seekg(n*sizeof(int));
    infile.read((char * )&c,sizeof(int));
    cout<<c<<'\n';
    ofstream outfile("even.bin",ios::binary |ios::out);
    if (!outfile)
    {   cout<<"不能打开输入文件 even.bin \n";
        exit(1);
    }
    outfile.seekp(m*sizeof(int));
    outfile.write((char *)&a,sizeof(int));
    outfile.flush();
    infile.seekg(m*sizeof(int));
    infile.read((char * )&c,sizeof(int));
    cout<<c<<'\n';
    infile.read((char* )s,sizeof(int)*26);          //一次读出 25 个整数
    for (i=1;i<=25;i++)
    {     cout<<setw(10)<<s[i]<<'\t';
          if ((i)%5==0) cout<<endl;
    }
    infile.close();
    outfile.close;
}
```