# 面向对象程序设计综合练习题

## 一.单项选择题

1．C++源程序文件的缺省扩展名为(　)。
A. cpp　　B. exe　　C. obj　　D. lik
2．由 C++源程序文件编译而成的目标文件的缺省扩展名为(　)。
A. cpp　　B. exe　　C. obj　　D. lik
3．由 C++目标文件连接而成的可执行文件的缺省扩展名为(　)。
A. cpp　　B. exe　　C. obj　　D. lik
4．编写 C++程序一般需经过的几个步骤依次是(　　)。
A. 编译、编辑、连接、调试　　B. 编辑、编译、连接、调试
C. 编译、调试、编辑、连接　　D. 编辑、调试、编辑、连接
5．以下标识符中不全是保留字的是（　）。
A. case　for　int　　B. default　then　while
C. bool　class　long　　D. goto　return　char
6．能作为 C++程序的基本单位是(　)。
A. 字符　　B. 语句　　C. 函数　　D. 源程序文件
7．程序中主函数的名字为（　）。
A. main　　B. MAIN　　C. Main　　D. 任意标识符
8．C++程序的基本模块为（　）。
A. 表达式　　B. 标识符　　C. 语句　　D. 函数
9．可用作 C++语言用户标识符的一组标识符是(　)。
A. void　define　+WORD
B. a3_b3　_123　YN
C. for　-abc　Case
D. 2a　DO　sizeof
10．存储以下数据，占用存储字节最多的是（　）。
A. 0　　B. ‘0’　　C. “0”　　D. 0.0
11．程序运行中需要从键盘上输入多于一个数据时，各数据之间应使用(　　)符号作为分隔符。
A. 空格或逗号　　B. 逗号或回车　　C. 逗号或分号　　D. 空格或回车
12．设”int a=12;”，则执行完语句”a+=a*a;”后，a 的值是(　)。
A. 12　　B. 144　　C. 156　　D. 288
13．假设在程序中 a、b、c 均被定义成整型，所赋的值都大于 1，则下列能正确表示代数式的表达式是(　)。
A. 1.0/a*b*c　　B. 1/(a*b*c)
C. 1/a/b/(float)c　　D. 1.0/a/b/c
14．设”int a=15,b=26;”，则”cout<<(a,b);”的输出结果是(　)。
A. 15　　B. 26,15
C. 15,26　　D. 26
15．设 x 是一个 bool 型的逻辑量，y 的值为 10，则表达式 x && y 的值为（　）。
A. 1　　B. 0
C. 与 x 值相同　　D. 与 x 值相反
16．x>0 && x<=10 的相反表达式为（　）。

A. x<=0 || x>10　　B. x<=0 && x>10
C. x<=0 || x<=10　　D. x>0 && x>10
17．x>0 || y==5 的相反表达式为（　　）。
A. x<=0 || y!=5　　B. x<=0 && y!=5
C. x>0 || y!=5　　D. x>0 && y==5
18．设 x 和 y 均为 bool 量，则 x && y 为真的条件是(　　)。
A. 它们均为真　　B. 其中一个为真
C. 它们均为假　　D. 其中一个为假
19．设 x 和 y 均为 bool 量，则 x || y 为假的条件是(　)。
A. 它们均为真　　B. 其中一个为真
C. 它们均为假　　D. 其中一个为假
20．字符串”a+b=12\n”的长度为（　）。
A. 6　　B. 7
C. 8　　D. 9
21．假定下列 x 和 y 均为 int 型变量，则不正确的赋值为（　　）。
A. x+=y++　　B. x++=y++
C. x=++y　　D. ++x=++y
22．下列的符号常量定义中，错误的定义是（　　）。
A. const M=10;　　B. const int M=20;
C. const char ch;　　D. const bool mark=true;
23．循环语句“for(int i=0; i<n; i++) cout<<i*i<<’ ’;”中循环体执行的次数为(　　)。
A. 1　　B. n-1
C. n　　D. n+1
24．在下面循环语句中循环体执行的次数为（　　）。
for(int i=0; i<n; i++)
if(i>n/2) break;
A. n/2　　B. n/2+1
C. n/2-1　　D. n-1
25．在下面循环语句中内层循环体 S 语句的执行总次数为（　）。
for(int i=0; i<n; i++)
for(int j=i; j<n; j++) S;
A. n2　　B. (n+1)/2
C. n(n-1)/2　　D. n(n+1)/2
26．在下面循环语句中循环体执行的次数为（　　）。
int i=0,s=0; while(s<20) {i++; s+=i;}
A. 4　　B. 5
C. 6　　D. 7
27．在下面循环语句中循环体执行的次数为（　）。
int i=0; do i++; while(i*i<10);
A. 4　　B. 3
C. 5　　D. 2
28．当处理特定问题时的循环次数已知时，通常采用（　）来解决。
A. for 循环　　B. while 循环
C. do 循环　　D. switch 语句
29．循环体至少被执行一次的语句为（　　）。

A. for 循环　　B. while 循环
C. do 循环　　D. 任一种循环
30．switch 语句能够改写为（　）语句。
A. for　　B. if
C. do　　D. while
31．do 语句能够改写为（　）语句。
A. 复合　　B. if
C. switch　　D. while
32．．在下面的一维数组定义中，哪一个有语法错误。（　）
A. int a[]={1,2,3};　　B. int a[10]={0};
C. int a[];　　D. int a[5];
33．在下面的字符数组定义中，哪一个有语法错误。（　）。
A. char a[20]=”abcdefg”;　　B. char a[]=”x+y=55.”;
C. char a[15];　　D. char a[10]=’5’;
34．在下面的二维数组定义中，正确的是(　)。
A. int a[5][];　　B. int a[][5];
C. int a[][3]={{1,3,5},{2}};　　D. int a[](10);
35.假定一个二维数组的定义语句为“int a[3][4]={{3,4},{2,8,6}};”,则元素 a[1][2]的值为（　）。
A. 2　　B. 4
C. 6　　D. 8
36.假定一个二维数组的定义语句为“int a[3][4]={{3,4},{2,8,6}};”,则元素 a[2][1]的值为（　）。
A. 0　　B. 4
C. 8　　D. 6
37．若定义了函数 double *function(), 则函数 function 的返回值为（　）。
A. 实数型　　B.实数的地址
C.指向函数的指针　　D.函数的地址
38．以下说法中正确的是（　）。
A. C++程序总是从第一个定义的函数开始执行
B. C++程序总是从 main 函数开始执行
C. C++函数必须有返回值，否则不能使用函数
D. C++程序中有调用关系的所有函数必须放在同一个程序文件中
39．以下叙述中不正确的是（　）。
A. 在一个函数中，可以有多条 return 语句
B. 函数的定义不能嵌套，但函数的调用可以嵌套
C. 函数必须有返回值
D. 不同的函数中可以使用相同名字的变量
40．函数重载是指（　）。
A. 两个或两个以上的函数取相同的函数名，但形参的个数或类型不同
B. 两个以上的函数取相同的名字和具有相同的参数个数，但形参的类型可以不同
C. 两个以上的函数名字不同，但形参的个数或类型相同
D. 两个以上的函数取相同的函数名，并且函数的返回类型相同
41．以下关于函数模板叙述正确的是（　）。
A. 函数模板也是一个具体类型的函数
B. 函数模板的类型参数与函数的参数是同一个概念
C. 通过使用不同的类型参数，函数模板可以生成不同类型的函数

D. 用函数模板定义的函数没有类型
42．下列（　　）的调用方式是引用调用。
A. 形参和实参都是变量　　B.形参是指针，实参是地址值
C. 形参是引用，实参是变量　　D.形参是变量，实参是地址值
43．为了提高程序的运行速度，可将不太复杂的功能用函数实现，此函数应选择（　　）。
A. 内联函数　　B.重载函数
C.递归函数　　D.函数模板
44．函数原型语句正确的是（　　）。
A. int Function(void a);　　B.void Function (int);
C. int Function(a);　　D.void int(double a);
45．C++中函数返回值的类型是由（　　）决定的。
A. return 语句中表达式的类型　　B.该函数定义时的类型
C.调用函数时的调用语句　　D.系统根据结果
46．以下函数的返回结果是（　　）。

```
int function(char *x) {
char *p=x; while(*p++); return(p-x-1);
}
```

A. 求字符串的长度　　B.将字符串 x 连接到字符串 p 后面
C. 将字符串 x 复制到字符串 p 中　　D.将字符串 x 反向存放
47．函数调用 func((exp1,exp2),(exp3,exp4,exp5))中所含实参的个数为(　　)个。
A.　1　　B.　2
C.　4　　D.　5
48．设有如下函数定义

```
int f(char *s) {
char *p=s;
while(*p!=’\0’) p++;
return(p-s);
}
```

在主函数中用 cout<<f(“good”)调用上述函数，则输出结果为（　　）。
A. 3　　B. 4
C. 5　　D. 6
49．以下正确的函数原型语句是(　　)。
A. double fun(int x,int y)　　B. double fun(int x;int y)
C. double fun(int,int);　　D. double fun(int x,y);
50．以下正确的说法是（　　）。
A. 用户调用标准库函数前，必须重新定义
B. 用户可以重新定义标准库函数，若如此，该函数将失去原有含义
C. 系统不允许用户重新定义标准库函数
D. 用户调用标准库函数前，不必使用预编译命令将该函数所在文件包括到用户源文件中
51．函数调用不可以（　　）。
A. 出现在一个表达式中　　B. 出现在执行语句中
C. 作为一个函数的实参　　D. 作为一个函数的形参
52．以下正确的描述是（　　）。
A. 函数的定义可以嵌套，函数的调用不可以嵌套
B. 函数的定义不可以嵌套，函数的调用可以嵌套

C. 函数的定义和函数的调用均可以嵌套
D. 函数的定义和函数的调用均不可以嵌套
53．若用数组名作为函数调用的实参，传递给形参的是（　　）。
A. 数组的首地址　　B. 数组中第一个元素的值
C. 数组全部元素的值　　D. 数组元素的个数
54．以下错误的描述是（　　）。
A. 被调用函数中可以不用 return 语句
B. 被调用函数中可以用多个 return 语句
C. 被调用函数中，如果有返回值，就一定要有 return 语句
D. 被调用函数中，一个 return 语句可返回多个值给调用函数
55．以下正确的描述是（　　）。
A. 不允许设置参数的默认值
B. 设置参数的默认值只能在定义函数时设置
C. 设置参数的默认值时，应该设置右边的参数
D. 设置参数的默认值时，应该全部参数都设置
56．采用重载函数的目的是（　　）。
A. 实现共享　　B. 减少空间
C. 提高速度　　D. 使用方便，提高可读性
57．将两个字符串连接起来组成一个字符串时，选用（　　）函数。
A. strlen()　　B. strcap()
C. strcat()　　D. strcmp()
58．以下叙述中正确的是(　　)。
A. 使用#define 可以为常量定义一个名字，该名字在程序中可以再赋另外的值
B. 使用 const 定义的常量名有类型之分，其值在程序运行时是不可改变的
C. 在程序中使用内联函数使程序的可读性变差
D. 在定义函数时可以在形参表的任何位置给出缺省形参值
59．下面的标识符中，(　　)是文件级作用域。
A. 函数形参　　B.语句标号
C.外部静态类标识符　　D.自动类标识符
60．以下叙述不正确的是(　　)。
A. 宏替换不占用运行时间　　B. 宏名无类型
C. 宏替换只是字符替换　　D. 宏名必须用大写字母表示
61．C++语言的编译系统对宏命令的处理是(　　)。
A. 在程序运行时进行
B. 在程序连接时进行
C. 和 C++程序的其它语句同时进行编译
D. 在对源程序中其它成分正式编译之前进行
62．当#include 后面的文件名用双引号括起来时，寻找被包含文件的方式是(　　)。
A. 直接按系统设定的标准方式搜索目录
B. 先在源程序所在目录搜索，再按系统设定的标准方式搜索目录
C. 仅仅搜索源程序所在目录
D. 搜索当前逻辑盘上的所有目录
63．当#include 后面的文件名用尖括号括起来时，寻找被包含文件的方式是(　　)。
A. 直接按系统设定的标准方式搜索目录
B. 先在源程序所在目录搜索，再按系统设定的标准方式搜索目录

C. 仅仅搜索源程序所在目录
D. 搜索当前逻辑盘上的所有目录
64．在下面存储类中，（　　）对象的可见性与生存期不一致。
A. 外部类　　B. 自动类
C. 内部静态类　　D. 寄存器类
65．在下面存储类中，（　　）的对象不是局部变量。
A. 外部静态类　　B. 自动类
C. 函数形参　　D. 寄存器类
66．关于局部变量，下面说法正确的是(　　)。
A. 定义该变量的程序文件中的函数都可以访问
B. 定义该变量的函数中的定义处以下的任何语句都可以访问
C. 定义该变量的复合语句中的定义处以下的任何语句都可以访问
D. 定义该变量的函数中的定义处以上的任何语句都可以访问
67．文件包含命令中被包含的文件的扩展名（　　）。
A. 必须是.h　　B. 不能是.h
C. 可以是.h 或.cpp　　D. 必须是.cpp
68．预处理命令在程序中都是以(　　)符号开头的。
A. *　　B. #
C. &　　D. @
69．设 array 为一个数组，则表达式 sizeof(array)/sizeof(array[0])的结果为(　　)。
A. array 数组首地址
B. array 数组中元素个数
C. array 数组中每个元素所占的字节数
D. array 数组占的总字节数
70．用 new 运算符创建一个含 10 个元素的一维整型数组的正确语句是(　　)。
A. int *p=new a[10];　　B. int *p=new float[10];
C. int *p=new int[10];　　D. int *p=new int[10]={1,2,3,4,5}
71．下列给字符数组赋初值时，正确的是(　　)。
A. char s1[]=”abcdef”;　　B. char s2[4]=”abcd”;
C. char s3[2][3]={“abc”,”xyz”};　　D. char s4[4][]={‘a’,’x’,’s’,’t’};
72．设有定义“static int data[5][6];”，设该数组在静态存储区中的起始地址为十进制的 100，若每个 int 型数占 4 个字节，则数组元素 data[2][3]的地址为(　　)。
A. 115　　B.124
C.130　　D.160
73．假定变量 m 定义为“int m=7;”，则定义变量 p 的正确语句为（　　）。
A.　int p=&m;　　B.　int *p=&m;
C.　int &p=*m;　　D.　int *p=m;
74．．假定 k 是一个 double 类型的变量，则关于变量 p 的正确定义语句为（　　）。
A. double p=&k;　　B. int *p=&k;
C. double &p=*k;　　D. char *p=” Thank you!”;
75．变量 s 的定义为“char *s=” Hello world!”;”，要使变量 p 指向 s 所指向的同一个字符串，则应选取（　　）。
A. char *p=s;　　B. char *p=&s;
C. char *p;p=*s;　　D. char *p; p=&s;
76．关于 void 指针，下列说法正确的是（　　）。

A. void 指针就是未指向任何数据的指针
B. void 指针就是已定义而未初始化的指针
C. 指向任何类型数据的指针可直接赋值给一个 void 指针
D. void 指针值可直接赋给一个非 void 指针
77．假定一条定义语句为“int a[10], x, *pa=a;”，若要把数组 a 中下标为 3 的元素值赋给 x，则不正确的语句为（　　）。
A. x=pa[3];　　B. x=*(a+3);
C. x=a[3];　　D. x=*pa+3;
78．假定有定义“int b[10]; int *pb;”，则不正确的赋值语句为（　　）。
A. pb=b;　　B. pb=&b[0];
C. *pb=new int;　　D. pb=b[5];
79．假定 p 是具有 double 类型的指针变量，则表达式++p 使 p 的值（以字节为单位）增加（　　）。
A. 1　　B. 4
C. sizeof(double)　　D. sizeof(p)
80．假定 p 指向的字符串为”string”，则 cout<<p+3 的输出结果为（　　）。
A. string　　B. ring
C. ing　　D. i
81．假定 p 指向的字符串为”string”，若要输出这个字符串的地址值，则使用（　　）。
A. cout<<*s;　　B. cout<<s;
C. cout<<&s;　　D. cout<<(void *)s;
82．定义 p 并使 p 指向动态空间中的包含 30 个整数的数组所使用的定义语句为（　　）。
A. int *p=new int[30];　　B. int *p=new int(30);
C. int *p=new [30];　　D. *p=new int[30];
83．假定 p 是具有 int**类型的指针变量，则给 p 赋值的正确语句为（　　）。
A. p=new int;　　B. p=new int*;
C. p=new int**;　　D. p=new int[10];
84．要使语句“p=new int[10][20];”能够正常执行，p 应被事先定义为（　　）。
A. int *p;　　B. int **p;
C. int *p[20];　　D. int(*p)[20];
85．假定有“struct BOOK{char title[40]; float price;}; BOOK *book=new BOOK;”，则正确的语句为（　　）。
A. strcpy(book->title,” Wang Tao”); B. strcpy(book.title,” Wang Tao”);
C. strcpy(*book.title,” Wang Tao”); D. strcpy((*book)->title,” Wang Tao”);
86．假定有“struct BOOK{char title[40]; float price;}; BOOK *book;”，则不正确的语句为（　　）。
A. BOOK *x=new book;　　B. BOOK x={"C++ Programming",27.0};
C. BOOK *x=new BOOK;　　D.　BOOK *x=book;
87．假定有“struct BOOK{char title[40]; float price;}book;”，则正确的语句为（　　）。
A. BOOK &x= &book;　　B. BOOK &x=book;
C. BOOK &x=new BOOK;　　D. BOOK &x=BOOK;
88．下列对引用的陈述中不正确的是（　　）。
A. 每一个引用都是其所引用对象的别名,因此必须初始化
B. 形式上针对引用的操作实际上作用于它所引用的对象
C. 一旦定义了引用，一切针对其所引用对象的操作只能通过该引用间接进行
D. 不需要单独为引用分配存储空间
89．假定变量 a 和 pa 定义为“double a[10], *pa=a;”，要将 12.35 赋值给 a 中的下标为 5 的元素，

不正确的语句是（　　）。
A. pa[5]=12.35;　　B. a[5]=12.35;
C. *(pa+5)=12.35;　　D. *(a[0]+5)=12.35;
90. 假定变量 b 和 pb 定义为“int b[10], *pb=b;”，要将 24 赋值给 b[1]元素中，不正确的语句是（　　）。
A. *(pb+1)=24;　　B. *(b+1)=24;
C. *++b=24;　　D. *++pb=24;
91. 假定指针变量 p 定义为“int *p=new int(100);”，要释放 p 所指向的动态内存，应使用语句（　　）。
A. delete p;　　B. delete *p;
C. delete &p;　　D. delete []p;
92. 假定指针变量 p 定义为“int *p=new int[30];”，要释放 p 所指向的动态内存，应使用语句（　　）。
A. delete p;　　B. delete *p;
C. delete &p;　　D. delete []p;
93. 假定变量 x 定义为“int x=5;”，要使 rx 成为 x 的引用（别名），rx 应定义为（　　）。
A. int rx=x;　　B. int rx=&x;
C. int *rx=&x;　　D. int &rx=x;
94. 关于消息，下列说法中不正确的是（　　）。
A. 发送消息的对象请求服务，接受消息的对象提供服务
B. 消息的发送者必须了解消息的接收者如何相应消息
C. 在 C++中，消息的发送具体体现为对接收消息的对象的某个函数的调用
D. 每个对象只能接收某些特定格式的消息
95. 关于封装，下列说法中不正确的是（　　）。
A. 通过封装，对象的全部属性和操作结合在一起，形成一个整体
B. 通过封装，一个对象的实现细节被尽可能地隐藏起来（不可见）
C. 通过封装，每个对象都成为相对独立的实体
D. 通过封装，对象的属性都是不可见的
96. 面向对象方法的多态性是指（　　）。
A. 一个类可以派生出多个特殊类
B. 一个对象在不同的运行环境中可以有不同的变体
C. 针对一消息，不同的对象可以以适合自身的方式加以响应
D. 一个对象可以是由多个其他对象组合而成的
97. 软件产品在需求发生变化、运行环境发生变化或发现软件产品本身的错误或不足时进行相应的软件更新的难易程度叫做软件的（　　）。
A. 可维护性　　B. 可复用性
C. 兼容性　　D. 正确性
98. 软件产品可被全部或部分地再用于新的应用的能力叫做软件的（　　）。
A. 可维护性　　B. 可复用性
C. 兼容性　　D. 正确性
99. 软件产品与其他软件产品组合成一个整体的难易程度叫做软件的（　　）。
A. 可维护性　　B. 可复用性
C. 兼容性　　D. 正确性
100. 软件产品准确执行软件需求规格说明书中所规定的任务的能力叫做软件的（　　）。
A. 可维护性　　B. 可复用性
C. 兼容性　　D. 正确性
101. 面向对象软件开发中使用的 OOA 表示（　　）。
A. 面向对象分析　　B. 面向对象设计

C. 面向对象语言　　D. 面向对象方法
102．面向对象软件开发中使用的 OOD 表示（　）。
A. 面向对象分析　　B. 面向对象设计
C. 面向对象语言　　D. 面向对象方法
103．关于面向对象系统分析，下列说法中不正确的是（　）。
A. 术语“面向对象分析”可以用缩写 OOA 表示
B. 面向对象分析阶段对问题域的描述比实现阶段更详细
C. 面向对象分析包括问题域分析和应用分析两个步骤
D. 面向对象分析需要识别对象的内部和外部特征
104．在一个类的定义中，包含有（　）成员的定义。
A. 数据　B. 函数　C. 数据和函数　D. 数据或函数
105．在类作用域中能够通过直接使用该类的（　）成员名进行访问。
A. 私有　B. 公用　C. 保护　D. 任何
106．在关键字 public 后面定义的成员为类的（　）成员。
A. 私有　B. 公用　C. 保护　D. 任何
107．在关键字 private 后面定义的成员为类的（　）成员。
A. 私有　B. 公用　C. 保护　D. 任何
108．假定 AA 为一个类，a 为该类公有的数据成员，x 为该类的一个对象，则访问 x 对象中数据成员 a 的格式为（　）。
A. x(a)　B. x[a]　C. x->a　D. x.a
109．假定 AA 为一个类，a()为该类公有的函数成员，x 为该类的一个对象，则访问 x 对象中函数成员 a()的格式为（　）。
A. x.a　B. x.a()　C. x->a　D. x->a()
110．假定 AA 为一个类，a 为该类公有的数据成员，px 为指向该类对象的一个指针，则访问 px 所指对象中数据成员 a 的格式为（　）。
A. px(a)　B. px[a]　C. px->a　D. px.a
111．假定 AA 为一个类，a 为该类私有的数据成员，GetValue()为该类公有函数成员，它返回 a 的值，x 为该类的一个对象，则访问 x 对象中数据成员 a 的格式为（　）。
A. x.a　B. x.a()　C. x->GetValue()　D. x.GetValue()
112．假定 AA 为一个类，int a()为该类的一个成员函数，若该成员函数在类定义体外定义，则函数头为（　）。
A. int AA::a()　　B. int AA:a()
C. AA::a()　　D. AA::int a()
113．假定 AA 为一个类，a 为该类公有的数据成员，若要在该类的一个成员函数中访问它，则书写格式为（　）。
A. a　B. AA::a　C. a()　D. AA::a()
114．若需要把一个类外定义的成员函数指明为内联函数，则必须把关键字（　）放在函数原型或函数头的前面。
A. in　B. inline　C. inLine　D. InLiner
115．在多文件结构的程序中，通常把类的定义单独存放于（　）中。
A. 主文件　B. 实现文件　C. 库文件　D. 头文件
116．在多文件结构的程序中，通常把类中所有非内联函数的定义单独存放于（　）中。
A. 主文件　B. 实现文件　C. 库文件　D. 头文件
117．在多文件结构的程序中，通常把含有 main()函数的文件称为（　）。
A. 主文件　B. 实现文件　C. 程序文件　D. 头文件

118．一个 C++程序文件的扩展名为（　　）。
A. .h　　B. .c　　C. .cpp　　D. .cp
119．在 C++程序中使用的 cin 标识符是系统类库中定义的（　　）类中的一个对象。
A. istream　　B. ostream　　C. iostream　　D. fstream
120．在 C++程序中使用的 cout 标识符是系统类库中定义的（　　）类中的一个对象。
A. istream　　B. ostream　　C. iostream　　D. fstream
121．假定 AA 是一个类，abc 是该类的一个成员函数，则参数表中隐含的第一个参数的类型为（　　）。
A. int　　B. char　　C. AA　　D. AA*
122．假定 AA 是一个类，abc 是该类的一个成员函数，则参数表中隐含的第一个参数为（　　）。
A. abc　　B. *this　　C. this　　D. this&
123．假定 AA 是一个类，“AA& abc();”是该类中一个成员函数的原型，若该函数存在对*this 赋值的语句，当用 x.abc()调用该成员函数后，x 的值（　　）。
A. 已经被改变　　B. 可能被改变
C. 不变　　D.不受函数调用的影响
124．假定 AA 是一个类，“AA* abc()const;”是该类中一个成员函数的原型，若该函数返回 this 值，当用 x.abc()调用该成员函数后，x 的值（　　）。
A. 已经被改变　　B. 可能被改变
C. 不变　　D. 受到函数调用的影响
125．类中定义的成员默认为（　　）访问属性。
A. public　　B. private　　C. protected　　D. friend
126．结构中定义的成员默认为（　　）访问属性。
A. public　　B. private　　C. protected　　D. friend
127．当类中一个字符指针成员指向具有 n 个字节的存储空间时，它所能存储字符串的最大长度为(　　)。
A. n　　B. n+1　　C. n-1　　D. n-2
128．在一个用数组实现的队列类中，假定数组长度为 MS，队首元素位置为 first，队列长度为 length，则队尾（即最后一个）元素的位置为(　　)。
A. length+1　　B. first+length
C. (first+length-1)%MS　　D. (first+length)%MS
129．在一个用数组实现的队列类中，假定数组长度为 MS，队首元素位置为 first，队列长度为 length，则队尾的后一个位置为(　　)。
A. length+1　　B. first+length
C. (first+length-1)%MS　　D. (first+length)%MS
130．在一个用数组实现的队列类中，假定数组长度为 MS，队首元素位置为 first，队列长度为 length，则队首的后一个位置为(　　)。
A. first+1　　B. (first+1)%MS
C. (first-1)%MS　　D. (first+length)%MS
131．在一个用链表实现的队列类中，假定每个结点包含的值域用 elem 表示，包含的指针域用 next 表示，链队的队首指针用 elemHead 表示，队尾指针用 elemTail 表示，若链队非空，则进行插入时必须把新结点的地址赋给(　　)。
A. elemHead　　B. elemTail
C. elemHead->next 和 elemHead　　D. elemTail->next 和 elemTail
132．在一个用链表实现的队列类中，假定每个结点包含的值域用 elem 表示，包含的指针域用 next 表示，链队的队首指针用 elemHead 表示，队尾指针用 elemTail 表示，若链队为空，则进行插入时必须把新结点的地址赋给(　　)。
A. elemHead　　B. elemTail

C. elemHead 和 elemTail　　D. elemHead 或 elemTail
133．队列具有（　）的操作特性。
A. 先进先出　　B. 先进后出　　C. 进出无序　　D. 进出任意
134．栈具有（　）的操作特性。
A. 先进先出　　B. 先进后出　　C. 进出无序　　D. 进出任意
135．对于一个类的构造函数，其函数名与类名(　　)。
A. 完全相同　　B. 基本相同　　C. 不相同　　D. 无关系
136．对于一个类的析构函数，其函数名与类名(　　)。
A. 完全相同　　B. 完全不同　　C. 只相差一个字符　　D. 无关系
137．类的构造函数是在定义该类的一个(　　)时被自动调用执行的。
A. 成员函数　　B. 数据成员　　C. 对象　　D. 友元函数
138．类的析构函数是一个对象被(　　)时自动调用的。
A. 建立　　B. 撤消　　C. 赋值　　D. 引用
139．一个类的构造函数通常被定义为该类的(　　)成员。
A. 公用　　B. 保护　　C. 私有　　D. 友元
140．一个类的析构函数通常被定义为该类的(　　)成员。
A. 私有　　B. 保护　　C. 公用　　D. 友元
141．假定 AB 为一个类，则执行 “AB x;”语句时将自动调用该类的(　　)。
A. 带参构造函数　　B. 无参构造函数
C. 拷贝构造函数　　D. 赋值重载函数
142．假定 AB 为一个类，则执行 “AB x(a,5);”语句时将自动调用该类的(　　)。
A. 带参构造函数　　B. 无参构造函数
C. 拷贝构造函数　　D. 赋值重载函数
143．假定 AB 为一个类，则执行 “AB *s=new AB(a,5);”语句时得到的一个动态对象为_____。
A. s　　B. s->a　　C. s.a　　D. *s
144．假定 AB 为一个类，则执行 “AB r1=r2;”语句时将自动调用该类的(　　)。
A. 无参构造函数　　B. 带参构造函数
C. 赋值重载函数　　D. 拷贝构造函数
145．若需要使类中的一个指针成员指向一块动态存储空间，则通常在(　　)函数中完成。
A. 析构　　B. 构造　　C. 任一成员　　D. 友元
146．当类中的一个整型指针成员指向一块具有 n*sizeof(int)大小的存储空间时，它最多能够存储(　　)个整数。
A. n　　B. n+1　　C. n-1　　D. 1
147．假定一个类的构造函数为 “A(int aa, int bb) {a=aa; b=aa*bb;}”，则执行 “A x(4,5);”语句后，x.a 和 x.b 的值分别为(　　)。
A. 4 和 5　　B. 5 和 4　　C. 4 和 20　　D. 20 和 5
148．假定一个类的构造函数为 “A(int aa=1, int bb=0) {a=aa; b=bb;}”，则执行 “A x(4);”语句后，x.a 和 x.b 的值分别为(　　)。
A. 1 和 0　　B. 1 和 4　　C. 4 和 1　　D. 4 和 0
149．假定 AB 为一个类，则(　　)为该类的拷贝构造函数的原型说明。
A. AB(AB x);　　B. AB(AB& x);
C. void AB(AB& x);　　D. AB(int x);
150．假定一个类的构造函数为 “B(int ax, int bx): a(ax), b(bx) {}”，执行 “B x(1,2),y(3,4);x=y;”语句序列后 x.a 的值为(　　)。
A. 1　　B. 2　　C. 3　　D. 4

151．假定一个类 AB 只含有一个整型数据成员 a，当用户不定义任何构造函数时，系统为该类定义的无参构造函数为(　　)。

A. AB() {a=0;}　　B. AB(int aa=0): a(aa) {}

C. AB(int aa): a(aa) {}　　D. AB() {}

152．假定一个类 AB 只含有一个整型数据成员 a，用户为该类定义的带参构造函数可以为(　　)。

A. AB() {}　　B. AB(): a(0){}

C. AB(int aa=0) {a=aa;}　　D. AB(int aa) {}

153．对于任一个类，用户所能定义的构造函数的个数至多为(　　)。

A. 0　　B. 1　　C. 2　　D. 任意个

154．对于任一个类，用户所能定义的析构函数的个数至多为(　　)。

A. 0　　B. 1　　C. 2　　D. 任意个

155．假定 AB 为一个类，则执行 “AB *px=new AB[n];”语句时将(　　)。

A. 动态分配一个数组　　B. 动态分配一个对象

C. 静态分配一个数组　　D. 静态分配一个对象

156．设 px 是指向一个类对象的指针变量，则执行 “delete px;”语句时，将自动调用该类的(　　)。

A. 无参构造函数　　B. 带参构造函数

C. 析构函数　　D. 拷贝构造函数

157．当一个类对象离开它的作用域时，系统自动调用该类的(　　)。

A. 无参构造函数　　B. 带参构造函数

C. 拷贝构造函数　　D. 析构函数

158．假定一个类对象数组为 A[n]，当离开它定义的作用域时，系统自动调用该类析构函数的次数为(　　)。

A. 0　　B. 1　　C. n　　D. n-1

159．假定 AB 为一个类，则执行 “AB a[10];”语句时调用该类无参构造函数的次数为(　　)。

A. 0　　B. 1　　C. 9　　D. 10

160．假定 AB 为一个类，则执行 “AB *px=new AB[n];”语句时调用该类无参构造函数的次数为(　　)。

A. n　　B. n-1　　C. 1　　D. 0

161．假定 AB 为一个类，则执行 “AB a, b(3), *p;”语句时共调用该类构造函数的次数为(　　)。

A. 2　　B. 3　　C. 4　　D. 5

162．假定 AB 为一个类，则执行 “AB a(2), b[3], *p[4];”语句时共调用该类构造函数的次数为(　　)。

A. 3　　B. 4　　C. 5　　D. 9

163．．假定 AB 为一个类，则执行 “AB a, b(2), c[3], *p=&a;”语句时共调用该类无参构造函数的次数为(　　)。

A. 5　　B. 6　　C. 3　　D. 4

164．假定 AB 为一个类，则执行 “AB *p=new AB(1,2);”语句时共调用该类构造函数的次数为(　　)。

A. 0　　B. 1　　C. 2　　D. 3

165．假定 AB 为一个类，px 为指向该类的一个含有 n 个对象的动态数组的指针，则执行 “delete []px;”语句时共调用该类析构函数的次数为(　　)。

A. 0　　B. 1　　C. n　　D. n+1

166．对类对象成员的初始化是通过构造函数中给出的(　　)实现的。

A. 函数体　　B. 初始化表

C. 参数表　　D. 初始化表或函数体

167．对类中常量成员的初始化是通过构造函数中给出的(　　)实现的。

A. 函数体　　B. 参数表

C. 初始化表　　D. 初始化表或函数体

168. 对类中引用成员的初始化是通过构造函数中给出的(　　)实现的。
A. 函数体　　B. 参数表
C. 初始化表　　D. 初始化表或函数体
169. 类的构造函数可以带有(　　)个参数。
A. 0　　B. 1　　C. 2　　D. 任意
170. 类的析构函数可以带有(　　)个参数。
A. 0　　B. 1　　C. 2　　D. 任意
171. 一个类的静态数据成员所表示属性 (　　)。
A. 是类的或对象的属性　　B. 只是对象的属性
C. 只是类的属性　　D. 类和友元的属性
172. 类的静态成员的访问控制（　　）。
A. 只允许被定义为 private
B. 只允许被定义为 private 或 protected
C. 只允许被定义为 public
D. 可允许被定义为 private、protected 或 public
173. 静态成员函数对类的数据成员访问（　　）。
A. 是不允许的
B. 只允许是静态数据成员
C. 只允许是非静态数据成员
D. 可允许是静态数据成员或非静态数据成员
174. 被非静态成员函数访问的类的数据成员(　)。
A. 可以是非静态数据成员或静态数据成员　　B. 不可能是类的静态数据成员
C. 只能是类的非静态数据成员　　D. 只能是类的静态数据成员
175. 静态数据成员的初始化是在（　　）中进行的。
A. 构造函数　　B. 任何成员函数
C. 所属类　　D. 全局区
176. 当将一个类 A 或函数 f()说明为另一个类 B 的友元后，类 A 或函数 f()能够直接访问类 B 的（　　）。
A. 只能是公有成员　　B. 只能是保护成员
C. 只能是除私有成员之外的任何成员　　D. 具有任何权限的成员
177. 引入友元的主要目的是为了（　　）。
A. 增强数据安全性　　B. 提高程序的可靠性
C. 提高程序的效率和灵活性　　D. 保证类的封装性
178. 一个类的成员函数也可以成为另一个类的友元函数，这时的友元说明（　　）。
A. 需加上类域的限定　　B. 不需加上类域的限定
C. 类域的限定可加可不加　　D. 不需要任何限定
179. 一个类的友元不是该类的成员，与该类的关系密切，所以它（　　）。
A. 有 this 指针，有默认操作的对象
B. 没有 this 指针，可以有默认操作的对象
C. 有 this 指针，不能执行默认操作
D. 没有 this 指针，也就没有默认操作的对象
180. 在重载一个运算符时，其参数表中没有任何参数，这表明该运算符是（　　）。
A. 作为友元函数重载的 1 元运算符　　B. 作为成员函数重载的 1 元运算符
C. 作为友元函数重载的 2 元运算符　　D. 作为成员函数重载的 2 元运算符
181. 在成员函数中进行双目运算符重载时，其参数表中应带有（　　）个参数。
A. 0　　B. 1　　C. 2　　D. 3

182．双目运算符重载为普通函数时，其参数表中应带有（　　）个参数。
A. 0　　B. 1　　C. 2　　D. 3
183．如果表达式 a+b 中的“+”是作为成员函数重载的运算符，若采用运算符函数调用格式，则可表示为（　　）。
A. a.operator+(b)　　B. b.operator+(a)
C. operator+(a,b)　　D. operator(a+b)
184．如果表达式 a==b 中的“==”是作为普通函数重载的运算符，若采用运算符函数调用格式，则可表示为（　　）。
A. a.operator==(b)　　B. b.operator==(a)
C. operator==(a,b)　　D. operator==(b,a)
185．如果表达式 a++中的“++”是作为普通函数重载的运算符，若采用运算符函数调用格式，则可表示为（　　）。
A. a.operator++()　　B. operator++(a)
C. operator++(a,1)　　D. operator++(1,a)
186．如果表达式++a 中的“++”是作为成员函数重载的运算符，若采用运算符函数调用格式，则可表示为（　　）。
A. a.operator++(1)　　B. operator++(a)
C. operator++(a,1)　　D. a.operator++()
187．关于运算符重载，下列说法正确的是（　　）。
A. 重载时，运算符的优先级可以改变。
B. 重载时，运算符的结合性可以改变。
C. 重载时，运算符的功能可以改变。
D. 重载时，运算符的操作数个数可以改变。
188．关于运算符重载，下列说法正确的是（　　）。
A. 所有的运算符都可以重载。
B. 通过重载，可以使运算符应用于自定义的数据类型。
C. 通过重载，可以创造原来没有的运算符。
D. 通过重载，可以改变运算符的优先级。
189．一个程序中数组 a 和变量 k 定义为“int a[5][10],k;”，且程序中包含有语句“a(2,5)=++k*3;”，则此语句中肯定属于重载操作符的是（　）。
A. ( )　　B. =　　C. ++　　D. *
190．假定 K 是一个类名，并有定义“K k; int j;”，已知 K 中重载了操作符 ( )，且语句“j=k(3);”和“k(5)=99;”都能顺利执行，说明该操作符函数的原形只可能是（　）。
A. K operator ( ) (int);　　B. int operator ( )(int&);
C. int & operator ( )(int);　　D. K &operator( )(int);
191. 假定 M 是一个类名，且 M 中重载了操作符=，可以实现 M 对象间的连续赋值，如“m1=m2=m3;”。重载操作符=的函数原型最好是（　）。
A. int operaotor=(M);　　B.　int operator=(M&);
C. M operator=(M&);　　D.　M& operator=(M);
192．下面是重载双目运算符+的普通函数原形，其中最符合+原来含义的是（　　）。
A. Value operator+(Value, Value);　　B. Value operator+(Value,int);
C. Value &operator+(Value, Value);　　D. Value &operator+(Value&, Value&);
193．下面是重载双目运算符-的成员函数原形，其中最符合-原来含义的是（　　）。
A. Value Value::operator-(Value);
B. Value Value::operator-(int);

C. Value& Value::operator-(Value);
D. Value& Value::operator-(Value&);
194．在重载一运算符时，若运算符函数的形参表中没有参数，则不可能的情况是（　　）。
A. 该运算符是一个单目运算符。
B. 该运算符函数有一个隐含的参数 this。
C. 该运算符函数是类的成员函数。
D. 该运算符函数是类的友元函数。
195．关于插入运算符<<的重载，下列说法不正确的是（　　）。
A. 运算符函数的返回值类型是 ostream & 。
B. 重载的运算符必须定义为类的成员函数。
C. 运算符函数的第一个参数的类型是 ostream & 。
D. 运算符函数有两个参数。
196．从一个基类派生出的各个类的对象之间(　　)。
A. 共享所有数据成员，每个对象还包含基类的所有属性
B. 共享部分数据成员，每个对象还包含基类的所有属性
C. 不共享任何数据成员，但每个对象还包含基类的所有属性
D. 共享部分数据成员和函数成员
197．如果是类 B 在类 A 的基础上构造，那么，就称（　　）。
A. 类 A 为基类或父类，类 B 为超类或子类
B. 类 A 为基类、父类或超类，类 B 为派生类或子类
C. 类 A 为派生类，类 B 为基类
D. 类 A 为派生类或子类，类 B 为基类、父类或超类
198．C++的继承性允许派生类继承基类的（　　）。
A. 部分特性，并允许增加新的特性或重定义基类的特性
B. 部分特性，但不允许增加新的特性或重定义基类的特性
C. 所有特性，并允许增加新的特性或重定义基类的特性
D. 所有特性，但不允许增加新的特性或重定义基类的特性
199．派生类的成员函数可以直接访问基类的（　　）成员。
A. 所有　　B. 公有和保护　　C. 保护和私有　　D. 私有
200．对于公有继承，基类的公有和保护成员在派生类中将(　　)成员。
A. 全部变成公有　　B. 全部变成保护
C. 全部变成私有　　D. 仍然相应保持为公有和保护
201．对于公有继承，基类中的私有成员在派生类中将（　　）。
A. 能够直接使用成员名访问　　B. 能够通过成员运算符访问
C. 仍然是基类的私有成员　　D. 变为派生类的私有成员
202．当保护继承时，基类的（　　）在派生类中成为保护成员，在类作用域外不能够通过派生类的对象来直接访问该成员。
A. 任何成员　　B.公有成员和保护成员
C. 保护成员和私有成员　　D.私有成员
203．在定义一个派生类时，若不使用保留字显式地规定采用何种继承方式，则默认为（　　）方式。
A. 私有继承　　B. 非私有继承
C. 保护继承　　D. 公有继承
204．建立包含有类对象成员的派生类对象时，自动调用构造函数的执行顺序依次为（　　）的构造函数。
A. 自己所属类、对象成员所属类、基类
B. 对象成员所属类、基类、自己所属类

C. 基类、对象成员所属类、自己所属类
D. 基类、自己所属类、对象成员所属类
205．当派生类中有和基类一样名字的成员时，一般来说，（　）。
A. 将产生二义性　　B. 派生类的同名成员将覆盖基类的成员
C. 是不能允许的　　D. 基类的同名成员将覆盖派生类的成员
206．C++中的虚基类机制可以保证：（　）。
A. 限定基类只通过一条路径派生出派生类
B. 允许基类通过多条路径派生出派生类，派生类也就能多次继承该基类
C. 当一个类多次间接从基类派生以后，派生类对象能保留多份间接基类的成员
D. 当一个类多次间接从基类派生以后，其基类只被一次继承
207．下列对派生类的描述中错误的说法是：（　　）。
A. 派生类至少有一个基类
B. 派生类可作为另一个派生类的基类
C. 派生类除了包含它直接定义的成员外，还包含其基类的成员
D. 派生类所继承的基类成员的访问权限保持不变
208．派生类的对象对其基类中（　　）可直接访问。
A. 公有继承的公有成员
B. 公有继承的私有成员
C. 公有继承的保护成员
D. 私有继承的公有成员

## 二.填空题

1．C++语言是在_____语言的基础上发展起来的。
2．C++语言的编译单位是扩展名为_____的_____文件。
3．行尾使用注释的开始标记符为_____。
4．多行注释的开始标记符和结束标记符分别为_____和_____。
5．用于输出表达式值的标准输出流对象是_____。
6．用于从键盘上为变量输入值的标准输入流对象是_____。
7．一个完整程序中必须有一个名为_____的函数。
8．一个函数的函数体就是一条_____语句。
9．当执行 cin 语句时，从键盘上输入每个数据后必须接着输入一个_____符，然后才能继续输入下一个数据。
10．在 C++程序中包含一个头文件或程序文件的预编译命令为_____。
11．程序中的预处理命令是指以_____字符开头的命令。
12．一条表达式语句必须以_____作为结束符。
13．在#include 命令中所包含的头文件，可以是系统定义的头文件，也可以是_____定义的头文件。
14．使用#include 命令可以包含一个头文件，也可以包含一个_____文件。
15．一个函数定义由_____和_____两部分组成。
16．若一个函数的定义处于调用它的函数之前，则在程序开始可以省去该函数的_____语句。
17．C++头文件和源程序文件的扩展名分别为_____和_____。
18．程序文件的编译错误分为_____和_____两类。
19．当使用_____保留字作为函数类型时，该函数不返回任何值。
20．当函数参数表用_____保留字表示时，则表示该参数表为空。
21．从一条函数原型语句“int fun1(void);”可知，该函数的返回类型为_____，该函数带有_____个参数。
22．当执行 cout 语句输出 endl 数据项时，将使 C++显示输出屏幕上的光标从当前位置移动到_____的开始位

置。

23. 假定 x=5，y=6，则表达式 x++*++y 的值为_____。

24. 假定 x=5，y=6，则表达式 x--*--y 的值为_____。

25. 假定 x=5，y=6，则执行表达式 y*=x++计算后，x 和 y 的值分别为_____和_____。

26. 假定 x=5，y=6，则执行表达式 y+=x--计算后，x 和 y 的值分别为_____和_____。

27. C++常数 0x145 对应的十进制值为_____。

28. C++常数 0345 对应的十进制值为_____。

29. 十进制常数 245 对应的十六进制的 C++表示为_____。

30. 十进制常数 245 对应的八进制的 C++表示为_____。

31. signed char 类型的值域范围是_____至_____之间的整数。

32. int 和 float 类型的数据分别占用_____和_____个字节。

33. float 和 double 类型的数据分别占用_____和_____个字节。

34. bool 和 char 类型的数据分别占用_____和_____个字节。

35. unsigned short int 和 int 类型的长度分别为_____和_____。

36. 字符串”This\’s a book.\n”的长度为_____。

37. 字符串”\nThis\’s a pen\n\n”的长度为_____。

38. 在 C++中存储字符串”abcdef”至少需要_____个字节。

39. 在 C++中存储字符串”a+b=c”至少需要_____个字节。

40. 假定 x 和 y 为整型，其值分别为 16 和 5，则 x%y 和 x/y 的值分别为_____和_____。

41. 假定 x 和 y 为整型，其值分别为 16 和 5，则 x/y 和 double(x)/y 的值分别为_____和_____。

42. 假定 x 是一个逻辑量，则 x && true 的值为_____。

43. 假定 x 是一个逻辑量，则 x || true 的值为_____。

44. 假定 x 是一个逻辑量，则 x && false 的值为_____。

45. 假定 x 是一个逻辑量，则 x || false 的值为_____。

46. 假定 x 是一个逻辑量，则!x || false 的值为_____。

47. 假定 x 是一个逻辑量，则 x && !x 的值为_____。

48. 假定 x 是一个逻辑量，则 x || !x 的值为_____。

49. 设 enum Printstatus{ready,busy,error}; 则 cout<<busy 的输出结果是_____。

50. 设 enum Printstatus{ready=2,busy,error}; 则 cout<<busy 的输出结果是_____。

51. 常数-4.205 和 6.7E-9 分别具有_____和_____位有效数字。

52. 枚举类型中的每个枚举值都是一个_____，它的值为一个_____。

53. 常数 100 和 3.62 的数据类型分别为_____和_____。

54. 若 x=5, y=10, 则计算 y*=++x 表达式后，x 和 y 的值分别为_____和_____。

55. 假定 x 和 ch 分别为 int 型和 char 型，则 sizeof(x)和 sizeof(ch)的值分别为_____和_____。

56. 假定 x=10，则表达式 x<=10?20:30 的值为_____。

57. 表达式 sqrt(81)和 pow(6,3)的值分别为_____和_____。

58. 含随机函数的表达式 rand()%20 的值在_____至_____区间内。

59. 在 switch 语句中，每个语句标号所含关键字 case 后面的表达式必须是_____。

60. 在 if 语句中，每个 else 关键字与它前面同层次并且最接近的_____关键字相配套。

61. 作为语句标号使用的 C++保留字 case 和 defaule 只能用于_____语句的定义体中。

62. 执行 switch 语句时，在进行作为条件的表达式求值后，将从某个匹配的标号位置起向下执行，当碰到下一个标号位置时（停止/不停止）_____执行。

63. 若 while 循环的“头”为“while(i++<=10)”，并且 i 的初值为 0，同时在循环体中不会修改 i 的值，则循环体将被重复执行_____次后正常结束。

64. 若 do 循环的“尾”为“while(++i<10)”，并且 i 的初值为 0，同时在循环体中不会修改 i 的值，则循环体将被

重复执行_____次后正常结束。

65．当在程序中执行到_____语句时，将结束本层循环类语句或 switch 语句的执行。

66．当在程序中执行到_____语句时，将结束所在循环语句中循环体的一次执行。

67．在程序中执行到_____语句时，将结束所在函数的执行过程，返回到调用该函数的位置。

68．在程序执行完_____函数调用后，将结束整个程序的执行过程，返回到 C++集成开发窗口。

69．元素类型为 int 的数组 a[10]共占用_____字节的存储空间。

70．元素类型为 double 的二维数组 a[4][6]共占用_____字节的存储空间。

71．元素类型为 char 的二维数组 a[10][30]共占用_____字节的存储空间。

72．存储字符’a’和字符串”a”分别需要占用_____和_____个字节。

73．空串的长度为_____，存储它需要占用_____个字节。

74．字符串”\’a\’xy=4\n”的长度为_____。

75．字符串”a:\\xxk\\数据”的长度为_____。

76．用于存储一个长度为 n 的字符串的字符数组的长度至少为_____。

77．若 a 是一个字符数组，则从键盘上向该数组输入一个字符串的表达式为_____。

78．若 a 是一个字符数组，则向屏幕输出 a 中所存字符串的表达式为_____。

79．一个二维字符数组 a[10][20]能够存储_____个字符串，每个字符串的长度至多为_____。

80.对一个二维字符数组 a 进行初始化的数据为{”123”,”456”,”789”},则 a[1]元素对应的字符串为_____。

81．若需要把一个字符串”aaa”赋值到字符数组 a 中，则需要执行_____函数的调用来实现。

82．假定对数组 a[]进行初始化的数据为{2,7,9,6,5,7,10}，则 a[2]和 a[5]分别被初始化为_____和_____。

83．假定对二维数组 a[3][4]进行初始化的数据为{{3,5,6},{2,8},{7}}，则 a[1][1]和 a[2][3]分别被初始化为_____和_____。

84．在 C++语言中，一个函数由函数头和_____组成。

85．重载一个函数的条件是：该函数必须在参数的个数或参数的_____上与其它同名函数有所不同。

86．如果一个函数只允许同一程序中的函数调用，则应在该函数定义前加上_____C++保留字。

87．若“double x=100;”是文件 F1.CPP 中的一个全局变量定义语句，若文件 F2.CPP 中的某个函数需要访问此 x，则应在文件 F2.CPP 中添加对 x 的声明语句为_____。

88．定义一个函数模板要用到的第一个修饰符是_____。

89．在函数模板的参数中，用 class 修饰的参数称为_____参数。

90．如果一个函数直接或间接地调用自身，这样的调用称为_____调用。

91．已知 int cubin(int n){return n*n*n;}和 double cubin(double n){return n*n*n;}是一个函数模板的两个实例，假定类型参数用 T 表示，则该函数模板的定义是_____。

92．对于无返回值函数，定义函数时要用_____修饰函数类型。

93．如果一个函数定义中使用了_____修饰，则该函数不允许被其它文件中的函数调用。

94．如果一个函数中有多个默认参数，则默认参数必须全部处在形参表的_____部分。

95．定义外部变量时，不用存储类说明符_____，而声明外部变量时用它。

96．调用系统函数时，要先使用#include 命令包含该系统函数的原型语句所在的_____。

97．函数形参的作用域是该函数的_____。

98．C++提供的预处理命令有宏定义命令，条件编译命令和_____。

99．程序的编译是以_____为单位进行的。

100．C++程序运行时的内存空间可以分成全局数据区，堆区，栈区和_____。

101．全局变量和静态局部变量具有静态生存期，存放在内存的_____区中。

102．局部变量具有局部生存期，存放在内存的_____区中。

103．若二维数组 a 有 m 列，设 a[0][0]位于数组的第一个位置上，则计算任一元素 a[i][j]在数组中位置序号的公式为_____。

104．若有定义“double a[3][5];”，则 a 数组中行下标和列下标的最大值分别为_____和_____。

105．若有定义“struct AA {int a; char b; double c;}x;”,则 x 占用空间大小为_____字节。

106．当定义一个结构体变量时，系统分配给该变量的内存大小等于各成员所需内存大小的_____。

107．一个指针类型的对象占用内存的_____个字节的存储空间。

108．一个指针指向一个数据对象，它保存着该数据对象的_____，若数据对象为 DataType 类型，则相应的指针类型为_____。

109．若要把一个整型指针 p 转换为字符指针，则采用的强制转换表达式为_____。

110．假定一个数据对象为 int*类型，则指向该对象的指针类型为_____。

111．假定 p 是一个指向整数对象的指针，则用_____表示该整数对象，用_____表示指针变量 p 的地址。

112．假定 p 是一个指针，则*p++运算首先访问_____，然后使_____的值增 1。

113．假定 p 是一个指针，则(*p)++运算首先访问_____，然后使_____的值增 1。

114．假定 p 所指对象的值为 25，p+1 所指对象的值为 42，则*p++的值为_____。

115．假定 p 所指对象的值为 25，p+1 所指对象的值为 42，则*++p 的值为_____。

116．假定 p 所指对象的值为 25，p+1 所指对象的值为 42，则执行(*p)++运算后，p 所指对象的值为_____。

117．假定 p 所指对象的值为 25，p+1 所指对象的值为 42，则执行*(p++)或*p++运算后，p 所指对象的值为_____。

118．假定 a 是一个一维指针数组，则 a+i 所指对象的地址比 a 大_____字节。

119．假定 a 是一个一维数组，则 a[i]的指针访问方式为_____。

120．假定 a 是一个一维数组，则 a[i]对应的存储地址（以字节为单位）为_____。

121．一个数组的数组名实际上是指向该数组_____元素的指针，并且在任何时候都不允许_____它。

122．假定指向一维数组 b[10]中元素 b[4]的指针为 p，则 p+3 所指向的元素为_____，p-2 所指向的元素为_____。

123．若要定义整型指针 p 并初始指向 x，则所使用的定义语句为_____。

124．若 p 指向 x，则_____与 x 的表示是等价的。

125．在一个二维数组 int a[m][n]中，包含的一维元素 a[i]的类型为_____，访问 a[i]时返回值的类型为_____。

126．假定一个二维数组为 c[5][8]，则 c[3]的值为二维元素_____的地址，c[3]+2 的值为二维元素_____的地址。

127．假定 p 为指向二维数组 int d[4][6]的指针，则 p 的类型为_____。

128．假定 a 是一个二维数组，则 a[i][j]的指针访问方式为_____。

129．若要把整型变量 y 定义为 x 的引用，则所使用的定义语句为_____。

130．若 y 是 x 的引用，则对 y 的操作就是对_____的操作。

131．若 y 是 x 的引用，则&y 和&x 的值_____，即为变量_____的地址。

132．执行 int p=new int 操作得到的一个动态分配的整型对象为_____。

133．执行 int *p=new int[10]操作，使 p 指向动态分配的数组中下标为 0 的元素，该元素可表示为_____或_____。

134．执行 char *p=new char(’a’)操作后，p 所指向的数据对象的值为_____。

135．执行 new char[m][n]操作时的返回值的类型为_____。

136．执行_____操作将释放由 p 所指向的动态分配的数据空间。

137．执行_____操作将释放由 p 所指向的动态分配的数组空间。

138．NULL 是一个符号常量，通常作为空指针值，它的具体值为_____。

139．变量 v 定义为“double v=23.4;”，要使指针 pv 指向 v，则定义 pv 的语句为_____。

140．已知语句“cout<<p;”的输出是“Hello!”，则语句“cout<<*p;”输出的是_____。

141．已知语句“cout<<s;”的输出是“apple”，则执行语句“cout<<s+2;”的输出结果为_____。

142．指针变量 pv 和 pc 定义为“void *pv =” Hello, word!”; char *pc;”，要将 pv 值赋给 pc，则正确的赋值语

句是_____。

143. 数组 b 定义为“int b[20][100];”，要使 p[j][k] 与 b[j][k] 等效，则指针 p 应定义为_____。

144. 与结构成员访问表达式 p->name 等价的表达式是_____。

145. 与结构成员访问表达式(*fp).score 等价的表达式是_____。

146. 已知变量 a 定义为“int a=5;”，要使 ra 成为 a 的引用，则 ra 应定义为_____。

147. 已知有定义“int x, a[]={5,7,9}, *pa=a;”，在执行“x=++*pa;”语句后，x 的值是_____。

148. 已知有定义“int x, a[]={6,10,12}, *pa=a;”，在执行“x=*++pa;”语句后，*pa 的值是_____。

149. 已知有定义“int x, a[]={15,17,19}, *pa=a;”，在执行“x=*pa++;”后，*pa 的值是_____。

150. 以面向对象方法构造的系统，其基本单位是_____。

151. 每个对象都是所属类的一个_____。

152. 对象将其大部分实现细节隐藏起来，这种机制称为_____。

153. 基类和派生类的关系称为_____。

154. 复杂对象可以由简单对象构成，这种现象称为_____。

155. 对象是对问题域中客观事物的_____，它是一组属性和在这些属性上操作的_____。

156. 特殊类的对象拥有其一般类的全部属性与操作，称特殊类_____了一般类。

157. 如果一个派生类的基类不止一个，则这种继承称为_____。

158. 如果一个派生类只有一个唯一的基类，则这样的继承关系称为_____。

159. C++支持两种多态性：_____时的多态性和_____时的多态性。

160. 在 C++中，编译时的多态性是通过_____实现的，而运行时的多态性则是通过_____实现的。

161. 面向对象软件开发的生命周期分为三个阶段，即分析、_____和_____。

162. 面向对象的分析包括_____分析和_____分析两步。

163. 类定义中，既包含数据成员，也包含_____成员。

164. 类中的数据成员的访问属性通常被指明为_____。

165. 类中的供外部调用定义的函数成员，其访问属性通常被定义为_____。

166. 对于类中定义的任何成员，其隐含访问权限为_____。

167. 对于结构中定义的任何成员，其隐含访问权限为_____。

168. 为了使类中的成员不能被类外的函数通过成员操作符访问，则应把该成员的访问权限定义为_____。

169. 若在类的定义体中给出了一个成员函数的完整定义，则该函数属于_____函数。

170. 若在类的定义体中只给出了一个成员函数的原型，则在类外给出完整定义时，其函数名前必须加上_____和两个冒号分隔符。

171. 若在类的定义体中只给出了一个成员函数的原型，则在类外给出完整定义时，其函数名前必须加上类名和两个_____分隔符。

172. 若要把类外定义的成员函数规定为内联函数，则必须把_____关键字放到函数原型或函数头的前面。

173. 把一个类的定义体和所有成员函数的定义体所构成的程序范围叫做该类的_____。

174. 假定 AA 是一个类，“AA* abc();”是该类中一个成员函数的原型，则在类外定义时的函数头为_____。

175. 成员函数的参数表在类作用域中，成员函数的返回值类型_____类作用域中。

176. 为了避免在调用成员函数时修改对象中的任何数据成员，则应在定义该成员函数时，在函数头的后面加上_____关键字。

177. 若只需要通过一个成员函数读取数据成员的值，而不需要修改它，则应在函数头的后面加上_____关键字。

178. 若采用 x.abc(y)表达式调用一个成员函数，在成员函数中使用的_____就代表了类外的 x 对象。

179. 若采用 p->abc(y)表达式调用一个成员函数，在成员函数中使用的_____就代表了类外的 p 指针。

180. 内联函数的定义模块与_____模块必须放在同一个文件中。

181. 假定 AA 是一个类，“AA* abc()const;”是该类中一个成员函数的原型，在该函数体中（能够/不能够）_____向*this 或其成员赋值。

182．在一个用数组实现的队列类中，包含有两个数据成员，一个指明队首元素位置，另一个指明_____。

183．在一个用数组实现的队列类中，包含有两个数据成员，一个指明队列长度，另一个指明_____元素的位置。

184．在一个用数组实现的队列类中，假定数组长度为 MS，队首元素位置为 first，队列长度为 length，则插入一个新元素的位置为_____。

185．在一个用数组实现的队列类中，假定数组长度为 MS，队首元素位置为 first，队列长度为 length，则删除一个元素后队首的位置为_____。

186．在一个用数组实现的队列类中，假定数组长度为 MS，队首元素位置为 first，队列长度为 length，则队列为空的条件为_____。

187．在一个用数组实现的队列类中，假定数组长度为 MS，队首元素位置为 first，队列长度为 length，则队列为满的条件为_____。

188．当一个队列为空时，不能对其做_____元素的操作。

189．当一个队列为满时，不能对其做_____元素的操作。

190．从一个队列中删除元素就是删除_____位置上的元素。

191．向一个队列中插入元素就是把该元素放到_____元素的后一位置上。

192．在一个用链表实现的队列类中，假定每个结点包含的值域用 elem 表示，包含的指针域用 next 表示，链队的队首指针用 elemHead 表示，队尾指针用 elemTail 表示，当链队非空时，_____指向队首结点的后继（即下一个）结点。

193．在一个用链表实现的队列类中，假定每个结点包含的值域用 elem 表示，包含的指针域用 next 表示，链队的队首指针用 elemHead 表示，队尾指针用 elemTail 表示，当链队非空时，新插入结点的地址应当赋给_____所指结点的 next 域。

194．在一个用链表实现的队列类中，队尾结点的指针域的值为_____。

195．在一个用链表实现的队列类中，若链队中只含有一个结点，则队首指针的值与队尾指针的值_____。

196．在一个用链表实现的队列类中，若链队为空或只含有一个结点，则队首指针的值与队尾指针的值_____。

197．在一个用链表实现的队列类中，若队首指针与队尾指针的值不同，则说明链队中至少包含有_____个结点。

198．一个类的_____函数实现对该类对象的初始化功能。

199．一个类的_____函数通常用于实现释放该类对象中指针成员所指向的动态存储空间的任务。

200．当用户为一个类定义有_____时，则系统不会为该类再自动生成一个默认构造函数。

201．假定用户没有给一个名为 AB 的类定义构造函数，则系统为其定义的构造函数为_____。

202．假定用户没有给一个名为 AB 的类定义析构函数，则系统为其定义的析构函数为_____。

203．定义类动态对象数组时，其元素只能靠自动调用该类的_____来进行初始化。

204．在一个类中定义拷贝构造函数的目的，是为了当利用该类的一个对象初始化另一个对象时，能够避免这两个对象的同一指针同时指向同一块_____。

205．为了释放类对象中指针成员所指向的动态存储空间，则需要为该类定义_____。

206．假定 AB 为一个类，则执行“AB a[10];”语句时，系统自动调用该类构造函数的次数为_____。

207．假定一个类对象数组为 A[N]，当离开它的作用域时，系统自动调用该类析构函数的次数为_____。

208．对类中对象成员的初始化是通过在构造函数中给出的_____来实现的。

209．对类中常量成员的初始化是通过在构造函数中给出的_____来实现的。

210．对类中引用成员的初始化只能通过在构造函数中给出的_____来实现。

211．对类中一般数据成员的初始化既可以通过在构造函数中给出的初始化表来实现，也可以通过构造函数中的_____来实现。

212．假定要把 aa 定义为 AB 类中的一个常量整数型数据成员，则定义语句为_____。

213．假定要把 aa 定义为 AB 类中的一个引用整数型数据成员，则定义语句为_____。

214．假定 AB 类中只包含一个整型数据成员 a，并且它是一个常量成员，若利用参数 aa 对其进行初始化，则该类的构造函数的定义为_____。

215．假定 AB 类中只包含一个整型数据成员 a，并且它是一个引用成员，若利用引用参数 aa 对其进行初始化，则该类的构造函数的定义为_____。

216．假定指针 p 指向一个动态分配的类对象，则当执行“delete p;”语句时，在释放 p 所指向的动态存储空间之前将自动调用该类的_____。

217．假定用户为类 AB 定义了一个构造函数“AB(int aa) {a=aa;}”，则系统（会/不会）_____为该类自动定义一个无参构造函数"AB() {}"。

218．假定用户为类 AB 定义了一个构造函数“AB(int aa, char *bb=NULL):a(aa),b(bb){}”，则该类中至少包含有_____个数据成员。

219．假定用户为类 AB 定义了一个构造函数“AB(int aa) {a=aa;}”，该构造函数实现对数据成员_____的初始化。

220．假定用户为类 AB 定义了一个构造函数“AB(int aa=0):a(aa){}”，则定义该类的对象时，可以有_____种不同的定义格式。

221．假定用户为类 AB 定义了一个构造函数“AB(int aa):a(aa){}”，则定义该类的对象时，有_____种定义格式。

222．假定用户只为类 AB 定义了一个构造函数“AB(int aa, int bb=0) {a=aa; b=bb;}”，则定义该类的对象时，其实参表中至多带有_____个实参。

223．假定用户只为类 AB 定义了一个构造函数“AB(int aa, int bb=0) {a=aa; b=bb;}”，则定义该类的对象时，其实参表中至少带有_____个实参。

224．假定用户为类 AB 定义了一个构造函数“AB(int aa=0, int bb=0) {a=aa; b=bb;}”，则定义该类的对象时，可以有_____种不同的定义格式。

225．假定用户只为类 AB 定义了一个构造函数“AB():a(0),b(0){}”，则定义该类对象 x 的定义语句“AB x();”是_____(正确/错误)的。

226．假定用户只为类 AB 定义了一个构造函数“AB():a(0),b(0){}”，则定义该类对象 x 的定义语句“AB x;”是_____(正确/错误)的。

227．假定用户只为类 AB 定义了一个构造函数“AB():a(0),b(0){}”，则定义该类对象 x 的定义语句“AB x(5);”是_____(正确/错误)的。

228．假定 AB 为一个类，则类定义体中的“AB(AB& x);”语句为该类_____的原型语句。

229．假定 AB 为一个类，则该类的拷贝构造函数的函数头为_____。

230．假定 AB 为一个类，该类中含有一个指向动态数组空间的指针成员 pa，则在该类的析构函数中应该包含有一条_____语句。

231．静态成员函数_____访问类的静态数据成员， _____ 访问类的非静态数据成员。

232．静态数据成员必须在所有函数的定义体外进行_____。

233．一个类的成员函数也可以成为另一个类的友元函数，这时的友元说明必须在函数名前加上_____的限定。

234．重载运算符时，该运算符的_____、结合性以及操作符的个数不允许改变。

235．一个单目运算符作为类的成员函数重载时有_____个参数；如果作为独立函数重载，则有_____个参数。

236．一个双单目运算符作为类的成员函数重载时有_____个参数；如果作为独立函数重载，则有_____个参数。

237．除了_____运算符外，其他重载的运算符都可以被派生类继承。

238．作为类的成员函数重载一个运算符时，参数表中只有一个参数，说明该运算符有_____个操作数。

239．在重载一个单目运算符时，参数表中没有参数，说明该运算符函数只能是类的_____。

240．重载插入运算符<<时，其运算符函数的返回值类型应当是_____。

241．重载抽取运算符>>时，其运算符函数的返回值类型应当是_____。

242．重载插入运算符<<或抽取运算符>>时，其运算符函数的参数有_____个。

243．重载插入运算符<<或抽取运算符>>时，其运算符函数不能被定义为类的_____函数。

244．类型转换函数没有_____类型，而且参数表为_____。

245．在一个或若干个类的基础上构造一个新类，被称为_____。

246．派生类的成员函数可以直接访问基类的_____成员，不能直接访问基类的_____成员。

247．当保护继承时，基类的_____成员在派生类中成为保护成员，派生类对象不能直接访问基类的_____成员。

248．在定义一个派生类时，使用_____关键字或者不显式地使用它则表示为_____继承。

249．若多个基类及其派生类中都定义了同名函数成员，要访问相应函数时，就需要在函数名前加上_____和类区分符。

250．若要保证一个公共的基类在派生类中只产生一个基类子对象，则必须都以_____的方式直接继承它。

251．引进虚基类的根本目的是为了消除_____。

252．在每个成员函数中，隐含的第一个参数的参数名为_____。

# 三.程序填充

1．斐波那契数列的第 1 和第 2 个数分别为 0 和 1 ，从第三个数开始，每个数等于其前两个数之和。求斐波那契数列中的前 20 个数，要求每行输出 5 个数。

```
#include<iostream.h>
void main()
{
  int f,f1,f2,i;
  cout<<" 斐波那契数列：\n";
  f1=0; f2=1;
  cout<<setw(6)<<f1<<setw(6)<<f2;
  for(i=3;i<=20;i++)
  {
      f=_______(1)_______;
      cout<<setw(6)<<f;
      if(_____(2)_______) cout<<endl;
      f1=f2;
      f2=_____(3)________;
   }
   cout<<endl;
}
```

(1)　　(2)　　(3)

2．打印出 2 至 99 之间的所有素数(即不能被任何数整除的数)。

```
#include<iostream.h>
#include<math.h>
void main()
{
    int i,n;
    for(n=2; ___(1)___; n++) {
        int temp=int(sqrt(n));  //求出 n 的平方根并取整
        for(i=2; ___(2)___; i++)
```

```
            if(n%i==0) ___(3)___;
        if(i>temp) cout<<n<<' ';
    }
    cout<<'\n';
}
```

(1)　　(2)　　(3)

3．采用辗转相除法求出两个整数的最大公约数。

```
#include<iostream.h>
void main()
{
    int a,b;
    cout<<"请输入两个正整数:";
    cin>>a>>b;
    while(a<=0 || __(1)___) {cout<<"重新输入:"; cin>>a>>b;}
    while(b) {
        int r;
        r=a%b;
        ___(2)___; ___(3)___;  //分别修改 a 和 b 的值
    }
    cout<<a<<endl;  //输出最大公约数
}
```

(1)　　(2)　　(3)

4．把从键盘上输入的一个大于等于 3 的整数分解为质因子的乘积。如输入 24 时得到的输出结果为“2 2 2 3”，输入 50 时得到的输出结果为“2 5 5”，输入 37 时得到的输出结果为“37”。

```
#include<iostream.h>
void main()
{
    int x;
    cout<<"请输入一个整数，若小于 3 则重输:";
    do cin>>x; while(___(1)___);
    int i=2;
    do{
        while(___(2)___) {
            cout<<i<<' ';
            x/=i;
        }
        ___(3)___;
    }while(i<x);
    if(x!=1) cout<<x;
    cout<<endl;
}
```

(1)　　(2)　　(3)

5．下面函数是求两个整型参数 a 和 b 的最小公倍数。

```
int f2(int a, int b)
{
    int i=2, p=1;
    do {
        while(a%i==0 && ___(1)___) {
            p*=i; a/=i; b/=i;
        }
        ___(2)___;
    }while(a>=i && ___(3)___);
    return p*a*b;
}
```

(1)　　(2)　　(3)

6．在输出屏幕上打印出一个由字符’*’组成的等腰三角形，该三角形的高为 5 行，从上到下每行的字符数依次为 1,3,5,7,9。

```
#include<iostream.h>
void main()
{
    int i,j;
    for(i=1;___(1)___;i++) {
        for(j=1;j<=9;j++)
            if(j<=5-i || ___(2)___) cout<<’ ’;
            else ___(3)___;
        cout<<endl;
    }
}
```

(1)　　(2)　　(3)

7．统计字符串中英文字母个数的程序。

```
#include <iostream.h>
int count (char str[]);
void main(){
    char s1[80];
    cout <<” Enter a line:”;
    cin >>s1;
    cout <<” count=” <<count(s1)<<endl;
}
int count(char str[]){
    int num=0;  //给统计变量赋初值
```

```
    for(int i=0;str[i];i++)
    if (str[i]>=’a’ && str[i]<=’z’ ||___(1)___ )
          ___(2)___;
     ___(3)___;
}
```

(1)　　(2)　　(3)

8. 主函数调用一个 fun 函数将字符串逆序。

```
#include<iostream.h>
#include<string.h>
___(1)___;
void main( ) {
    char s[80];
    cin>>s;
    ___(2)___;
    cout<<”逆序后的字符串:”<<s<<endl ;
}
void fun(char ss[]) {
    int n=strlen(ss);
    for(int i=0; ___(3)____; i++) {
       char c=ss[i];
       ss[i]=ss[n－1－i];
       ss[n－1－i]=c;
    }
}
```

(1)　　(2)　　(3)

9. 从一个字符串中删除所有同一个给定字符后得到一个新字符串并输出。

```
#include<iostream.h>
const int len=20;
void delstr(char a[],char b[],char c);
void main() {
    char str1[len],str2[len];
    char ch;
    cout<<"输入一个字符串:";
    cin>>str1;
    cout<<"输入一个待删除的字符:";
    cin>>ch;
    delstr(str1,str2,ch);
    cout<<str2<<endl;
}
void delstr(char a[],char b[],char c)
{
```

```
    int j=0;
    for(int i=0; ___(1)___; i++)
        if(___(2)___) b[j++]=a[i];
    b[j]=___(2)___;
}
```

(1)　　(2)　　(3)

10. 采用指针访问方式从键盘给数组 a[N]输入数据，然后对元素值重新按逆序存放并输出。

```
#include <iostream.h>
const int N=8;
void main()
{
    int a[N],*p,*q;
    for(p=a; p<a+N; p++) ___(1)___;
    p=a;q=a+N-1;
    while(p<q) {
        int r=*p; *p=*q; *q=r;
        ___(2)___; ___(3)___;
    }
    for(p=a;p<a+N; p++)
        cout<<*p<<' ';
    cout<<endl;
}
```

(1)　　(2)　　(3)

11. 从键盘上输入一个正整数，然后把它转换成的二进制数的每一位存放到一维数组中，最后输出该二进制数。注意二进制数的存放是按照从低位到高位的次序进行的。

```
#include <iostream.h>
void main()
{
    int x;
    cout<<"输入一个整数:";
    cin>>x;
    int a[20],k=0,r;
    do {
        r=x%2;
        a[k++]=r;
        x=___(1)___;
    } while(___(2)___);
    for(--k;k>=0;k--) ___(3)___;
    cout<<endl;
}
```

(1)　　(2)　　(3)

12. 对数组 a[n]按升序进行的选择排序算法

```
void SelectSort(int a[], ___(1)___)
{
    int i,j,k;
    for(i=1;i<n;i++) {  //进行 n-1 次选择和交换
        k=i-1;
        for(j=i;j<n;j++)
            if(a[j]<a[k]) ___(2)___;
        int x=a[i-1]; a[i-1]=a[k]; ___(3)___;
    }
}
```

(1)　　(2)　　(3)

13. 对数组 a[n]按升序进行的插入排序算法

```
void InsertSort(___(1)___, int n)
{
    int i,j,x;
    for(i=1;i<n;i++) {  //进行 n-1 次循环
        x=a[i];
        for(j=i-1;j>=0;j--)  //为 x 顺序向前寻找合适的插入位置
            if(x<a[j]) ___(2)___;
            else ___(3)___;
        a[j+1]=x;
    }
}
```

(1)　　(2)　　(3)

14. 对按从小到大排列的有序数组 a[n]进行二分查找 x 的算法，若查找成功返回该元素下标，否则返回-1。

```
int BinarySearch(int a[],int x)
{
    int low=0, high=N-1;  //定义并初始化区间下界和上界变量
    int mid;  //定义保存中点元素下标的变量
    while(low<=high) {
        mid=___(1)___;
        if(x==a[mid]) ___(2)___;
        else if(x<a[mid]) high=mid-1;
        else ___(3)___;
    }
    return -1;
}
```

(1)　　(2)　　(3)

15．用插入排序方法对 table 指针数组中 size 个指针所指向的字符串进行按升序排序的算法。

```
void sort(char *table[], int size){
for(int i=1,___(1)___; i++){
    char *p=table[i];
    for(int j=i-1; j>=0 ; j--)
        if(strcmp(p,table[j])<0) ___(2)___;
        else break;
    table[j+1]=___(3)___;
}
}
```

(1)　　(2)　　(3)

16．假定有定义为“struct NODE{int data; NODE* next;};”，下面算法根据 table 数组中的 n 个元素建立一个表头指针为 L 的链表，链表中结点值的顺序与数组元素值的顺序相同。

```
void f5(NODE*& L, int table[], int n)
{
if(n<=0) {L=NULL; return;}
L=new NODE;  //生成附加的头结点
int i=0;
NODE* p=L;
while(___(1)___) {
    p=p->next=___(2)___;
    p->data=___(3)___;
    i++;
}
p->next=NULL;  //把最后一个结点的指针域置空
p=L;
L=L->next;  //使 L 指向链表的第一个带值的结点
delete p;
}
```

(1)　　(2)　　(3)

17．假定有定义为“struct NODE{int data; NODE* next;};”，下面算法根据 table 数组中的 n 个元素建立一个表头指针为 L 的链表，链表中结点值的顺序与数组元素值的顺序正好相反。

```
void f6(NODE*& L, int table[], int n)
{
L=NULL;
if(n<=0) return;
int i=0;
NODE* p;
while(___(1)___) {
```

```
    p=new NODE;
    p->data=___(2)___;
    p->next=L;
    ___(3)___;
    i++;
}
}
```

(1)　　(2)　　(3)

18．假定有定义为“struct NODE{int data; NODE* next;};”，下面算法是依次显示输出以 L 为表头指针的链表中各结点的值。

```
void f7(NODE* L)
{
for(___(1)___; p!=NULL; ___(2)___)
    cout<<___(3)___ <<' ';
cout<<endl;
}
```

(1)　　(2)　　(3)

19．假定有定义为“struct NODE{int data; NODE* next;};”，下面算法是把以 L 为表头指针的链表中各结点依次按相反次序链接并返回新链表的表头指针。

```
NODE* f8(NODE* L)
{
if(L==NULL) return NULL;
NODE *p=NULL, *q=L, *t;
while(q!=NULL) {
    t=q;
    q=___(1)___;
    t->next=___(2)___;
    p=t;
}
___(3)___;
}
```

(1)　　(2)　　(3)

20．已知一维数组类 ARRAY 的定义如下，ARRAY 与普通一维数组区别是：其重载的运算符[ ]要对下标是否越界进行检查。

```
class ARRAY{
    int *v;         //指向存放数组数据的空间
    int s;          //数组大小
public:
    ARRAY(int a[], int n);
```

```
    ~ ARRAY(){delete []v;}
    int size(){ return s;}
    int& operator[](int n);
};
___(1)___ operator[](int n)   //[ ]的运算符成员函数定义
{
    if(n<0 || ___(2)___) {cerr<<"下标越界！"; exit(1);}
    return ___(3)___;
}
```

(1)　　(2)　　(3)

21．已知一维数组类 ARRAY 的定义如下，构造函数的作用是把参数 n 的值赋给 s，给 v 动态分配长度为 n 的数组空间，接着利用数组参数 a 初始化 v 所指向的数组。

```
class ARRAY{
    int *v;         //指向存放数组数据的空间
    int s;          //数组大小
public:
    ARRAY(int a[], int n);
    ~ ARRAY(){delete []v;}
    int size(){ return s;}
    int& operator[](int n);
};
___(1)___ ARRAY(int a[], int n)
{
    if(n<=0) {v=NULL;s=0;return;}
    s=n;
    v=___(2)___;
    for(int i=0; i<n; i++) ___(3)___;
}
```

(1)　　(2)　　(3)

22．下面是一维数组类 ARRAY 的定义，ARRAY 与普通一维数组区别是：(a)用()而不是[]进行下标访问，(2)下标从 1 而不是从 0 开始，(c)要对下标是否越界进行检查。

```
class ARRAY{
    int *v;        //指向存放数组数据的空间
    int s;          //数组大小
public:
    ARRAY(int a[], int n);
    ~ ARRAY(){delete []v;}
    int size(){ return s;}
    int& operator()(int n);
}; ___(1)___ operator()(int n)
{   // ()的运算符函数定义
```

```
    if(___(2)___) {cerr<<"下标越界！"; exit(1);}
    return ___(3)___;
}
```

(1)　　(2)　　(3)

23. 已知一个类的定义如下：

```
#include<iostream.h>
class AA {
    int a[10];
    int n;
public:
    void SetA(int aa[], int nn); //用数组 aa 初始化数据成员 a，
                                 //用 nn 初始化数据成员 n
    int MaxA();  //从数组 a 中前 n 个元素中查找最大值
    void SortA();  //采用选择排序的方法对数组 a 中前 n 个元素
                   //进行从小到大排序
    void InsertA();//采用插入排序的方法对数组 a 中前 n 个元素进行从小到大排序
    void PrintA();  //依次输出数组 a 中的前 n 个元素
};
```

该类中 MaxA()函数的实现如下，请在标号位置补充适当的内容。

```
int _____(1)_____
{
    int x=a[0];
    for(int i=1; i<n; i++)
        if(a[i]>x) ___(2)___;
    ___(3)___;
}
```

(1)　　(2)　　(3)

24. 已知一个类的定义如下：

```
#include<iostream.h>
class AA {
    int a[10];
    int n;
public:
    void SetA(int aa[], int nn); //用数组 aa 初始化数据成员 a，
                                 //用 nn 初始化数据成员 n
    int MaxA();  //从数组 a 中前 n 个元素中查找最大值
    void SortA();  //采用选择排序的方法对数组 a 中前 n 个元素
                   //进行从小到大排序
    void InsertA();//采用插入排序的方法对数组 a 中前 n 个元素进行从小到大排序
    void PrintA();  //依次输出数组 a 中的前 n 个元素
```

```
};

void AA::SortA()
{
    int i,j;
    for(i=0; ___(1)___; i++) {
        int x=a[i], k=i;
        for(j=i+1; j<n; j++)
            if(a[j]<x) ___(2)___
        a[k]=a[i];
        ___(3)___;
    }
}
```

(1)　　(2)　　(3)

25. 已知一个类的定义如下：

```
#include<iostream.h>
class AA {
    int a[10];
    int n;
public:
    void SetA(int aa[], int nn); //用数组 aa 初始化数据成员 a，
                                 //用 nn 初始化数据成员 n
    int MaxA();  //从数组 a 中前 n 个元素中查找最大值
    void SortA();  //采用选择排序的方法对数组 a 中前 n 个元素
                   //进行从小到大排序
    void InsertA();//采用插入排序的方法对数组 a 中前 n 个元素进行从小到大排序
    void PrintA();  //依次输出数组 a 中的前 n 个元素
};

void ___(1)___
{
    int i,j;
    for(i=1; i<n; i++) {
        int x=a[i];
        for(j=i-1; j>=0; j--)
            if(x<a[j]) ___(2)___;
            else ___(3)___;
        a[j+1]=x;
    }
}
```

(1)　　(2)　　(3)

26. 已知一个类的定义如下：

```
#include<iostream.h>
class AA {
    int a[10];
    int n;
public:
    void SetA(int aa[], int nn); //用数组 aa 初始化数据成员 a，
                                 //用 nn 初始化数据成员 n
    int MaxA();  //从数组 a 中前 n 个元素中查找最大值
    void SortA();  //采用选择排序的方法对数组 a 中前 n 个元素
                   //进行从小到大排序
    void InsertA();//采用插入排序的方法对数组 a 中前 n 个元素进行从小到大排序
    void PrintA();  //依次输出数组 a 中的前 n 个元素
                    //最后输出一个换行
};
```

使用该类的主函数如下：

```
void main()
{
    int a[10]={23,78,46,55,62,76,90,25,38,42};
    AA x;
    ___(1)___;
    int m=___(2)___;
    ___(3)___;
    cout<<m<<endl;
}
```

该程序运行结果为:

```
23 78 46 55 62 76
78
```

(1)　　(2)　　(3)

27. 已知一个类的定义如下：

```
#include<iostream.h>
class AA {
    int a[10];
    int n;
public:
    void SetA(int aa[], int nn); //用数组 aa 初始化数据成员 a，
                                 //用 nn 初始化数据成员 n
    int MaxA();  //从数组 a 中前 n 个元素中查找最大值
    void SortA();  //采用选择排序的方法对数组 a 中前 n 个元素
                   //进行从小到大排序
    void PrintA();  //依次输出数组 a 中的前 n 个元素，
                    //最后输出一个换行
```

```
};
```

使用该类的主函数如下：

```
void main()
{
    int a[10]={23,78,46,55,62,76,90,25,38,42};
    ___(1)___;
    x.SetA(a,8);
    int ___(2)___;
    ___(3)___;
    x.PrintA();
    cout<<m<<endl;
}
```

该程序运行结果为:

23 25 46 55 62 76 78 90

90

(1)　　(2)　　(3)

28. 已知一个利用数组实现栈的类定义如下：

```
const int ARRAY_SIZE=10;
class Stack {
public:
    void Init() {top=-1;}          //初始化栈为空
    void Push(int newElem);       //向栈中压入一个元素
    int Pop();                    //从栈顶弹出一个元素
    bool Empty() {  //判栈空
        if(top==-1) return true;else return false;}
    int Depth() {return top+1;}  //返回栈的深度
    void Print();
        //按照后进先出原则依次输出栈中每个元素，直到栈空为止
private:
    int elem[ARRAY_SIZE];  //用于保存栈元素的数组
    int top;               //指明栈顶元素位置的指针
};

void Stack::Push(int newElem) {
    if(___(1)___) {
        cout<<"栈满!"<<endl;
        exit(1);  //中止运行
    }
    ___(2)___;
    elem[top]=___(3)___;
}
```

(1)　　(2)　　(3)

29. 已知一个利用数组实现栈的类定义如下：

```
const int ARRAY_SIZE=10;
class Stack {
public:
    void Init() {top=-1;}       //初始化栈为空
    void Push(int newElem);    //向栈中压入一个元素
    int Pop();                 //从栈顶弹出一个元素
    bool Empty() {  //判栈空
        if(top==-1) return true;else return false;}
    int Depth() {return top+1;}  //返回栈的深度
    void Print();
        //按照后进先出原则依次输出栈中每个元素，直到栈空为止
private:
    int elem[ARRAY_SIZE];  //用于保存堆栈元素的数组
    int top;               //指明栈顶元素位置的指针
};
```

该类的 Pop 和 Print 函数的实现分别如下：

```
___(1)___ {
    if(top==-1) {
        cout<<"栈空!"<<endl;
        exit(1);  //中止运行
    }
    return ___(2)___;
}

void Stack::Print() {
    while(!Empty())
        cout<<___(3)___ <<' ';
}
```

(1)　　(2)　　(3)

30. 

```
class A {
  int a;
public:
  A() {a=0;}
  ___(1)___{}  //定义构造函数，用参数 aa 初始化数据成员 a
};
main() {
  ___(2)___;  //定义类 A 的指针对象 p
  ___(3)__;   //用 p 指向动态对象并初始化为整数 5
```

```
}
```

(1)　　(2)　　(3)

```
31. class A {
  char *a;
public:
  ___(1)___  //定义无参构造函数，使 a 的值为空
  A(char *aa) {
    a=___(2)___;
    strcpy(a,aa);  //用 aa 所指字符串初始化 a 所指向的动态存储空间
  }
  ___(3)___  //定义析构函数，删除 a 所指向的动态存储空间
};
```

(1)　　(2)　　(3)

```
32. class A {
   int a,b;
public:
   A(int aa=0, int bb=0) ___(1)___ {} //分别用 aa 和 bb 对应初始化 a 和 b
};
main() {
   ___(2)___ ;  //定义类 A 的对象 x 并用 5 初始化，同时定义 y 并用 x 初始化
   ___(3)___ ;  //定义 p 指针，使之指向对象 x
}
```

(1)　　(2)　　(3)

```
33. class A {
   int a,b;
public:
   ___(1)___ //定义构造函数，使参数 aa 和 bb 的默认值为 0，
             //在函数体中用 aa 初始化 a，用 bb 初始化 b
};
main() {
   A *p1, *p2;
   ___(2)___ ;  //调用无参构造函数生成由 p1 指向的动态对象
   ___(3)___ ;  //调用带参构造函数生成由 p2 指向的动态对象，
                //使 a 和 b 成员分别被初始化为 4 和 5
}
```

(1)　　(2)　　(3)

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

```
#include<stdlib.h>
class A {
  int *a; int n; int MaxLen;
public:
  A(): a(0), n(0), MaxLen(0) {}
  A(int *aa, int nn, int MM) {
    n=nn;
    MaxLen=MM;
    if(n>MaxLen) exit(1);
    ___(1)___;  //由 a 指向长度为 MaxLen 的动态数组
    for(int i=0; i<n; i++) a[i]=aa[i];
  }
  ~A() {delete []a;}
  int GetValue(int i) ___(2)___ //函数体返回 a[i]的值
};
void main()
{
    int b[10]={1,2,3,4,5,6,7,8,9,10};
    A r(b,10,10);
    int i,s=0;
    for(i=0; i<10; i++); ___(3)___ //把 r 对象的 a 数据成员中的每个
                                   //元素值依次累加到 s 中
    cout<<"s="<<s<<endl;
}
```

(1)　　(2)　　(3)

35.

```
#include<iostream.h>
#include<stdlib.h>
class A {
  int *a; int n; int MaxLen;
public:
  A(): a(0), n(0), MaxLen(0) {}
  A(int *aa, int nn, int MM) {
    n=nn;
    MaxLen=MM;
    if(n>MaxLen) exit(1);
    a=new int[MaxLen];
    ___(1)___;  //以 i 为循环变量把 aa 数组中每个元素值
                //传送给 a 数组的对应元素中
  }
  ~A();
  int GetValue(int i) {return a[i];} //函数体返回 a[i]的值
};
```

___(2)___  //析构函数的类外定义

```
void main()
{
    int b[10]={1,2,3,4,5,6,7,8,9,10};
    A r(b,10,10);
    int i,s=0;
    ___(3)___ ; //以 i 为循环变量，把 r 对象的 a 数据成员中的
                //每个元素值依次累加到 s 中
    cout<<"s="<<s<<endl;
}
```

(1)　　(2)　　(3)

36. 一种类定义如下:

```
class Goods
{
  private:
    char gd_name[20];          //商品名称
    int weight;                //商品重量
    static int totalweight;    //同类商品总重量
  public:
    Goods (char*str,int w){    //构造函数
    strcpy(gd_name,str);
    weight=w;
    totalweight+=weight;
}
~ Goods (){totalweight -= weight;}
char* GetN(){___(1)___;}  //返回商品名称
int GetW(){return weight;}
___(2)___  GetTotal_Weight() {  //定义静态成员函数返回总重量
    ___(3)___;
}
}
```

(1)　　(2)　　(3)

37.

```
class Point
{
  private:
    int x, y;
  public:
    Point(){x=y=0;}
    Point(int x0,int y0) {x=x0;y=y0;}
```

```
    int GetX() { return x; }
    int GetY() { return y; }
    void Print(){cout<<"Point("<<x<<","<<y<<")"<<endl;}
    ___(1)___;  //友元函数声明
    ___(2)___;  //友元函数声明
};

Point operator+(Point& pt,int dd)
    //加号操作符重载函数，实现 Point 类对象与整数加法
{
  Point temp=pt;
  temp.x+=dd;
  temp.y+=dd;
  return temp;
}

Point operator+(Point& pt1,Point& pt2)
//加号操作符重载函数，实现两个 Point 类对象的加法
{
  Point temp=pt1;
  temp.x+=pt2.x;
  ___(3)___;
  return temp;
}
```

(1)　　(2)　　(3)

38．在下面一段类定义中， Derived 类是由直接基类 Base 1 和 Base 2 所派生的，Derived 类包含有两个间接基类 BaseBase，在初始化函数 Init 中，需要把 x1 和 x2 的值分别赋给属于基类 Base1 的 x 成员和属于基类 Base2 的 x 成员。

```
class BaseBase {
  protected:
    int x;
  public:
    BaseBase(){ x = 1;}
};

class Base1: public BaseBase {
  public:
    Base1(){}
};

class Base2: public BaseBase {
  public:
    Base2(){}
};
```

```
class Derived: ___(1)___
{
  public:
       Derived() {}
       void Init(int x1, int x2) {
            ___(2)___; ___(3)___;
       }

       void output() {cout<<Base1::x<<' '<<Base2::x<<endl;}
};
```

(1)　　(2)　　(3)

39. 在下面一段类定义中， Derived 类公有继承了基类 Base。需要填充的函数由注释内容给出了功能。

```
class Base
{
  private:
    int mem1,mem2;     //基类的数据成员
  public:
       Base(int m1,int m2) {
         mem1=m1; mem2=m2;
       }
       void output(){cout<<mem1<<' '<<mem2<<' ';}
      //...
};

class Derived: public Base
{
  private:
    int mem3;          //派生类本身的数据成员
  public:
      //构造函数,由 m1 和 m2 分别初始化 mem1 和 mem2，由 m3 初始化 mem3
    Derived(int m1,int m2, int m3);
  //输出 mem1,mem2 和 mem3 数据成员的值
    void output(){
        ___(1)___; cout<<mem3<<endl;
    }
//...
};

Derived::Derived(int m1,int m2, int m3): ___(2)___ {___(3)___;}
```

(1)　　(2)　　(3)

40．在下面一段类的定义中，需要填充的函数由注释内容给出了功能。

```
class Point //定义坐标点类
{
  public:
    int x,y;    //点的横坐标和纵坐标
    Point(){x=0;y=0;}
    Point(int x0,int y0) {x=x0; y=y0;}
    int X(){return x;}
    int Y(){return y;}
    void PrintP(){cout<<"Point:("<<x<<","<<y<<")"<<endl;}
};

class Line: public Point    //利用坐标点类定义直线类
{
  private:
    class Point pt1,pt2;    //直线的两个端点
  public:
    Line(Point pts, Point pte);  //构造函数，分别用参数初始化对应的端点
    double Dx(){return pt2.x-pt1.x;}
    double Dy(){return pt2.y-pt1.y;}
    double Length(){ //计算直线的长度
        return sqrt(___(1)___);
    };
    void PrintL();  //输出直线的两个端点和直线长度
};

Line::Line(Point pts, Point pte) ___(2)___
void Line::PrintL()
{
  cout<<"1st ";
  pt1.PrintP();
  cout<<"2nd ";
  pt2.PrintP();
  cout<<"The Length of Line: "<<___(3)___ <<endl;
}
```

(1)　　(2)　　(3)

41．在下面一段类的定义中，自行车类的虚基类为车辆类，机动车类的虚基类也为车辆类，摩托车类的基类为自行车类和机动车类，类之间均为公有继承。

```
class vehicle  //车辆类
{
  private:
    int MaxSpeed;  //最大车速
    int Weight;    //车重
```

```
  public:
    vehicle(){MaxSpeed=0; Weight=0;};
    virtual void Run() {cout<<"A vehicle is running!"<<endl;}
};

class bicycle : ___(1)___  //自行车类
{
  private:
    int Height;  //车高
  public:
    bicycle(){};
    void Run() {cout<<"A bicycle is running!"<<endl;}
};

class motorcar : ___(2)___  //机动车类
{
  private:
    int SeatNum;  //乘人数
  public:
    motorcar(){};
    void Run() {cout << "A motorcar is running!" << endl; }
};

class motorcycle: ___(3)___  //摩托车类
{
  public:
    motorcycle (){};
    void Run() {cout<<"A motorcycle is running!"<<endl;}
};
```

(1)　　(2)　　(3)

# 四.写出程序运行结果

1.
```
#include<iostream.h>
#include<stdlib.h>
void main()
{
    int a[8]={25,48,32,85,64,18,48,29};
    int max,min;
    max=min=a[0];
    for(int i=0; i<8; i++) {
        if(x>a[i]) max=a[i];
         if(x<a[i]) min=a[i];
    }
```

```
        cout<<"max:"<<max<<endl;
        cout<<"min:"<<min<<endl;
    }
```

2.
```
    #include<iostream.h>
    void main()
    {
        int a,b;
        for(a=1,b=2; b<50;) {
            cout<<a<<' '<<b<<' ';
            a=a+b;
            b=a+b;
        }
        cout<<endl;
        cout<<a<<' '<<b<<' '<<endl;
    }
```

3.
```
    #include<iostream.h>
    const int M=3, N=4;
    void main()
    {
        int i,j,s=0;
        for(i=1;i<=M;i++)
            for(j=1;j<=N;j++)
                s+=i*j;
        cout<<” s=” <<s<<endl;
    }
```

4.
```
#include<iostream.h>
void main()
{
    int a=2,b=5,c=0;
    if(a+b>10) c=a*b; else c=3*a+b;
    if(c<=20) cout<<c*c; else cout<<4+c-5;
    cout<<endl;
    a=a+b; b=a+b;c+=a+b;
    cout<<"a,b,c="<<a<<','<<b<<','<<c<<endl;
}
```

5.
```
#include<iostream.h>
void main()
{
    int x=5;
    switch(2*x-3) {
      case 4: cout<<x<<’  ’ ;
```

```
        case 7: cout<<2*x+1<<' ';
        case 10: cout<<3*x-1<<' '; break;
        default: cout<<"default"<<endl;
    }
    cout<<"switch end."<<endl;
}
```

6.
```
#include<iomanip.h>
#include<math.h>
int a[4]={36,-5,73,8};
void main()
{
    int i,y;
    for(i=0; i<4; i++) {
        if(a[i]<0) y=1;
        else if(a[i]<10) y= a[i]* a[i]+3;
        else if(a[i]<60) y=4*a[i]-5;
        else y=int(sqrt(a[i]));  // sqrt(x)为取 x 的平方根函数
        cout<<setw(5)<<a[i]<<setw(5)<<y;
    }
}
```

7.
```
#include<iostream.h>
int a[8]={36,25,20,43,12,70,66,35};
void main()
{
    int s0,s1,s2;
    s0=s1=s2=0;
    for(int i=0; i<8; i++) {
        switch(a[i]%3) {
          case 0: s0+=a[i];break;
          case 1: s1+=a[i];break;
          case 2: s2+=a[i];break;
        }
    }
    cout<<s0<<' '<<s1<<' '<<s2<<endl;
}
```

8.
```
#include<iomanip.h>
const int N=5;
void main()
{
    int i,p=1,s=0;
    for(i=1;i<N; i++) {
        p=p*i;
```

```
            s=s+p;
            cout<<setw(5)<<i<<setw(5)<<p;
            cout<<setw(5)<<s<<endl;
        }
    }
```

9. 
```
#include<iomanip.h>
const int M=20;
void main()
{
    int c2,c3,c5;
    c2=c3=c5=0;
    for(int i=1; i<=M; i++) {
        if(i%2==0) c2++;
        if(i%3==0) c3++;
        if(i%5==0) c5++;
    }
    cout<<c2<<' '<<c3<<' '<<c5<<endl;
}
```

10. 
```
#include<iomanip.h>
void main()
{
    int i,j;
    for(i=0;i<5;i++) {
        for(j=i;j<5;j++) cout<<' *' ;
        cout<<endl;
    }
}
```

11. 
```
#include<iostream.h>
void main()
{
    for(int i=1,s=0;i<20;i++) {
        if(i%2==0 || i%3==0) continue;
        cout<<i<<'  ' ;
        s+=i;
    }
    cout<<s<<endl;
}
```

12. 
```
#include<iostream.h>
const int T=6;
void main()
{
```

```
    int i,j;
    for(i=1;i<=T;i+=2)
        for(j=2;j<=T;j+=2) {
            if(i+j<T) cout<<' +' ;
            else cout<<' *' ;
        }
}
```

13.
```
#include<iostream.h>
void main()
{
    int a,b,c=0;
    for(a=1;a<4;a++)
        for(b=6;b>1;b-=2) {
            if((a+b)%2==0) c+=a+b; else c+=a*b;
            if(b==2) cout<<a<<'  ' <<b<<'  ' <<c<<endl;
        }
}
```

14.
```
#include<iostream.h>
const int B=2;
void main()
{
    int p=1,s=1;
    while(s<50) {
        p*=B;
        s+=p;
    }
    cout<<"s="<<s<<endl;
}
```

15.
```
#include<iostream.h>
void main()
{
    int x=24,y=88;
    int i=2,p=1;
    do {
        while(x%i==0 && y%i==0) {
            p*=i;
            x/=i;
            y/=i;
        }
        i++;
    }while(x>=i && y>=i);
    cout<<p*x*y<<endl;
```

```
}
```

16.
```
#include<iomanip.h>
const int N=3;
void main()
{
    int a[N][N]={{7,-5,3},{2,8,-6},{1,-4,-2}};
    int b[N][N]={{3,6,-9},{2,-8,3},{5,-2,-7}};
    int i,j,c[N][N];
    for(i=0;i<N;i++)   //计算矩阵 C
        for(j=0;j<N;j++)
            c[i][j]=a[i][j]+b[i][j];
    for(i=0;i<N;i++) {   //输出矩阵 C
        for(j=0;j<N;j++)
            cout<<setw(5)<<c[i][j];
        cout<<endl;
    }
}
```

17.
```
#include<iostream.h>
int a=5;
void main() {
    int b=a+20;
    int a=10;
    cout<<a<<' '<<b<<endl;
    {    int a=0,b=0;
        for(int i=1; i<6; i++) {
             a+=i; b+=a;
        }
        cout<<a<<' '<<b<<endl;
    }
    cout<<a<<' '<<b<<endl;
}
```

18.
```
#include<iostream.h>
int f1(int x, int y)
{
    x=x+y; y=x+y;
    cout<<"x="<<x<<", y="<<y<<endl;
    return x+y;
}
void main()
{
    int x=5,y=8;
    int z=f1(x,y);
```

```
    cout<<"x="<<x<<", y="<<y;
    cout<<", z="<<z<<endl;
}
```

19.
```
#include<iostream.h>
void f2(int& x, int& y)
{
    int z=x; x=y; y=z;
}
void f3(int* x, int* y)
{
     int z=*x; *x=*y; *y=z;
}
void main()
{
    int x=10,y=26;
    cout<<"x,y="<<x<<", "<<y<<endl;
    f2(x,y);
    cout<<"x,y="<<x<<", "<<y<<endl;
    f3(&x,&y);
    cout<<"x,y="<<x<<", "<<y<<endl;
    x++; y--;
    f2(y,x);
    cout<<"x,y="<<x<<", "<<y<<endl;
}
```

20.
```
#include<iostream.h>
void f4(int a[], int n, int& s)
{
    s=0;
    for(int i=0; i<n; i++) s+=a[i];
}
void main()
{
    int b[8]={4,8,6,9,2,10,7,12};
    int x; f4(b,5,x);
    cout<<x<<' ';
    int y; f4(b+3,4,y);
    cout<<y<<' ';
    cout<<x+y<<endl;
}
```

21.
```
#include<iostream.h>
void main() {
    int a[8]={36,25,48,14,55,40,50,24};
```

```
    int b1, b2;
    b1=b2=a[0];
    for(int i=1;i<8;i++)
        if(a[i]>b1) {b2=b1; b1=a[i];}
        else if(a[i]>b2) b2=a[i];
    cout<<b1<<' '<<b2<<endl;
}
```

22.
```
#include<iostream.h>
void main() {
    char a[]="abcdabcabfgacd";
    int i1=0, i2=0, i=0;
     while(a[i]) {
        if(a[i]=='a') i1++;
        if(a[i]=='b') i2++;
        i++;
    }
    cout<<i1<<' '<<i2<<' '<<i<<endl;
}
```

23.
```
#include<iostream.h>
void main() {
    int a[10]={76,83,54,62,40,75,90,92,77,84};
    int b[4]={60,70,90,101};
    int c[4]={0};
    for(int i=0;i<10;i++) {
        int j=0;
        while(a[i]>=b[j]) j++;
        c[j]++;
    }
    for(i=0;i<4;i++) cout<<c[i]<<' ';
    cout<<endl;
}
```

24.
```
#include<iostream.h>
#include<string.h>
void main() {
    char a[5][10]={"student","worker","soldier","cadre","peasant"};
    char s1[10], s2[10];
    strcpy(s1,a[0]); strcpy(s2,a[0]);
    for(int i=1;i<5;i++) {
        if(strcmp(a[i], s1)>0) strcpy(s1,a[i]);
        if(strcmp(a[i], s2)<0) strcpy(s2,a[i]);
    }
    cout<<s1<<' '<<s2<<endl;
```

```
}
```

25.
```
#include<iostream.h>
const int N=5;
void fun();
void main()
{
    for(int i=1; i<N; i++)
    fun();
}
void fun()
{
    static int a;
    int b=2;
    cout<<(a+=3,a+b)<<' ';
}
```

26.
```
#include<iostream.h>
void main()
{
      char s[3][5]={"1234","abcd","+-*/"};
    char *p[3];
    for(int I=0;I<3;I++) p[I]=s[I];
    for(I=2;I>=0;I--) cout<<p[I]<<' ';
    cout<<endl;
}
```

27.
```
#include<iostream.h>
void main()
{
    int i,j,len[3];
    char a[3][8]={"year","month","day"};
    for(i=0;i<3;i++) {
        for(j=0;j<8;j++)
            if(a[i][j]=='\0') {
                len[i]=j;break;
        }
        cout<<a[i]<<":"<<len[i]<<endl;
    }
}
```

28.
```
#include<iostream.h>
#include<string.h>
class CD {
    char* a;
```

```
    int b;
  public:
    void Init(char* aa, int bb)
    {
        a=new char[strlen(aa)+1];
        strcpy(a,aa);
        b=bb;
    }
    char* Geta() {return a;}
    int Getb() {return b;}
    void Output() {cout<<a<<' '<<b<<endl;}
} dx;
void main()
{
CD dy;
dx.Init("abcdef",30);
dy.Init("shenyafen",3*dx.Getb()+5);
dx.Output();
dy.Output();

}
```

29. .
```
#include<iostream.h>
    #include<string.h>
    class CD {
        char* a;
        int b;
      public:
        void Init(char* aa, int bb)
        {
            a=new char[strlen(aa)+1];
            strcpy(a,aa);
            b=bb;
        }
        char* Geta() {return a;}
        int Getb() {return b;}
        void Output() {cout<<a<<' '<<b<<endl;}
    };
void main()
{
CD dx,dy;
char a[20];
dx.Init("abcdef",30);
strcpy(a,dx.Geta());
strcat(a,"xyz");
```

```
dy.Init(a,dx.Getb()+20);
dx.Output();
dy.Output();

}
```

30.
```
#include<iostream.h>
    class CE {
      private:
        int a,b;
        int getmax() {return (a>b? a:b);}
      public:
        int c;
        void SetValue(int x1,int x2, int x3) {
            a=x1; b=x2; c=x3;
        }
        int GetMax();
    };
int CE::GetMax() {
    int d=getmax();
    return (d>c? d:c);
}
void main()
{
int x=5,y=12,z=8;
CE ex, *ep=&ex;
ex.SetValue(x,y,z);
cout<<ex.GetMax()<<endl;
ep->SetValue(x+y,y-z,20);
cout<<ep->GetMax()<<endl;
}
```

31.
```
#include<iostream.h>
    class CE {
      private:
        int a,b;
        int getmin() {return (a<b? a:b);}
      public:
        int c;
        void SetValue(int x1,int x2, int x3) {
            a=x1; b=x2; c=x3;
        }
        int GetMin();
        };
int CE::GetMin() {
```

```
    int d=getmin();
    return (d<c? d:c);
}
void main()
{
int x=5,y=12,z=8;
CE *ep;
ep=new CE;
ep->SetValue(x+y,y-z,10);
cout<<ep->GetMin()<<endl;
CE a=*ep;
cout<<a.GetMin()*3+15<<endl;
}
```

32.
```
#include<iostream.h>
    class Franction {   //定义分数类
        int nume;   //定义分子
        int deno;   //定义分母
      public:
      //把*this 化简为最简分数，具体定义在另外文件中实现
        void FranSimp();
      //返回两个分数*this 和 x 之和，具体定义在另外文件中实现
        Franction FranAdd(const Franction& x);
      //置分数的分子和分母分别 0 和 1
        void InitFranction() {nume=0; deno=1;}
      //置分数的分子和分母分别 n 和 d
        void InitFranction(int n, int d) {nume=n; deno=d;}
      //输出一个分数
        void FranOutput() {cout<<nume<<'/'<<deno<<endl;}
    };

    void main()
    {
        Franction a,b,c,d;
        a.InitFranction(7,12);
        b.InitFranction(-3,8);
        c.InitFranction();
        c=a.FranAdd(b);
        d=c.FranAdd(a);
        cout<<"a: "; a.FranOutput();
        cout<<"b: "; b.FranOutput();
        cout<<"c: "; c.FranOutput();
        cout<<"d: "; d.FranOutput();
    }
```

33. #include<iostream.h>

```
class Franction {  //定义分数类
    int nume;  //定义分子
    int deno;  //定义分母
  public:
  //把*this 化简为最简分数，具体定义在另外文件中实现
    void FranSimp();
  //返回两个分数*this 和 x 之和，具体定义在另外文件中实现
    Franction FranAdd(const Franction& x);
  //置分数的分子和分母分别 0 和 1
    void InitFranction() {nume=0; deno=1;}
  //置分数的分子和分母分别 n 和 d
    void InitFranction(int n, int d) {nume=n; deno=d;}
  //输出一个分数
    void FranOutput() {cout<<nume<<'/'<<deno<<endl;}
};

void main()
{
    Franction a,b,c,d;
    a.InitFranction(6,15);
    b.InitFranction(3,10);
    c.InitFranction();
    c=a.FranAdd(b);
    d=c.FranAdd(a);
    cout<<"a: "; a.FranOutput();
    cout<<"b: "; b.FranOutput();
    cout<<"c: "; c.FranOutput();
    cout<<"d: "; d.FranOutput();
}
```

34.

```
#include<iostream.h>
#include<string.h>
class A {
   char *a;
public:
   A(char *s) {
   a=new char[strlen(s)+1];
   strcpy(a,s);
   cout<<a<<endl;
   }
   ~A() {
        delete []a;
      cout<<"Destructor!"<<endl;
```

```
    }
};
void main() {
A x("xuxiaokai");
A *y=new A("weirong");
delete y;
}
```

35.

```
#include<iostream.h>
class A {
    int *a;
public:
    A(int x=0):a(new int(x)){}
    ~A() {delete a;}
    int getA() {return *a;}
    void setA(int x) {*a=x;}
};
void main() {
A x1,x2(3);
A *p=&x2;
p->setA(x2.getA()+5);
x1.setA(15+x1.getA());
cout<<x1.getA()<<' '<<x2.getA()<<endl;
}
```

36.

```
#include<iostream.h>
class A {
    int a;
public:
    A(int aa=0): a(aa) {cout<<a<<' ';}
    ~A() {cout<<"Xxk";}
};
void main() {
A *p;
A x[3]={1,2,3},y=4;
cout<<endl;
p=new A[3];
cout<<endl;
delete []p;
cout<<endl;
}
```

37.

```
#include<iostream.h>
class A {
int a,b;
public:
A() {a=b=0;}
A(int aa, int bb) {a=aa; b=bb;}
int Sum() {return a+b;}
int* Mult() {
    int *p=new int(a*b);
    return p;
}
};
void main() {
int *k;
A x(2,3), *p;
p=new A(4,5);
cout<<x.Sum()<<' '<<*(x.Mult())<<endl;
cout<<p->Sum()<<' '<<*(k=p->Mult())<<endl;
delete k;
}
```

38.

```
#include<iostream.h>
class A {
int a[10]; int n;
  public:
    A(int aa[], int nn): n(nn) {
       for(int i=0; i<n; i++) a[i]=aa[i];
    }
    int Get(int i) {return a[i];}
    int SumA(int n) {
    int s=0;
    for(int j=0; j<n; j++) s+=a[j];
    return s;
}
};
void main() {
int a[]={2,5,8,10,15,20};
A x(a,4);
A y(a,6);
int d=1;
for(int i=0; i<4; i++) d*=x.Get(i);
int f=y.SumA(5);
cout<<"d="<<d<<endl;
cout<<"f="<<f<<endl;
```

```
}
```

39.

```
#include<iostream.h>
class A {
    int a,b;
public:
A(int aa, int bb) {a=aa; b=bb;}
float Multip(char op) {
  switch(op) {
    case '+': return a+b;
    case '-': return a-b;
    case '*': return a*b;
    case '/': if(b!=0)return float(a)/b;
        else {cout<<"除数为 0!"<<endl; return 0;}
    default: cout<<'\n'<<op<<"非法运算符!"<<endl;
            return 0;
    }
  }
};
void main() {
    A x(10,4);
    char a[6]="+-*/@";
    int i=0;
    while(a[i]) {
        float k=x.Multip(a[i]);
        if(k!=0) cout<<k<<' ';
        i++;
    }
    cout<<endl;
}
```

40.

```
#include <iostream.h>
class Point {
    int x,y;
public:
    Point(int x1=0, int y1=0) :x(x1), y(y1) {
        cout<<"Point:"<<x<<' '<<y<<'\n';
    }
    ~Point() {
        cout<<"Point destructor!\n";
    }
};
class Circle {
```

```
    Point center;      //圆心位置
    int radius;          //半径
public:
    Circle(int cx,int cy, int r):center(cx,cy),radius(r) {
        cout<<"Circle radius:"<<radius<<'\n';
    }
    ~Circle() {cout<<"Circle destructor!\n";}
};

void main()
{
    Circle c(3,4,5);
}
```

41.

```
#include <iostream.h>
#include <string.h>
class Point {
    int x,y;
public:
    Point(int x1=0, int y1=0) :x(x1), y(y1) {
        cout<<"Point:"<<x<<' '<<y<<'\n';
    }
    ~Point() {
        cout<<"Point des!\n";
    }
};
class Text {
    char text[100];     //文字内容
public:
    Text(char * str) {
        strcpy(text,str);
        cout<<"Text con!\n";
    }
    ~Text()     {cout<<"Text des!\n";}
};
class CircleWithText : public Point,public Text   {
public:
    CircleWithText(int cx,int cy, char *msg):
                    Point(cx,cy),Text(msg) {
        cout<<"Point with Text con!\n";
    }
    ~CircleWithText() {cout<<"Point with Text des\n";}
};
```

```
void main()
{
    CircleWithText cm(3,4,"hello");
}
```

42.

```
#include <iostream.h>
class Date
{
  public:
    void SetDate(int y,int m,int d){ Year=y; Month=m; Day=d; }
    void PrintDate(){ cout<<Year<<"/"<<Month<<"/"<<Day<<endl;}
    Date(){SetDate(2000,1,1);}
    Date(int y,int m,int d){SetDate(y,m,d);}
  protected:
    int Year,Month,Day;
};

class Time
{
  public:
    void SetTime(int h,int m,int s){ Houre=h; Minutes=m; Seconds=s;}
    void PrintTime(){ cout<<Houre<<":"<<Minutes<<":"<<Seconds<<endl;}
    Time(){SetTime(0,0,0);}
    Time(int h,int m,int s){SetTime(h,m,s);}
 protected:
V  int Houre, Minutes, Seconds;
};

class Date_Time: public Date, public Time
{
  public:
    Date_Time( ):Date(),Time(){};
    Date_Time(int y,int mo,int d,int h,int mi,int s):
                Date(y,mo,d), Time(h,mi,s){}
    void PrintDate_Time(){PrintDate();PrintTime();}
};

void main( )
{
  Date_Time dt_a, dt_b(2002,10,1,6,0,0);
  dt_a.PrintDate_Time();
  dt_b.SetTime(23,59,59);
  dt_b.PrintDate_Time();
  dt_a.SetDate(2002,12,31);
```

```
  dt_a.PrintDate_Time();
}
```

43.

```
#include <iostream.h>
class Date
{
  public:
    Date(int y=2001,int m=1,int d=1){Year=y; Month=m; Day=d;}
    void PrintDate(){ cout<<Year<<"/"<<Month<<"/"<<Day<<endl;}
  protected:
    int Year,Month,Day;
};
class Time
{
  public:
    Time(int h=5,int m=30,int s=0){Houre=h; Minutes=m; Seconds=s;}
    void PrintTime(){ cout<<Houre<<":"<<Minutes<<":"<<Seconds<<endl;}
  protected:
    int Houre, Minutes, Seconds;
};
class Date_Time: public Date, public Time
{
  public:
    Date_Time( ){};
    Date_Time(int y,int mo,int d,int h=0,int mi=0,int s=0):
                Date(y,mo,d), Time(h,mi,s){}
    void PrintDate_Time(){PrintDate();PrintTime();}
};

void main( )
{
  Date_Time a, b(2002,10,1,6,20,0), c(2003,3,8,6,7);
  a.PrintDate_Time();
  b.PrintDate_Time();
  c.PrintDate_Time();
}
```

44.

```
//**********************test.h*******************//
#include <iostream.h>
class Base
{
  public:
    Base (int i,int j){ x0=i; y0=j;}
```

```
    void Move(int x,int y){ x0+=x; y0+=y;}
    void Show(){ cout<<"Base("<<x0<<","<<y0<<")"<<endl;}
  private:
    int x0,y0;
};
class Derived: private Base
{
  public:
    Derived(int i,int j,int m,int n):Base(i,j){ x=m; y=n;}
    void Show (){cout<<"Next("<<x<<","<<y<<")"<<endl;}
    void Move1(){Move(2,3);}
    void Show1(){Base::Show();}
  private:
    int x,y;
};
//**************************test.cpp************************//
#include "test.h"
void main( )
{
  Base b(1,2);
  b.Show();
  Derived d(3,4,10,15);
  d.Move1();
  d.Show();
  d.Show1();
}
```

45.

```
/************* test.h ***********************/
#include <iostream.h>
class Point
{
  public:
    void InitP(float x0=0, float y0=0) {X=x0;Y=y0;}
    void Move(float xf, float yf) {X+=xf;Y+=yf;}
    float GetX() {return X;}
    float GetY() {return Y;}
  private:
    float X,Y;
};

class Rectangle: public Point
{
  public:
    void InitR(float x, float y, float w, float h) {
```

```
        InitP(x,y);W=w;H=h;
    }
    void ZoomR(float k){W*=k,H*=k;}
    float GetH() {return H;}
    float GetW() {return W;}
  private:
    float W,H;
};

//*************************test.cpp************************//
#include "test.h"
void main()
{
  Rectangle rect;
  rect.InitR(10,20,30,40);
  cout<<rect.GetX()<<","<<rect.GetY()<<","
      <<rect.GetW()<<","<<rect.GetH()<<endl;
  rect.Move(5,6);
  cout<<rect.GetX()<<","<<rect.GetY()<<","
      <<rect.GetW()<<","<<rect.GetH()<<endl;
  rect.ZoomR(7);
  cout<<rect.GetX()<<","<<rect.GetY()<<","
     <<rect.GetW()<<","<<rect.GetH()<<endl;
}
```

46.

```
//*********************test.h*******************//
#include <iostream.h>
class Base
{
  public:
    virtual void Set(int b){x=b;}
    virtual int Get(){ return x;}
  private:
    int x;
};
class Derived: public Base
{
  public:
    void Set(int d){y = d;}
    int Get(){return y;}
  private:
    int y;
};
//*************************test.cpp************************//
```

```
#include "test.h"
void main( )
{
  Base B_obj;
  Derived D_obj;
  Base *p=&B_obj;
  p->Set(100);
  cout<<"B_obj x="<<p->Get()<<endl;
  p=&D_obj;
  p->Set(200);
  cout<<"D_obj y="<< p->Get()<<endl;
  p->Base::Set(300);
  cout<<"B_obj x="<< p->Base::Get()<<endl;
  p->Set(p->Get()+200);
  cout<<"D_obj y="<< p->Get()<<endl;
}
```

## 五.指出程序或函数的功能

1.
```
#include<iostream.h>
void main()
{
    int i,s=0;
    for(i=2;i<=30;i+=2) s+=i*i;
    cout<<"s="<<s<<endl;
}
```

2.
```
#include<iostream.h>
#include<stdlib.h>
#include<math.h>
void main()
{
    int i=10,a;
    while(i>0) {
       a=rand()%100+10;
       int j, k=int(sqrt(a)+1e-5);   //sqrt(x)为求 x 的平方根函数
       for(j=2; j<=k; j++)
          if(a%j==0) break;
       if(j>k) {cout<<a<<' '; i--;}
    }
}
```

3.
```
void trans(int x)
{
    char a[10];
```

```
    int i=0,rem;
    do {
        rem=x%16;
        x=x/16;
        if(rem<10) a[i]=48+rem;   //’0’字符的 ASCII 码为 48
        else a[i]=55+rem;          //’A’字符的 ASCII 码为 65
        i++;
    }while(x!=0);
    while(i>0) cout<<a[??i];
    cout<<endl;
}
```

4. 
```
#include<iostream.h>
double f1(int n) {
    double sign=1,s=1;
    for(int i=2;i<=n; i++) {
        s+=sign/(i*i);
        sign*=-1;
    }
    return s;
}
void main()
{
    int a;
    cin>>a;
    cout<<f1(a)<<endl;
}
```

5. 
```
double f1(double a, double b, char op) {
    switch(op) {
      case ’+’ : return a+b;
      case ’-’ : return a-b;
      case ’*’ : return a*b;
      case ’/’ : if(b==0) {
                     cout<<"divided by 0!"<<endl;
                     exit(1);
                 }
                 else return a/b;
      default: cout<<"operator error!"<<endl;
               exit(1);
    }
}
```

6. 
```
#include<iostream.h>
#include<math.h>
```

```
void main()
{
    int x,y;
    cin>>x;
    y=int(sqrt(x));  //sqrt(x)为求 x 的算术平方根
    for(int i=1;i<=y;i++)
        if(x%i==0) cout<<” x=” <<i<<’ *’ <<x/i<<endl;
}
```

7. 
```
#include<iostream.h>
void main()
{
    int i,p=1,s=0;
    int N;
    cout<<"输入一个正整数:";
    cin>>N;
    for(i=1;i<=N;i++) {
        p*=i;
        s+=p;
    }
    cout<<s<<endl;
}
```

8. 
```
#include<iostream.h>
#include<stdlib.h>
#include<time.h>
const N=10;
int ff(int x, int y) {
    int z;
    cout<<x<<'+'<<y<<'=';
     cin>>z;
    if(x+y==z) return 1; else return 0;
}
void main()
{
    int a,b,c=0;
    srand(time(0));  //初始化随机数序列
    for(int i=0;i<N;i++) {
        a=rand()%20+1;    //rand()函数产生 0-32767 之间的一个随机数
        b=rand()%20+1;
        c+=ff(a,b);
    }
    cout<<"得分:"<<c*10<<endl;
}
```

9. int s1(int n)
```
{
    int x;
    if(n==1) x=1;
    else x=s1(n-1)+n*n;
    return x;
}
```

10. void fun5(char* a, const char* b)
```
{
    while(*b) *a++=*b++;
    *a=0;
}
```

11. template<class T>
```
bool fun8(T a[], int n, T key)
{
    for(int i=0;i<n;i++)
        if(a[i]==key) return true;
    return false ;
}
```

12. void f2(double a[], int n)
```
{
    int i; double sum=0;
    for(i=0;i<n;i++) sum+=a[i];
    sum/=n;
    for(i=0;i<n;i++)
        if(a[i]>=sum) cout<<a[i]<<' ';
    cout<<endl;
}
```

13. void f4(char a[M][N])
```
{
    int c1,c2,c3;
    c1=c2=c3=0;
    for(int i=0;i<M;i++)
        if(strlen(a[i])<5) c1++;
        else if(strlen(a[i])>=5 && strlen(a[i])<15) c2++;
        else c3++;
    cout<<c1<<' '<<c2<<' '<<c3<<endl;
}
```

14. void fun3(int a[][N], int m, int& row, int& col)
```
{
```

```
    int x=a[0][0];
    row=col=0;
    for(int i=0;i<m;i++)
        for(int j=0;j<N;j++)
            if(a[i][j]>x) {
                x=a[i][j]; row=i; col=j;
            }
}
```

15.
```
int fun6(int m, int n, int b=2)
{
    if(m<b && n<b) return m*n;
    else if(m%b==0 && n%b==0) return b*fun6(m/b,n/b,b);
    else return fun6(m,n,++b);
}
```

16.
```
char* f8(char* str1, const char* str2)
{
    int i=0,j=0;
    while(str1[i]) i++;
    while(str2[j]) str1[i++]=str2[j++] ;
    str1[i]='\0';
    return str1;
}
```

17.
```
int f8(const char* str1, const char* str2)
{
    int i=0;
    while(str1[i] && str2[i])
        if(str1[i]==str2[i]) i++;
        else if(str1[i]>str2[i]) return 1;
        else return -1;
    if(str1[i]==str2[i]) return 0;
    else if(str1[i]>str2[i]) return 1;
    else return -1;
}
```

18.
```
IntNode* FindMax(IntNode *f)
{
    if(!f) return NULL;
    IntNode *p=f;
    f=f->next;
    while(f) {
        if(f->data>p->data) p=f;
        f=f->next;
```

```
    }
    return p;
}
```
假定 IntNode 的类型定义为：
```
struct IntNode {
    int data;        //结点值域
    IntNode* next;   //结点指针域
};
```

19. 
```
int Count(IntNode *f)
{
    if(!f) return 0;
    int c=0;
    while(f) {
        c++;
        f=f->next;
    }
     return c;
}
```
假定 IntNode 的类型定义为：
```
struct IntNode {
    int data;        //结点值域
    IntNode* next;   //结点指针域
};
```

20. 
```
void Output(IntNode *f)
{
    if(!f) return;
    while(f) {
        cout<<f->data<<' ';
        f=f->next;
    }
    cout<<endl;
}
```
假定 IntNode 的类型定义为：
```
struct IntNode {
    int data;        //结点值域
    IntNode* next;   //结点指针域
};
```

21. 
```
void Input(IntNode*& f)
{
    int n;
    cout<<"从键盘给 n 输入一个整数:";
    do cin>>n; while(n<0);
```

```
    if(n==0) {f=NULL; return;}
    f=new IntNode;
    IntNode* p=f;
    cout<<”从键盘输入”<<n<<”个整数:”;
    while(n--) {
        p=p->next=new IntNode;
        cin>>p->data;
    }
    p->next =NULL;
    p=f; f=f->next; delete p;
}
```

假定 IntNode 的类型定义为：

```
struct IntNode {
    int data;        //结点值域
    IntNode* next;   //结点指针域
};
```

22. 
```
int f(const char *s)
{
    int i=0;
    while(*s++)i++;
    return i;
};
```

23. 
```
char *f(char *s){
    int n=strlen(s);
    char* r=new char[n+1];
    for(int i=0; i<n; i++)
        if(s[i]>='a' && s[i]<='z') r[i]=s[i]-'a'+'A';
        else r[i]=s[i];
    r[n]=’\0’;
    return r;
}
```

# 六.程序改错

1．在下面的定义中，NODE 是链表结点的结构，appendToList 则是一函数，其功能是：在 list 所指向的链表的末尾添加一个新的值为 x 的结点，并返回表头指针。函数中有两处错误，指出错误所在行的行号并提出改正意见。

```
struct NODE{
    int data;
    NODE *next;
};
    NODE* appendToList(NODE *list, int x){      //1 行
        NODE *p=new int;                        //2 行
        p->data=x;                              //3 行
```

```
        p->next=NULL;                                   //4 行
        if(list==NULL) return p;                        //5 行
        NODE *p1=list;                                  //6 行
        while(p1->next!=NULL) p1=p1->next;              //7 行
        p1=p;                                           //8 行
        return list;
    }
```

错误行的行号为_____和_____。

分别改正为_____和_____。

2．在下面的定义中，NODE 是链表结点的结构，addToList 则是一函数，其功能是：将一个值为 x 的新结点添加到以 plist 为表头指针的链表的首部(即第一个结点的前面)并返回表头指针。函数中有两处错误，指出错误所在行的行号并提出改正意见。

```
struct NODE{
    int data;
    NODE *next;
};
    NODE* adndToList(NODE * plist, int x){      //1 行
        NODE *p;                                //2 行
        *p=new NODE;                            //3 行
        p->data=x;                              //4 行
        p->next=NULL;                           //5 行
        plist=p;                                //6 行
        return p;                               //7 行
}
```

错误行的行号为_____和_____。

分别改正为_____和_____。

3．假定要求下面程序的输出结果为“11/15”,其主函数中存在着三行语句错误，请指出错误语句行的行号并改正错误行。

```
#include<iostream.h>
class Franction {   //定义分数类
    int nume;   //定义分子
    int deno;   //定义分母
  public:
  //把*this 化简为最简分数，具体定义在另外文件中实现
    void FranSimp();
  //返回两个分数*this 和 x 之和，具体定义在另外文件中实现
    Franction FranAdd(const Franction& x);
  //置分数的分子和分母分别 0 和 1
    void InitFranction() {nume=0; deno=1;}
  //置分数的分子和分母分别 n 和 d
    void InitFranction(int n, int d) {nume=n; deno=d;}
```

```
  //输出一个分数
    void FranOutput() {cout<<nume<<'/'<<deno<<endl;}
};

void main()                                    //1 行
{                                              //2 行
    Franction a,b,c;                           //3 行
    a.InitFranction(6,15);                     //4 行
    b.InitFranction(1);                        //5 行
    c.InitFranction();                         //6 行
    c=FranAdd(a,b);                            //7 行
    cout<<c.nume<<’/’<<c.deno<<endl;           //8 行
}                                              //9 行
```

错误行的行号为_____、_____和_____。

分别改正为_____、_____和_____。

4．假定要求下面程序的输出结果为“23/20”,其主函数中存在着三条语句错误，请指出错误语句行的行号并改正。

```
#include<iostream.h>
class Franction {   //定义分数类
    int nume;   //定义分子
    int deno;   //定义分母
  public:
  //把*this 化简为最简分数，具体定义在另外文件中实现
    void FranSimp();
  //返回两个分数*this 和 x 之和，具体定义在另外文件中实现
    Franction FranAdd(const Franction& x);
  //置分数的分子和分母分别 0 和 1
    void InitFranction() {nume=0; deno=1;}
  //置分数的分子和分母分别 n 和 d
    void InitFranction(int n, int d) {nume=n; deno=d;}
  //输出一个分数
    void FranOutput() {cout<<nume<<'/'<<deno<<endl;}
};

void main()                                    //1 行
{                                              //2 行
    Franction *a=new Franction;                //3 行
    Franction *b=new Franction;                //4 行
    a->InitFranction(6,15);                    //5 行
    b.InitFranction(3,4);                      //6 行
    Franction c;                               //7 行
    c.InitFranction();                         //8 行
    c=a.FranAdd(b);                            //9 行
    cout<<c.FranOutput()<<endl;                //10 行
```

```
}                                                    //11 行
```

错误行的行号为_____、_____和_____。

分别改正为_____、_____和_____。

5．下面是一个类的定义，存在着 3 处语法错误，请指出错误行的行号并改正。

```
class CE {                                           //1 行
      private:                                       //2 行
         int a,b;                                    //3 行
         int getmin() {return (a<b? a:b);}           //4 行
      public                                         //5 行
         int c;                                      //6 行
         void SetValue(int x1,int x2, int x3) {      //7 行
              a=x1; b=x2; c=x3;                      //8 行
         };                                          //9 行
         int GetMin();                               //10 行
};                                                   //11 行
int GetMin() {                                       //12 行
    int d=getmin();                                  //13 行
    return (d<c? d:c);                               //14 行
}                                                    //16 行
```

错误行的行号为_____、_____和_____。

分别改正为_____、_____和_____。

6．下面程序段第 4-10 行中存在着三条语句的语法错误，请指出错误语句的行号并改正。

```
class A {                                            //1 行
   int a,b;                                          //2 行
   const int c;                                      //3 行
 public                                              //4 行
   A():c(0);a(0);b(0) {}                             //5 行
   A(int aa, int bb) c(aa+bb); {a=aa; b=bb;}         //6 行
 };                                                  //7 行
 A a,b(1,2);                                         //8 行
 A *x=&a, &y=b;                                      //9 行
 A *z=new A, w[10];                                  //10 行
```

错误行的行号为_____、_____和_____。

分别改正为_____、_____

和_____。

```
 Public:   A():c(0),a(0),b(0) {}
A(int aa, int bb): c(aa+bb) {a=aa; b=bb;}
```

7．下面程序段第 4-9 行中存在着三条语句错误，请指出错误语句的行号并说明原因。

```
class A {                                            //1 行
```

```
  int a,b;                                          //2 行
  const int c;                                      //3 行
public:                                             //4 行
  A() {a=b=c=0;}                                      //5 行
  A(int aa, int bb):c(aa+bb) {a=aa; b=bb;}          //6 行
};                                                  //7 行
A a,b(1,2,3);                                       //8 行
A x(2,3), y(4);                                     //9 行
```

错误行的行号为_____、_____和_____。

错误原因分别为_____、_____和_____。

8．下面程序段第 10-17 行中存在着三条语句错误，请指出错误语句的行号并说明原因。

```
class A {                                           //1 行
   int a;                                           //2 行
public:                                             //3 行
   A(int aa=0):a(aa){}                                //4 行
};                                                  //5 行
class B {                                           //6 行
   int a,b;                                         //7 行
   const int c;                                     //8 行
   A d;                                             //9 行
 public:                                            //10 行
   B():c(0) {a=b=0;}                                  //11 行
   B(int aa, int bb):d(aa+bb) {                     //12 行
      a=aa; b=bb; c=aa-bb;                            //13 行
   }                                                //14 行
 }                                                  //15 行
 B a,b(1,2);                                        //16 行
 B x=a,y(b),z(1,2,3),;                              //17 行
```

错误行的行号为_____、_____和_____。

错误原因分别为_____、_____

和_____。

9．假定要求下面程序输出结果为“d=800,f=60”，在第 4-23 行中存在着三条语句错误，请指出错误语句的行号并改正。

```
#include<iostream.h>
class A {                                           //1 行
   int a[10]; int n;                                //2 行
 public:                                            //3 行
   A(int aa[], int nn): n(nn) {                     //4 行
      for(int i=0; i<n; i++) aa[i]=a[i];            //5 行
   }                                                //6 行
   int Get(int i) {return a[i];}                    //7 行
```

```
    int SumA(int n);                          //8 行
};                                            //9 行
int A::SumA(int n) {                          //10 行
    int s=0;                                  //11 行
    for(int j=0; j<n; j++) s+=a[j];           //12 行
    return s;                                 //13 行
}                                             //14 行
void main() {                                 //15 行
    int a[]={2,5,8,10,15,20};                 //16 行
    A x(a,6);                                 //17 行
    int d=1;                                  //18 行
    for(int i=0; i<4; i++) d*=x.a[i];         //19 行
    int f=SumA(6);                            //20 行
    cout<<"d="<<d<<’ ,’ ;                     //21 行
    cout<<"f="<<f<<endl;                      //22 行
}                                             //23 行
```

错误行的行号为_____、_____和_____。5 19 20

分别改正为_____、_____和_____。

10．下面是分数类 fract 的定义及测试主程序，在类定义及其友元函数定义中有两处错误，更正错误后程序应显示 41/28，请指出错误所在行的行号并给出改正意见。

```
class fract{
    int den;    //分子
    int num;    //分母
public:
    fract(int d=0,int n=1):den(d),num(n){}      //1 行
    friend fract &operator+=(fract,fract&);     //2 行
    void show(){ cout<<den<<'/'<<num;}          //3 行
};                                              //4 行
friend fract &operator+=(fract f1,fract f2)     //5 行
{                                               //7 行
    f1.den=f1.den*f2.num+f1.num*f2.den;         //8 行
    f1.num*=f2.num;                             //9 行
    return f1;                                  //10 行
}
void main(){
    fract fr(3,4);
    fr+=fract(5,7);
    fr.show();
}
```

错误行的行号为_____和_____。

分别改正为_____

和_____。

# 七.程序设计

1．编一程序求出满足不等式 1+1/2+1/3+...+1/n≥5 的最小 n 值。

2．计算 $1+3+3^2+\cdots+3^{10}$ 的值并输出，假定分别用 i,p,s 作为循环变量、累乘变量和累加变量的标识符。

3．求满足不等式 $2^2+4^2+\cdots+n^2<1000$ 的最大 n 值，假定分别用 i 和 s 作为取偶数值和累加值的变量，并限定使用 do 循环编程。

4．已知，$y=\begin{cases}\sqrt{a^2+x^2} & (x\le 0)\\ 3ax^2+4ax-1 & (x>0)\end{cases}$ 求出并显示当 x 分别取-3.8, 6.4, 2.3, -4.2,8.9, 3.5, -5.0, 4.5 时所对应的 y 值，要求把 a 定义为常量，其值设定为 10.2，x 的每个值由键盘输入，并假定用-100 作为终止标志，求平方根函数为 sqrt(x)。

5．求出从键盘上输入的 10 个整数中的最大值，要求输入变量用 x 表示，存储最大值的变量用 max 表示。

6．已知 6≤a≤30，15≤b≤36，求满足不定方程 2a+5b=126 的全部整数组解。如(13, 20)就是一个整数组解，并以所给的样式输出每个解。

7．某班级学生进行百米跑测试，规定成绩在 12 秒以内（含 12 秒）为优秀，在 12 秒以上至 15 秒为达标，在 15 秒以上为不达标，编一程序，从键盘上输入每个人的成绩，以 x 作为输入变量，并以小于 0 的任何数作为终止标志，分别用变量 c1,c2 和 c3 统计出成绩为优秀、达标和不达标的人数。

8．编写一个函数，分别求出由指针 a 所指向的字符串中包含的每种十进制数字出现的次数，把统计结果保存在数组 b 的相应元素中。

9．按照下面函数原型语句编写一个函数，返回二维数组 a[m][n]中所有元素的平均值，假定采用变量 v 存放平均值。

10．按照下面函数原型语句编写一个递归函数计算出数组 a 中 n 个元素的平方和并返回。

```
int f(int a[],int n);
```

11．按照函数原型语句“void p(int n);”编写一个递归函数显示出如下图形，此图形是 n=5 的情况。

```
55555
4444
333
22
1
```

12．按照函数原型语句“void p(int n);”编写一个递归函数显示出如下图形，此图形是 n=5 的情况。

```
1
22
333
4444
55555
```

13．根据下面类中 Count 函数成员的原型和注释写出它的类外定义。

```
class AA {
    int* a;
    int n;
    int MS;
public:
    void InitAA(int aa[], int nn, int ms) {
```

```
        if(nn>ms) {cout<<"Error!"<<endl; exit(1);}
        MS=ms;
        n=nn;
        a=new int[MS];
        for(int i=0; i<MS; i++) a[i]=aa[i];
    }
    int Count(int x);   //从数组 a 的前 n 个元素中统计出其
                        //值等于 x 的个数并返回。
};
```

14. 根据下面类中 Search 函数成员的原型和注释写出它的类外定义。

```
class AA {
    int* a;
    int n;
    int MS;
public:
    void InitAA(int aa[], int nn, int ms) {
        if(nn>ms) {cout<<"Error!"<<endl; exit(1);}
        MS=ms;
        n=nn;
        a=new int[MS];
        for(int i=0; i<MS; i++) a[i]=aa[i];
    }
    int Search(int x); //从数组 a 的前 n 个元素中顺序查找值为 x 的元素，
                       //若查找成功则返回元素的下标，否则返回-1。
};
```

15. 根据下面类中 MaxMin 函数成员的原型和注释写出它的类外定义。

```
class AA {
    int* a;
    int n;
    int MS;
public:
    void InitAA(int aa[], int nn, int ms) {
        if(nn>ms) {cout<<"Error!"<<endl; exit(1);}
        MS=ms;
        n=nn;
        a=new int[MS];
        for(int i=0; i<MS; i++) a[i]=aa[i];
    }
    int MaxMin(int& x, int& y); //从数组 a 的前 n 个元素中求出
   //最大值和最小值，并分别由引用参数 x 和 y 带回，
   //同时若 n 大于 0 则返回 1，否则返回 0。
};
```

16. 根据下面类中 Compare 函数成员的原型和注释写出它的类外定义。

```
class AA {
    int* a;
    int n;
    int MS;
public:
    void InitAA(int aa[], int nn, int ms) {
        if(nn>ms) {cout<<"Error!"<<endl; exit(1);}
        MS=ms;
        n=nn;
        a=new int[MS];
        for(int i=0; i<MS; i++) a[i]=aa[i];
    }
    int Compare(AA b); //比较*this 与 b 的大小，若两者中
        //的 n 值相同，并且数组中前 n 个元素值对应
        //相同，则认为两者相等返回 1，否则返回 0。
};
```

17. 根据下面类中 CompareBig 函数成员的原型和注释写出它的类外定义。

```
class AA {
    int* a;
    int n;
    int MS;
public:
    void InitAA(int aa[], int nn, int ms) {
        if(nn>ms) {cout<<"Error!"<<endl; exit(1);}
        MS=ms;
        n=nn;
        a=new int[MS];
        for(int i=0; i<MS; i++) a[i]=aa[i];
    }
int CompareBig(AA b);  //比较*this 与 b 的大小，从前向后按两数组
     //中的对应元素比较，若*this 中元素值大则返回 1，若 b 中
     //元素值大则返回-1，若相等则继续比较下一个元素，直到
     //一个数组中无元素比较，此时若两者的 n 值相同则返回 0，
     //否则若*this 中的 n 值大则返回 1，若 b 中的 n 值大则返回-1。
};
```

18. 根据下面类中 Reverse 函数成员的原型和注释写出它的类外定义。

```
class AA {
    int* a;
    int n;
    int MS;
public:
    void InitAA(int aa[], int nn, int ms) {
```

```
        if(nn>ms) {cout<<"Error!"<<endl; exit(1);}
        MS=ms;
        n=nn;
        a=new int[MS];
        for(int i=0; i<MS; i++) a[i]=aa[i];
    }
    AA* Reverse();  //对于调用该函数的对象，将其 a 数组中前 n 个
     //元素值按相反的次序排列，返回指向该对象的指针。
};
```

19. 根据下面类中 Reverse1 函数成员的原型和注释写出它的类外定义。

```
class AA {
    int* a;
    int n;
    int MS;
public:
    void InitAA(int aa[], int nn, int ms) {
        if(nn>ms) {cout<<"Error!"<<endl; exit(1);}
        MS=ms;
        n=nn;
        a=new int[MS];
        for(int i=0; i<MS; i++) a[i]=aa[i];
    }
    AA* Reverse1();  //通过动态存储分配得到一个对象，并动态分配
     //a[MS]数组空间,要求该对象中的 n 和 MS 的值与*this 中的
     //对应成员的值相同，数组元素的值是按照*this 中数组元
     //素的相反次序排列得到的，要求该函数返回动态对象的地址。
};
```

20. 根据下面类中构造函数的原型和注释写出它的类外定义。

```
class Array {
    int *a;  //指向动态分配的整型数组空间
    int n;    //记录数组长度
public:
    Array(int aa[], int nn);  //构造函数，利用 aa 数组长度 nn 初始化 n，
                              //利用 aa 数组初始化 a 所指向的数组空间
    Array(Array& aa);   //拷贝构造函数
    Array& Give(Array& aa);  //实现 aa 赋值给*this 的功能并返回*this
    Array Uion(Array& aa);   //实现*this 和 aa 中的数组合并的
                             //功能，把合并结果存入临时对象并返回
    int Lenth() {return n;}  //返回数组长度
    void Print() {          //输出数组
        for(int i=0; i<n; i++)
            cout<<a[i]<<' ';
        cout<<endl;
```

```
    }
};
```

21. 根据下面类中拷贝构造函数的原型写出它的类外定义。

```
class Array {
    int *a;  //指向动态分配的整型数组空间
    int n;     //记录数组长度
public:
    Array(int aa[], int nn);  //构造函数，利用 aa 数组长度 nn 初始化 n，
                              //利用 aa 数组初始化 a 所指向的数组空间
    Array(Array& aa);  //拷贝构造函数
    Array& Give(Array& aa);  //实现 aa 赋值给*this 的功能并返回*this
    Array Uion(Array& aa);    //实现*this 和 aa 中的数组合并的
                              //功能，把合并结果存入临时对象并返回
    int Lenth() {return n;}  //返回数组长度
    void Print() {              //输出数组
        for(int i=0; i<n; i++)
            cout<<a[i]<<' ';
        cout<<endl;
    }
};
```

22. 根据下面类中 Give 函数的原型和注释写出它的类外定义。

```
class Array {
    int *a;  //指向动态分配的整型数组空间
    int n;     //记录数组长度
public:
    Array(int aa[], int nn);  //构造函数，利用 aa 数组长度 nn 初始化 n，
                              //利用 aa 数组初始化 a 所指向的数组空间
    Array(Array& aa);  //拷贝构造函数
    Array& Give(Array& aa);  //实现 aa 赋值给*this 的功能并返回*this
    Array Uion(Array& aa);  //实现*this 和 aa 中的数组合并的
                            //功能，把合并结果存入临时对象并返回
    int Lenth() {return n;}  //返回数组长度
    void Print() {         //输出数组
        for(int i=0; i<n; i++)
            cout<<a[i]<<' ';
        cout<<endl;
    }
};
```

23. 根据下面类中 Uion 函数的原型和注释写出它的类外定义。

```
class Array {
    int *a;  //指向动态分配的整型数组空间
    int n;     //记录数组长度
```

```
public:
    Array(int aa[], int nn);  //构造函数，利用 aa 数组长度 nn 初始化 n，
                  //利用 aa 数组初始化 a 所指向的数组空间
    Array(Array& aa);  //拷贝构造函数
    Array& Give(Array& aa);  //实现 aa 赋值给*this 的功能并返回*this
    Array Uion(Array& aa);  //实现*this 和 aa 中的数组合并的功能，把合并
                  //结果(其长度为两数组长度之和)存入临时对象并返回
    int Lenth() {return n;}  //返回数组长度
    void Print() {         //输出数组
        for(int i=0; i<n; i++)
            cout<<a[i]<<' ';
        cout<<endl;
    }
};
```

24. 根据下面类中构造函数的原型和注释写出它的类外定义。

```
class Strings {
    char *s;  //指向动态分配的字符串数组空间
    int n;    //记录字符串长度
public:
    Strings(char*str);  //构造函数，利用 str 字符串长度初始化 n，
                        //利用 str 字符串初始化 s 所指的字符串空间
    Strings(Strings& str);          //拷贝构造函数
    Strings& Give(Strings& str);  //实现 str 赋值给*this 的功能
    Strings Uion(Strings& str);   //实现*this 和 str 中的字符串合并的
                  //功能，把合并结果存入临时对象并返回
    int Lenth() {return n;}         //返回字符串长度
    void Print() {cout<<s<<endl;} //输出字符串
};
```

25. 根据下面类中拷贝构造函数的原型写出它的类外定义。

```
class Strings {
    char *s;  //指向动态分配的字符串数组空间
    int n;    //记录字符串长度
public:
    Strings(char*str);  //构造函数，利用 str 字符串长度初始化 n，
                  //利用 str 字符串初始化 s 所指的字符串空间
    Strings(Strings& str);          //拷贝构造函数
    Strings& Give(Strings& str);  //实现 str 赋值给*this 的功能
    Strings Uion(Strings& str);   //实现*this 和 str 中的字符串合并的
                  //功能，把合并结果存入临时对象并返回
    int Lenth() {return n;}         //返回字符串长度
    void Print() {cout<<s<<endl;} //输出字符串
};
```

26. 根据下面类中 Give 函数的原型和注释写出它的类外定义。

```
class Strings {
    char *s;  //指向动态分配的字符串数组空间
    int n;    //记录字符串长度
public:
    Strings(char*str);  //构造函数，利用 str 字符串长度初始化 n，
                        //利用 str 字符串初始化 s 所指的字符串空间
    Strings(Strings& str);  //拷贝构造函数
    Strings& Give(Strings& str);  //实现 str 赋值给*this 的功能并返回*this
    Strings Uion(Strings& str);  //实现*this 和 str 中的字符串合并的
                        //功能，把合并结果存入临时对象并返回
    int Lenth() {return n;}  //返回字符串长度
    void Print() {cout<<s<<endl;}  //输出字符串
};
```

27. 根据下面类中 Uion 函数的原型和注释写出它的类外定义。

```
class Strings {
    char *s;  //指向动态分配的字符串数组空间
    int n;    //记录字符串长度
public:
    Strings(char*str);  //构造函数，利用 str 字符串长度初始化 n，
                        //利用 str 字符串初始化 s 所指的字符串空间
    Strings(Strings& str);  //拷贝构造函数
    Strings& Give(Strings& str);  //实现 str 赋值给*this 的功能并返回*this
    Strings Uion(Strings& str);   //实现*this 和 str 中的字符串连接的功能，
                                  //把连接结果存入临时对象并返回
    int Lenth() {return n;}  //返回字符串长度
    void Print() {cout<<s<<endl;}  //输出字符串
};
```

28. 下列程序段中，A_class 的成员函数 Variance()可求出两数的平方差，请改写该程序段，把 Variance()函数从 A_class 类中分离出来，用友元函数来实现该函数的功能。

```
class A_class {
  private:
    int x,y,t;
  public:
    A_class(int i,int j):x(i),y(j) {
      if(y>x){t=x;x=y;y=t;}
    }
  int Variance(){return x*x-y*y;}
  //其它函数从略
};
void main() {
  A_class A_obj(3,5);
  cout<<"Result:"<<A_obj.Variance()<<endl;
```

```
}
```

29．下面给出了矩阵类 Matrix 定义。为了求两个矩阵对象的乘积，需要定义一个 Matrix 的友元函数 Multiply()。请按照友元函数 Multiply（）的声明编写出该函数的定义。

```
class Matrix {
  public:
    Matrix(int row,int col);    //构造一个具有 row 行 col 列的矩阵
    ~Matrix() {delete []mem;}          //析构函数
    friend bool Multiply(Matrix &m1, Matrix &m2, Matrix &m3);
          //定义 Multiply()为友元函数，该函数把 m1×m2 的值赋给 m3
          //其他成员函数从略
  private:
    int *mem;                          //动态申请矩阵空间
    const int rows,cols;                //矩阵的行数和列数
};
Matrix::Matrix(int row,int col):rows(row),cols(col)
{
    mem = new int[row*col];
}
bool Multiply(Matrix &m1, Matrix &m2, Matrix &m3)
{
//确定矩阵是否能够进行相乘
  if(m1.rows != m3.rows ||m2.cols != m3.cols || m1.cols != m2.rows)    return false;
//定义 sum 变量，用于计算乘积矩阵 m3 中每个元素的值
  int sum;
//请在下面编写剩余部分
}
```

30．已知类定义如下，其中 Shape 为基类，Circle 和 Rectangle 分别 Shape 的直接派生类，Square 为 Rectangle 的直接派生类和 Shape 的间接派生类。请模仿 Circle 类，写出 Rectangle 类的所有成员函数。

```
/******************文件 shape.h*************************/
const float PI=3.14159f;  //定义圆周率常量

class Shape    //几何图形抽象类
{
  public:
    virtual float GetPerimeter()=0;      //纯虚函数，计算周长
    virtual float GetAre()=0;             //纯虚函数，计算面积
};

class Circle: public Shape     //圆类
{
  public:
    Circle(float rad):rad(rad){}
    ~Circle(){}
    float GetPerimeter() {return 2*PI*rad;}    //计算圆形周长
```

```
    float GetAre() {return PI*rad *rad;}            //计算圆形面积
  private:
    float rad;  //圆的半径
};

class Rectangle: public Shape     //矩形类
{
  public:  //在下面编写每个成员函数

  private:
    float length, width;  //矩形的长和宽
};

class Square: public Rectangle    //正方形类
{
  public:
    Square(float len): Rectangle(len,len){}
    ~Square(){}
};
```

31．已知类定义如下，其中 Shape 为基类，Circle 和 Rectangle 分别 Shape 的直接派生类，Square 为 Rectangle 的直接派生类和 Shape 的间接派生类。请模仿 Rectangle 类，写出 Circle 类的所有成员函数。

```
/******************文件 shape.h*************************/
const float PI=3.14159f;  //定义圆周率常量
class Shape    //几何图形抽象类
{
  public:
    virtual float GetPerimeter()=0;      //纯虚函数，计算周长
    virtual float GetAre()=0;             //纯虚函数，计算面积
};
class Rectangle: public Shape     //矩形类
{
  public:
    Rectangle (float len,float wid):length(len),width(wid){}
    ~Rectangle (){}
    float GetPerimeter() {return 2*(length+width);}    //计算矩形周长
    float GetAre() {return length*width;}              //计算矩形面积
  private:
    float length, width;  //矩形的长和宽
};
class Circle: public Shape     //圆类
{
  public:  //在下面编写每个成员函数
private:
    float rad;  //圆的半径
};
```

```
class Square: public Rectangle    //正方形类
{
public:
    Square(float len): Rectangle(len,len){}
    ~Square(){}
};
```