

一、单项选择题

1. 面向对象软件开发中使用的 OOA 表示 (面向对象分析)。
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
2. 面向对象软件开发中使用的 OOD 表示 (面向对象设计)。
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
3. 关于面向对象系统分析, 下列说法中不正确的是 (面向对象分析阶段对问题域的描述比实现阶段更详细)。
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
4. 在一个类的定义中, 包含有 (数据和函数) 成员的定义。
A. 数据 B. 函数 C. 数据和函数 D. 数据或函数
5. 在类作用域中能够通过直接使用 该类的 (任何) 成员名 进行访问。
A. 私有 B. 公用 C. 保护 D. 任何
6. 在关键字 **public** 后面定义的成员为类的 (公用) 成员。
A. 私有 B. 公用 C. 保护 D. 任何
7. 在关键字 **private** 后面定义的成员为类的 (私有) 成员。
A. 私有 B. 公用 C. 保护 D. 任何
8. 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 (x.a)。
A. x(a) B. x[a] C. x->a D. x.a
9. 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 (x->a())。
A. x.a B. x.a() C. x->a D. x->a()
答案有错吧
10. 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 (px->a)。
A. px(a) B. px[a] C. px->a D. px.a
11. 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 (x.GetValue())。
A. x.a B. x.a() C. x->GetValue() D. x.GetValue()
12. 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 (int AA::a())。
A. int AA::a() B. int AA:a()
C. AA::a() D. AA::int a()
int AA::a()
13. 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. a B. AA::a C. a() D. AA::a()
14. 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 (inline) 放在函数原型或函数头的前面。

A. in B. inline C. inLine D. InLiner

15. 在多文件结构的程序中，通常把类的定义单独存放于（头文件）中。

A. 主文件 B. 实现文件 C. 库文件 D. 头文件

16. 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（实现文件）中。

A. 主文件 B. 实现文件 C. 库文件 D. 头文件

17. 在多文件结构的程序中，通常把含有 `main()` 函数的文件称为（主文件）。

A. 主文件 B. 实现文件 C. 程序文件 D. 头文件

18. 一个 C++ 程序文件的扩展名为（.cpp）。

A. .h B. .c C. .cpp D. .cp

19. 在 C++ 程序中使用的 `cin` 标识符是系统类库中定义的（`istream`）类中的一个对象。

A. `istream` B. `ostream` C. `iostream` D. `fstream`

20. 在 C++ 程序中使用的 `cout` 标识符是系统类库中定义的（`ostream`）类中的一个对象。

A. `istream` B. `ostream` C. `iostream` D. `fstream`

21. 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（`AA*`）。

A. `int` B. `char` C. `AA` D. `AA*`

22. 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（`this`）。

A. `abc` B. `*this` C. `this` D. `this&`

23. 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（已经被改变）。

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

24. 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（不变）。

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

25. 类中定义的成员默认为（`private`）访问属性。

A. `public` B. `private` C. `protected` D. `friend`

26. 结构中定义的成员默认为（`public`）访问属性。

A. `public` B. `private` C. `protected` D. `friend`

27. 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（`n-1`）。

A. `n` B. `n+1` C. `n-1` D. `n-2`

28. 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（`(first+length-1)%MS`）。

A. `length+1` B. `first+length`
C. `(first+length-1)%MS` D. `(first+length)%MS`

29. 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（`(first+length)%MS`）。

A. `length+1` B. `first+length`

C. $(first+length-1)\%MS$ D. $(first+length)\%MS$

30. 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. $(first+length)\%MS$

31. 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($elemTail->next$ 和 $elemTail$)。

A. $elemHead$ B. $elemTail$
C. $elemHead->next$ 和 $elemHead$ D. $elemTail->next$ 和 $elemTail$

32. 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($elemHead$ 和 $elemTail$)。

A. $elemHead$ B. $elemTail$
C. $elemHead$ 和 $elemTail$ D. $elemHead$ 或 $elemTail$

33. 队列具有(先进先出)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

34. 栈具有(先进后出)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

35. 对于一个类的构造函数, 其函数名与类名(完全相同)。

A. 完全相同 B. 基本相同 C. 不相同 D. 无关系

36. 对于一个类的析构函数, 其函数名与类名(只相差一个字符)。

A. 完全相同 B. 完全不同 C. 只相差一个字符 D. 无关系

37. 类的构造函数是在定义该类的一个(对象)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

38. 类的析构函数是一个对象被(撤消)时自动调用的。

A. 建立 B. 撤消 C. 赋值 D. 引用

39. 一个类的构造函数通常被定义为该类的(公用)成员。

A. 公用 B. 保护 C. 私有 D. 友元

40. 一个类的析构函数通常被定义为该类的(公用)成员。

A. 私有 B. 保护 C. 公用 D. 友元

41. 假定 AB 为一个类, 则执行 " $AB\ x;$ " 语句时将自动调用该类的(带参构造函数)。

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

42. 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(带参构造函数)。

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

43. 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 $*s$ 。

A. s B. $s->a$ C. $s.a$ D. $*s$

44. 假定 AB 为一个类, 则执行 " $AB\ r1=r2;$ " 语句时将自动调用该类的(拷贝构造函数)。

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

45. 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(构造)函数中完成。

A. 析构 B. 构造 C. 任一成员 D. 友元

46. 当类中的一个整型指针成员指向一块具有 $n * \text{sizeof}(\text{int})$ 大小的存储空间时，它最多能够存储(n)个整数。

A. n B. $n+1$ C. $n-1$ D. 1

47. 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(4 和 20)。

A. 4 和 5 B. 5 和 4 C. 4 和 20 D. 20 和 5

48. 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(4 和 0)。

A. 1 和 0 B. 1 和 4 C. 4 和 1 D. 4 和 0

49. 假定 AB 为一个类，则(AB(AB& x))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);

50. 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(3)。

A. 1 B. 2 C. 3 D. 4

二、填空

1. 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 $4*i$ 字节。

2. 假定 a 是一个一维数组，则 a[i] 的指针访问方式为 $*(a+i)$ 。

3. 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 $a+i*\text{sizeof}(a[i])$ 。

4. 一个数组的数组名实际上是指向该数组第一个元素的指针，并且在任何时候都不允许修改它。

5. 假定指向一维数组 b 中元素 b[4] 的指针为 p，则 p+3 所指向的元素为 b[7]，p-2 所指向的元素为 b[2]。

6. 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 int *p=&x。

7. 若 p 指向 x，则 *p 与 x 的表示是等价的。

8. 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 int[n]，访问 a[i] 时返回值的类型为 int *。

9. 假定一个二维数组为 c[5][8]，则 c[3] 的值为二维元素 c[3][0] 的地址，c[3]+2 的值为二维元素 c[3][2] 的地址。

假定 p 为指向二维数组 int d[4][6] 的指针，则 p 的类型为 int(*)[6]。

11. 假定 a 是一个二维数组，则 a[i][j] 的指针访问方式为 *(a[i]+j) 或 *(*(a+i)+j) 或 *(a+i)[j]。

12. 若要把 y 定义为整型变量 x 的引用，则所使用的定义语句为 int &y=x;。

13. 若 y 是 x 的引用，则对 y 的操作就是对 x 的操作。

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

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

16. 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 p[0] 或 *p。

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

18. 执行 new char[m][n] 操作时的返回值的类型为 char(*)[n]。

19. 执行__ delete p ____操作将释放由 p 所指向的动态分配的数据空间。
20. 执行__ delete []p ____操作将释放由 p 所指向的动态分配的数组空间。
21. NULL 是一个符号常量, 通常作为空指针值, 它的具体值为__0__ (或'\0') ____。
22. 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____
double *pv=&v;_____。
*double *pv = &v*
23. 已知语句"cout<<p;"的输出是"Hello!" , 则语句"cout<<*p;"输出的是__H____。
H
24. 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__ple____。
25. 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__ pc=(char *)pv ____。
pc = pv
26. 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__
int (*p)[100]=b____。
*int (*p) [100] = b*
27. 与结构成员访问表达式 p->name 等价的表达式是__(*p).name ____。
28. 与结构成员访问表达式(*fp).score 等价的表达式是__ fp->score ____。
29. 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__ int &ra=a____。
30. 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__6____。
31. 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__10____。
32. 已知有定义"int x, a[]={15,17,19}, *pa=a;" , 在执行"x=*pa++;"后, *pa 的值是__17____。
33. 以面向对象方法构造的系统, 其基本单位是__对象____。
34. 每个对象都是所属类的一个__实例____。
35. 对象将其大部分实现细节隐藏起来, 这种机制称为__封装____。
36. 基类和派生类的关系称为__继承____。
37. 复杂对象可以由简单对象构成, 这种现象称为__聚合____。
组合
38. 对象是对问题域中客观事物的__抽象____, 它是一组属性和在这些属性上操作的封装体____。
39. 特殊类的对象拥有其一般类的全部属性与操作, 称特殊类__继承____了一般类。
40. 如果一个派生类的基类不止一个, 则这种继承称为__多继承 (或多重继承)____。
41. 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__单继承____。
42. C++支持两种多态性: __编译____时的多态性和__运行____时的多态性。
43. 在 C++中, 编译时的多态性是通过__重载____实现的, 而运行时的多态性则是通过__虚函数____实现的。
44. 面向对象软件开发的生命周期分为三个阶段, 即分析、__设计____和__实现____。
45. 面向对象的分析包括__问题域____分析和__应用____分析两步。

46. 类定义中，既包含数据成员，也包含_函数_____成员。
47. 类中的数据成员的访问属性通常被指明为__ private (或私有)_____。
48. 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__ public(或公有)_____。
49. 对于类中定义的任何成员，其隐含访问权限为__ private _____。
50. 对于结构中定义的任何成员，其隐含访问权限为__ public(或公有)_____。
51. 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__ private _____。
52. 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_内联_____函数。
53. 若在类的定义体中只给出了一个成员函数的原型，而在类外给出完整定义时，则其函数名前必须加上__类名_____和两个冒号分隔符。
54. 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__:: (或双冒号)_____分隔符。
55. 若要把类外定义的成员函数规定为内联函数，则必须把__ inline _____关键字放到函数原型或函数头的前面。
56. 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__作用域_____。
57. 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__ AA* AA::abc()_____。
58. 成员函数的参数表在类作用域中，成员函数的返回值类型__不在_____类作用域中。
59. 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__ const _____关键字。
60. 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__ const _____关键字。
61. 在每个成员函数中，隐含的第一个参数的参数名为__ this _____。
62. 数组 a 定义为"int a[100];"，要使 p[j] 与 a[j] 等效，则指针 p 应定义为__ int *p=a; (或 int *p=&a[0];)_____。

int *p=a;

三、程序填充，对程序、函数或类中划有横线的位置，根据题意按标号把合适的内容填写到程序下面相应标号的后面

1. 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) int& ARRAY:: __ operator[](int n) //[ ]的运算符成员函数定义
{
    if(n<0 || __ (2) _ n>=s __) {cerr<<"下标越界! "; exit(1);}
    return __ (3) v[n] (或*(v+n))__;
}

(1) (2) (3)

```

2. 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:: __ ARRAY(int a[], int n)
{
    if(n<=0) {v=NULL;s=0;return;}
    s=n;
    v=__ (2) _ new int[n] __;
    for(int i=0; i<n; i++) __ (3) _ v[i]=a[i]__;
}

(1) (2) (3)

```

3. 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) int& ARRAY:: ____ operator()(int n)
{ // ()的运算符函数定义
  if(____(2) n<1 || n>s ____){cerr<<"下标越界! "; exit(1);}
  return ____ (3) v[n-1] (或*(v+n-1))____;
}

(1) (2) (3)
```

4. 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) AA::MaxA() ____
{
  int x=a[0];
  for(int i=1; i<n; i++)
    if(a[i]>x) ____ (2) x=a[i] ____;
  ____ (3) return x ____;
}

(1) (2) (3)
```

5. 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<n-1 (或 i<=n-2) ____; i++) {
        int x=a[i], k=i;
        for(j=i+1; j<n; j++)
            if(a[j]<x) ____ (2) {x=a[j]; k=j;} ____
        a[k]=a[i];
        ____ (3) _ a[i]=x ____;
    }
}

```

(1) (2) (3)

6. 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) AA::InsertA() ____ //插入排序函数
{
    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) a[j+1]=a[j] ____;
        else ____ (3) _ break ____;
    a[j+1]=x;
}
}

```

(1) (2) (3)

7. 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) _ x.SetA(a,6)____;
    int m= ____ (2) x.MaxA() ____;
    ____ (3) x.PrintA() ____;
    cout<<m<<endl;
}

```

该程序运行结果为:

23 78 46 55 62 76

78

(1) (2) (3)

2449489

8. 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) AA x ____;
    x.SetA(a,8);
    int ____ (2) m=x.MaxA() ____;
    ____ (3) x.SortA() ____;
    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)

9. 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)_ top==ARRAY_SIZE-1__) {
    cout<<"栈满!"<<endl;
    exit(1); //中止运行
}
__(2)_ top++ (或++top) __;
elem[top]=__(3) newElem __;
}

(1)          (2)          (3)

```

10. 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)_ int Stack::Pop()__ {
if(top==-1) {

```

```

cout<<"栈空!"<<endl;
exit(1); //中止运行
}
return ____(2) elem[top--]____;
}

```

```

void Stack::Print() {
while(!Empty())
cout<<____(3)_ Pop()__ <<' ';
}

```

(1) (2) (3)

四、写出程序运行结果

1. #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;
}

```

答案: 2 1 5 2

$c[0] = 2$
 $c[1] = 1$

$76 > 60$
 $c[2]++$

$c[2] = 4$
 $c[4] = 2$

2 1 5 2

2. #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;
}

```

worker cadre

}

答案: worker cadre

3. #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)<<' ';

}

答案: 5 8 11 14

4. #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;

}

答案: +-*/ abcd 1234

5. #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;

}

year = 4

month = 5

day = 3

```
}
```

答案: year:4

month:5

day:3

6.

```
#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();
```

```
}
```

答案:

7. #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();

}

```

答案：

```

8. #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;
}

```

答案：

```

9. #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;
}

```

答案：

10. #include<iostream.h>

```

class Fraction { //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
    Fraction FranAdd(const Fraction& x);
    //置分数的分子和分母分别 0 和 1
    void InitFraction() {nume=0; deno=1;}
    //置分数的分子和分母分别 n 和 d
    void InitFraction(int n, int d) {nume=n; deno=d;}
    //输出一个分数
    void FranOutput() {cout<<nume<<"/"<<deno<<endl;}
};

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

```

答案：

```

11. #include<iostream.h>
class Fraction { //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
Fracton FranAdd(const Fracton& x);
//置分数的分子和分母分别 0 和 1
void InitFracton() {nume=0; deno=1;}
//置分数的分子和分母分别 n 和 d
void InitFracton(int n, int d) {nume=n; deno=d;}
//输出一个分数
void FranOutput() {cout<<nume<< '/' <<deno<<endl;}
};
void main()
{
    Fracton a,b,c,d;
    a.InitFracton(6,15);
    b.InitFracton(3,10);
    c.InitFracton();
    c=a.FranAdd(b);
    d=c.FranAdd(a);
    cout<<"a: "; a.FranOutput();
    cout<<"b: "; b.FranOutput();
    cout<<"c: "; c.FranOutput();
    cout<<"d: "; d.FranOutput();
}

```

答案：

五、程序改错，指出错误的程序行并改正

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 NODE;                        //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->next=p;p1=p;                    //8 行
    return list;
}

```

错误行的行号为__2__和__8__。

分别改正为_ NODE *p=new NODE _____和_ p1->next=p;_____。

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=plist;                           //5 行
    plist=p;                             //6 行
return p;                                //7 行
}

```

错误行的行号为__3__和__5__。

分别改正为__ p=new NODE _____和__ p->next=plist _____。

3. 下面程序的主函数中第 7 和 8 行有错误, 请把它们改正过来。

```

#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

```

```

voi FranOutput() {cout<<nume<<'\n'<<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= a.FranAdd(b); //7 行
    c.franoutput(); //8 行
} //9 行

```

第 7 行改正为___ c=a.FranAdd(b)_____。

第 8 行改正为___ c.FranOutput()_____。

4. 假定要求下面程序的输出结果为“23/20”,其主函数中第 6, 9, 10 行有错误, 请给予改正。

```

#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<<'\n'<<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

```

```

    Frunction c;                //7 行
    c.InitFrunction();          //8 行
c=a->FranAdd(*b);              //9 行
    c.FranOutput();            //10 行
    }                          //11 行

```

错误行的行号为____、____和____。

分 别 改 正 为 ____ b->InitFrunction(3,4)____、____
c=a->FranAdd(*b)____和____ c.FranOutput()_____。

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____5____、____9____和____12____。

分别改正为____ public:____、____}____和____ int CE::GetMin()
{_____。

6. 下面程序段第 4-10 行中存在着三行语法错误，请指出错误行的行号并改正。

```

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_4____、5____和_6____。

分别改正为_ `public:`____、_ `A():c(0),a(0),b(0) {}`____
和____ `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_5_、_8_和_9_。

错误原因分别为_在函数体给常量 c 赋值_、_定义 b 多一个参数_和_定义 y 少一个参数_。

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__13__、__15__和__17__。

错误原因分别为__在函数体给常量 c 赋值__、__缺少分号__和__定义 z 多一个参数__。

六、编程

1. 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
```

答案: void p(int n)

```
{
    if(n!=0) {
        for(int i=0; i<n; i++) cout<<n;
        cout<<endl;
        p(n-1);
    }
}
```

2. 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
```

答案: void p(int n)

```
{
    if(n!=0) {
        p(n-1);
        for(int i=0; i<n; i++) cout<<n;
        cout<<endl;
    }
}
```

3. 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<n; 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。
};答案 int AA::Count(int x)
{
    int i,c=0;
    for(i=0; i<n;i++)
        if(a[i]==x) c++;
    return c;
}
```

4. 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<n; 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 的第一个元素,
                    //若查找成功则返回元素的下标, 否则返回-1。
};
```

```
};
```

答案: int AA::Search(int x)

```
{
```

```
int i;
```

```
for(i=0; i<n; i++)
```

```
    if(a[i]==x) return i;
```

```
return -1;
```

```
}
```

5. 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<n; 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。
```

```
};
```

答案: int AA::MaxMin(int& x, int& y)

```
{
```

```
int mx,my;
```

```
mx=my=a[0];
```

```
for(int i=1; i<n; i++) {
```

```
    if(a[i]>mx) mx=a[i];
```

```
    if(a[i]<my) my=a[i];
```

```
}
```

```
x=mx; y=my;
```

```
if(n>0) return 1; else return 0;
```

```
}
```

6. 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<n; 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。
};
```

```
答案: int AA::Compare(AA b)
{
    if(n!=b.n) return 0;
    for(int i=0; i<n; i++)
        if(a[i]!=b.a[i]) return 0;
    return 1;
}
```

7. 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<n; 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。
```

```
};
```

答案: int AA::CompareBig(AA b)

```
{
    int k;
    if(n>b.n) k=b.n; else k=n;
    for(int i=0; i<k; i++)
        if(a[i]>b.a[i]) return 1;
        else if(a[i]<b.a[i]) return -1;
        if(k==n && k==b.n) return 0;
    else if(k<n) return 1;
    else return -1;
}
```

C++语言程序设计练习参考解答

一、单项选择题

1. A 2. B 3. B 4. C 5. D 6. B 7. A 8. D 9. B 10. C
 11. D 12. A 13. A 14. B 15. D 16. B 17. A 18. C 19. A 20. B
 21. D 22. C 23. A 24. C 25. B 26. A 27. C 28. C 29. D 30. B
 31. D 32. C 33. A 34. B 35. A 36. C 37. C 38. B 39. A 40. C
 41. B 42. A 43. D 44. D 45. B 46. A 47. C 48. D 49. B 50. C

二、填空

1. 4*i 2. *(a+i) 3. a+i*sizeof(a[i]) 4. 第一个 修改
 5. b[7] b[2] 6. int *p=&x; 7. *p 8. int[n] int *
 9. c[3][0] c[3][2] 10. int(*)[6]
 11. *(a[i]+j) 或*(a+i+j)或*(a+i)[j] 12. int &y=x;
 13. x 14. 相等 x 15. *p 16. p[0] *p
 17. 'a' 18. char(*)[n] 19. delete p 20. delete []p
 21. 0 (或'\0') 22. double *pv=&v; 23. H 24. ple
 25. pc=(char *)pv; 26. int (*p)[100]=b; 27. (*p).name 28. fp->score
 29. int &ra=a; 30. 6 31. 10 32. 17
 33. 对象 34. 实例 35. 封装 36. 继承
 37. 聚合 38. 抽象 封装体 39. 继承 40. 多继承 (或多重继承)
 41. 单继承 42. 编译 运行 43. 重载 虚函数 44. 设计 实现
 45. 问题域 应用 46. 函数 47. private (或私有) 48. public(或公有)
 49.private 50. public(或公有) 51. private 52. 内联
 53. 类名 54. :: (或双冒号) 55. inline 56. 作用域

57. AA* AA::abc() 58. 不在 59. const 60. const
61. this 62. int *p=a; (或 int *p=&a[0];)

三、程序填充，对程序、函数或类中划有横线的位置，根据题意按标号把合适的内容填写到程序下面相应标号的后面

1. (1) int& ARRAY:: (2) n>=s (3) v[n] (或*(v+n))
2. (1) ARRAY:: (2) new int[n] (3) v[i]=a[i]
3. (1) int& ARRAY:: (2) n<1 || n>s (3) v[n-1] (或*(v+n-1))
4. (1) AA::MaxA() (2) x=a[i] (3) return x
5. (1) i<n-1 (或 i<=n-2) (2) {x=a[j]; k=j;} (3) a[i]=x
6. (1) AA::InsertA() (2) a[j+1]=a[j] (3) break
7. (1) x.SetA(a,6) (2) x.MaxA() (3) x.PrintA()
8. (1) AA x (2) m=x.MaxA() (3) x.SortA()
9. (1) top==ARRAY_SIZE-1 (2) top++ (或++top) (3) newElem
- 10.(1) int Stack::Pop() (2) elem[top--] (3) Pop()

四、写出程序运行结果

1. 2 1 5 2
2. worker cadre
3. 5 8 11 14
4. +-* / abcd 1234
5. year:4
month:5
day:3
6. abcdef 30
shenyafen 95
7. abcdef 30
abcdefxyz 50
8. 12
20
9. 4
27
10. a: 7/12
b: -3/8
c: 5/24
d: 19/24
11. a: 6/15
b: 3/10
c: 7/10
d: 11/10

五、程序改错，指出错误的程序行并改正

1. 2 8

```
NODE *p=new NODE; p1->next=p;
```

2. 3 5

```
p=new NODE; p->next=plist;
```

3. c=a.FranAdd(b); c.FranOutput()

4. b->InitFrunction(3,4); c=a->FranAdd(*b); c.FranOutput()

5. 5 9 12

```
public: } int CE::GetMin() {
```

6. 4 5 6

```
public:
```

```
A():c(0),a(0),b(0) {}
```

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

7. 5 8 9

在函数体给常量 c 赋值

定义 b 多一个参数

定义 y 少一个参数

8. 13 15 17

在函数体给常量 c 赋值 缺少分号 定义 z 多一个参数

六、编程

1.

```
void p(int n)
```

```
{
```

```
if(n!=0) {
```

```
for(int i=0; i<n; i++) cout<<n;
```

```
cout<<endl;
```

```
p(n-1);
```

```
}
```

```
}
```

2.

```
void p(int n)
```

```
{
```

```
if(n!=0) {
```

```
p(n-1);
```

```
for(int i=0; i<n; i++) cout<<n;
```

```
cout<<endl;
```

```
}
```

```
}
```

3.

```
int AA::Count(int x)
```

```
{
```

```

int i,c=0;
for(i=0; i<n;i++)
    if(a[i]==x) c++;
return c;
}

```

4.

```

int AA::Search(int x)
{
    int i;
    for(i=0; i<n;i++)
        if(a[i]==x) return i;
    return -1;
}

```

5.

```

int AA::MaxMin(int& x, int& y)
{
    int mx,my;
    mx=my=a[0];
    for(int i=1; i<n; i++) {
        if(a[i]>mx) mx=a[i];
        if(a[i]<my) my=a[i];
    }
    x=mx; y=my;
    if(n>0) return 1; else return 0;
}

```

6.

```

int AA::Compare(AA b)
{
    if(n!=b.n) return 0;
    for(int i=0; i<n; i++)
        if(a[i]!=b.a[i]) return 0;
    return 1;
}

```

7.

```

int AA::CompareBig(AA b)
{
    int k;
    if(n>b.n) k=b.n; else k=n;
    for(int i=0; i<k; i++)
        if(a[i]>b.a[i]) return 1;
}

```

```
else if(a[i]<b.a[i]) return -1;  
    if(k==n && k==b.n) return 0;  
else if(k<n) return 1;  
else return -1;  
}
```