一、单选题

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1/abc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<<\*p; B. cout<<p; **C. cout<<&p;** D. cout<<(void \*)p;

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