一、 是非题

1. C++程序中，每条语句结束时都要加一个分号“;”。
2. C++程序中，标识符的大小写字母是没有区别的。
3. 析构函数是一种函数体为空的成员函数。
4. 使用 new 创建对象时会自动调用构造函数。
5. 构造函数可以返回整数类型。
6. 可以在类的构造函数中对静态数据成员进行初始化。
7. 对象是一种由用户定义的复杂数据类型的变量。
8. while 循环语句的循环体至少执行一次。
9. const 成员函数不能更新对象的数据成员，也不能调用该类中没有用 const 修饰的成员函数。

二、 填空题

1. 在每个 C++程序中都必须包含有这样一个函数，该函数的函数名为_____。
2. 执行“cout <<43<<’-‘<<18<<’=’<<43-18<<endl;”语句后得到的输出结果为_____。
3. 由于数据隐蔽的需要，静态数据成员通常被说明为_____（公有 or 私有）的，而通过定义公有的_____来访问静态数据成员。
4. 设 px 是指向一个类动态分配的对象的指针变量，则执行“delete px;”语句时，将自动调用该类的_____。
5. 执行“typedef int ABC;”语句把 ABC 定义为_____。
6. 使用 const 语句定义一个标识符常量时，则必须对它同时进行_____。
7. 类中的每一个成员都具有一定的访问属性，其中 protected 访问属性的含义为_____。
8. 当一个成员函数被调用时，该成员函数的_____指向调用它的对象。
9. 在 C++中，函数的参数有两种传递方式：值传递和_____。
10. 描述命题"A 小于 B 或小于 C"的表达式为_____。
11. 用 new 申请某一个类的动态对象数组时，在该类中必须能够匹配到_____构造函数，否则应用程序会产生一个编译错误。
12. 设"int a=3,b=4,c=5;"，表达式"（a+b）>c&&b==c"的值是_____。
13. 在 C++类中，有一种不能定义对象的类，这样的类只能被继承，称之为_____，定义该类至少具有一个_____。

14. C++中有两种重用一个类的方式：_____、_____。
15. 在 C++中，用指针和_____作为函数参数，能够将参数值带回。
16. 在 C++中，虽然友元提供了类之间数据进行访问的一种方式，但它破坏了面向对象程序设计的_____特性。
17. 在 C++中，构造派生类对象时，总是先从_____的初始化开始的。
18. 拷贝构造函数是在用一个对象初始化另一个对象时被调用，系统缺省的拷贝构造函数的工作方法是_____。
19. 类是用户定义的类型，具有类类型的变量称作_____。
20. 在 5 个运算符"*(乘号)、<=、!=、!、&&"中，优先级由高到低的顺序是_____。
21. 在 C++中，类定义一般用 class 关键字，不过用 struct 关键字也可以定义类，它们定义类的区别在于_____。
22. 静态的成员函数没有隐含的_____，所以，它们只能访问静态的数据成员。
23. 与"(!a==0)"等价的 C++表达式是_____。
24. 若 a=6，b=4，c=2，则表达式"!(a-b)+c-1&&b+c/2"的值是_____。
25. 在下面的程序段中，语句"const int* c=&a;"和语句"int* const d=&b;"的含义分别是_____、_____。
```
const int a=78;
int b=28;
const int* c=&a;
int* const d=&b;
```
26. 用 new 申请有 10 个元素的指针数组 str，假定数组元素是指向字符型数据的指针，该 C++语句为_____。
27. 在 C++中构造一个对象时，其数据成员在构造函数中初始化。对于内嵌的对象、_____、_____数据成员需要在构造函数的成员初始化列表中初始化。
28. 在类的定义中，说明为 protected 的数据成员称为保护成员。保护数据成员具有双重作用：对于其派生类而言，_____；而对于其外部的程序而言，_____。
29. C++中有两种数据类型___和___可以使用 signed 修饰符。
30. C++中，对象保存在内存中，_____内存是自动分配和释放的，而_____内存需要用户自己申请和释放。
31. 在 C++函数中，可用 return 语句带回一个值。如果有多个返回值，可用_____、_____等带回。

三、 选择题

1. 以下不属于类的存取权限的是_____。
   A.public　B.static　C.protected　D.private
2. 在 C++中，封装是借助于_____达到的。
   A.结构体　B.类　C.数组　D.函数
3. 以下 C++中结构体的特性对类也同样适用的是_____。
   A.对象之间可以相互赋值　B.对象可以用作数组的元素
   C.对象可以用作函数参数　D.对象可以用作另一对象的成员
4. 下面的哪个保留字不能作为函数的返回类型（　）。
   A. void　B. int　C. new　D. long
5. 预处理命令在程序中都是以(　)开头的.
   A. *　B. #　C. :　D. /
6. 若想使用带默认形参值（缺省参数）的函数，默认形参值必须(　)定义。
   A．按从左到右的顺序　B．按从右到左的顺序
   C．全部　D．只给最后一个参数
7. 在下面的函数声明中，存在着语法错误的是（　）。
   A. void BC(int a, int)　B. void BD(int, int)
   C. void BE(int , int=5)　D. int BF(int x, int y)
8. 对于类中定义的成员，其隐含访问权限为(　)。
   A．public　B. protected　C．private　D. static
9. 下列(　)不是构造函数的特征.
   A.构造函数的函数名和类名相同　B.构造函数可以重载
   C.造函数可以设置缺省参数　D.构造函数必须指定返回类型
10. 重载函数在调用时选择的依据中，(　)是错误的.
   A. 参数的个数　B. 参数的类型
   C. 函数名字　D. 函数返回值的类型
11. 假定 AB 为一个类，则执行“AB a(4), b[3], * p ;”语句时，自动调用该类构造函数的次数为(　)。
   A. 3　B. 4　C. 6　D. 9
12. 关于 C++与 C 语言关系的描述中，（ ）是错误的。
   a. C 语言是 C++语言的一个子集
   b.C 语言与 C++语言是兼容的
   c.C++语言对 C 语言进行了一些改进
   d.C++语言和 C 语言都是面向对象的
13. 按照标识符的要求，（ ）符号不能组成标识符。
   a.连接符　b.下划线　c.大小写字母　d.数字字符

14. 在"int a[ ][3]={{1},{3,2},{4,5,6},{0}};"中，a[2][1]的值是（ ）。
    a.1　　b.3　　c.5　　d.4
15. 对于"int *pa[5]；"的描述中，（ ）是正确的。
    a.pa 是一个指向数组的指针，所指向的数组是 5 个 int 型元素
    b.pa 是一个指向某数组中第 5 个元素的指针，该元素是 int 型变量
    c.pa [5]表示某个元素的第 5 个元素的值
    d.pa 是一个具有 5 个元素的指针数组，每个元素是一个 int 型指针
16. 在下列表示引用的方法中，（ ）是正确的。已知：int m=10;
    a.int &x=m;　　b.int &y=10;
    c.int &z;　　d.float &t=&m;
17. 下列 for 循环的次数为（ ）。for（i=0, x=0; !x&&i<=5; i++）
    a.5　　b.6　　c.1　　d.无限
18. 对于 C/C++语言的函数，下列叙述中正确的是（ ）。
    a.函数的定义不能嵌套，但函数调用可以嵌套
    b.函数的定义可以嵌套，但函数调用不能嵌套
    c.函数的定义和调用都不能嵌套
    d.函数的定义和调用都可以嵌套
19. 在一个被调用函数中，关于 return 语句使用的描述，（ ）是错误的。
    a.被调用函数中可以不用 return 语句
    b.被调用函数中可以使用多个 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 语句可以返回多个值给调用函数
20. 在一个函数中，要求通过函数来实现一种不太复杂的功能，并且要求加快执行速度，选用（ ）。
    a.内联函数　　b.重载函数　c.递归调用 d.嵌套调用
21. 使用 fseek 函数可以实现的操作是（ ）。
    a.改变文件指针的当前位置
    b.文件的顺序读写
    c.文件的随机读写
    d.以上都不对
22. 在如下结构定义中，不正确的是（ ）。
    a.struct student {
        int no;
        char name[10];
        float score;
      };
    b.struct stud[20]{
        int no;
        char name[10];

```
        float score;
    };
c.struct student{
        int no;
        char name[10];
        float score;
    } stud[20];
d.struct{
        int no;
        char name[10];
        float score;
    }stud[100] ;
```

23. 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.strcpy( )　c.strcat( )　d.strcmp( )
24. 已知：类 A 中一个成员函数说明如下：void Set(A&a);其中，A&的含义是( )。
    a.指向类 A 的指针为 a
    b.将 a 的地址值赋给变量 Set
    c.a 是类 A 对象的引用，用来作函数 Set 函数的参数
    d.变量 A 与 a 按位与作为函数 Set 函数的参数
25. 已知：print( )函数是一个类的 const 成员函数，它无返回值，下列表示中，（ ）是正确的。
    a.void print( ) const;
    b.const void print( );
    c.void const print( );
    d.void print(const);
26. 关于虚函数的描述中，（ ）是正确的。
    a.虚函数可能是一个 static 类型的成员函数
    b.虚函数可能是一个非成员函数
    c.基类中说明了虚函数后，派生类中将其对应的函数可不必说明为虚函数
    d.派生类改写基类中定义的虚函数，可以与基类的虚函数具有不同的参数个数和类型
27. 关于 new 运算符的下列描述中，（ ）是错的。
    a.它可以用来动态创建对象和对象数组
    b.使用它创建的对象和对象数组可以使用运算符 delete 删除
    c.使用它创建对象时要调用构造函数
    d.使用它创建对象数组时必须指定初始值

四、 问答题

1. 若程序员没有定义拷贝构造函数，则编译器自动生成一个缺省的拷贝构造函数，它可能会产生什么问题？
2. 简述成员函数、全局函数和友元函数的差别。
3. 简述结构化的程序设计、面向对象的程序设计的基本思想。
4. 结构 struct 和类 class 有什么异同？
5. 在定义拷贝构造函数时，为什么通常还要定义一个重载的赋值运算符？
6. 简述局部作用域、全局作用域和类作用域的异同。
7. 虚函数是否一定要有 virtual 关键字？什么叫纯虚函数和抽象类？多态调用需要满足怎样的条件？
8. 虚析构函数有什么作用？
9. 拷贝构造函数在哪几种情况下调用？
10. 函数重载与函数覆盖有什么不同，它们与多态有什么关系？
11. C++继承是如何工作的？
12. 类与对象有什么区别？

五、 写出下面程序的运行结果。

1.

```
#include <stdio.h>
int main(){
   char a='a',b='j';
   float x;
   x=(b-a)/('F'-'A');
   printf("%d\n",(int)(3.14*x));
}
```

2.

```
#include <iostream>
using namespace std;
int main( ) {
   int i=1;
   while (i<=15){
      i++;
      if (i%3!=2) continue;
      else cout <<"i="<<i<<endl;
   }
```

```
}
```
3.
```
#include <iostream>
using namespace std;
class test {
private:
   int num;
   float fl;
public:
   test( );
   int getint( ){return num;}
   float getfloat( ){return fl;}
   ~test( );
};
test::test( ) {
   cout << "Initalizing default" << endl;
   num=0;fl=0.0;
}
test::~test( ) {
   cout << "Desdtructor is active" << endl;
}
int main( ) {
   test array[2];
   cout<<array[1].getint()<<" "<<array[1].getfloat();
}
```
4.
```
#include <iostream>
using namespace std;
class A {
public:
   A(){ cout<<"A::A() called.\n";}
   virtual ~A(){cout<<"A::~A() called.\n";}
};
class B:public A {
public:
   B(int i){ cout<<"B::B() called.\n"; buf=new char[i]; }
   virtual ~B(){ delete []buf; cout<<"B::~B() called.\n";}
private:
```

```
    char *buf;
};
void fun(A *a) {
    delete a;
}
int main() {
    A *a=new B(15);
    fun(a);
}
```

5.

```
#include <stdio.h>
int a[ ]={1,3,5,7,9};
int *p[ ]={a,a+1,a+2,a+3,a+4};
int main( ){
    printf("%d\t%d\t%d\n",a[4],*(a+2),*p[1]);
    printf("%d\t%d\t%d\n",**(p+1)+a[2],*(p+4)-*(p+0),*(a+3)%a[4]);
}
```

6.

```
#include <iostream>
#define N 100
using namespace std;
class CStack {
public:
    CStack() {top=0;cout<<"Hello ";}
    ~CStack() {cout<<"Bye";}
    void push(int i);
    int pop();
private:
    int stack[N];
    int top;
};
void CStack::push(int i) {
    if (top==N) cout<<"Overflow";
    else stack[top++]=i;
}
int CStack::pop( ) {
```

```
    if (top==0){ cout<<"Underflow"; return 0;}
    else return stack[--top];
}
int main() {
    CStack *ptr=new CStack;
    ptr->push(10);
    ptr->push(50);
    cout <<ptr->pop( )<<" ";
    cout << "OK!"<<endl;
}
```

7.

```
#include <iostream>
using namespace std;
class B {
public:
    B(){}
    B(int i){ b=i;}
    virtual void virfun(){ cout<<"B::virfun() called.\n";}
private:
    int b;
};
class D:public B {
public:
    D(){}
    D(int i,int j):B(i){d=j;}
private:
    int d;
    void virfun(){ cout<<"D::virfun() called.\n";}
};
void fun(B *obj) {
    obj->virfun();
}
int main() {
    D *pd=new D;
    fun(pd);
}
```

8.

```
#include<iostream>
int main() {
   using namespace std;
   int num = 500;
   int &ref = num;
   cout << ref;
   ref = ref + 100;
   cout << " " << num; //输出空格和num的数值
   num = num + 50;
   cout << " " << ref; //输出空格和ref的数值
}
```

9.

```
#include <iostream>
using namespace std;
class B1 {
public:
   B1(int i) {cout<<"constructing B1 "<<i<<endl;}
   ~B1() {cout<<"destructing B1 "<<endl;}
};
class B2 {
public:
   B2(int j) {cout<<"constructing B2 "<<j<<endl;}
   ~B2() {cout<<"destructing B2 "<<endl;}
};
class B3 {
public:
   B3(){cout<<"constructing B3 *"<<endl;}
   ~B3() {cout<<"destructing B3 "<<endl;}
};
class C: public B2, public B1, public B3{
public:
   C(int a, int b, int c, int d)
   : B1(a), memberB2(d), memberB1(c), B2(b) {}
private:
   B1 memberB1;
   B2 memberB2;
   B3 memberB3;
```

```
};
int main() {
   C obj(1,2,3,4);
}
```

10.

```
#include <iostream>
using namaespace std;
class Clock {
public:
   void init(int, int, int = 0);  // 初始化
   void pass();          // 走过1秒钟
   void display();             // 显示时间
private:
   int   hour, minute, second; // 时间的内部表示
};
void Clock::init(int h0, int m0, int s0) {
   hour = h0; minute = m0; second = s0;
}
void Clock::pass() {
   second++;
   if (second == 60) {second = 0; minute++;}
   if (minute == 60) { minute = 0; hour++;}
   if (hour == 24) hour = 0;
}
void Clock::display() {
   cout << hour << ":" << minute << ":" << second << endl;
}
int main() {
   Clock myclock;
   myclock.init(7, 59, 58);
   myclock.pass();
   myclock.display();              //写出输出结果1，原因?
   myclock.pass();
   myclock.display();              //写出输出结果2，原因?
   myclock.init(6, 30);
   myclock.display();              //写出输出结果3，原因?
```

```
    for (int i = 0; i < 100; i++) myclock.pass();
    myclock.display();              //写出输出结果4，原因？
}
```

11.

```
#include <iostream>
using namespace std;
class B {
public:
    int i;
    void printi() {cout << i << " inside B\n"; }
};
class D1: public B {
public:
    void printi() {cout << i << " inside D1\n"; }
};
class D2: private B {
public:
    D2() {B::i = 4; }
    int i;
    void printi() {cout<<i<<"inside D2,B::i is"<<B::i<<"\n";}
};
int main() {
    B  b;
    B  *pb = &b;
    D1  f, *pf;
    D2  h, *ph;
    h.i = 1 + (f.i = 1 + (b.i = 1));
    pb->printi();
    pf = &f;
    pf->printi();
    ph = &h;
    ph->printi();
    pb->printi();
}
```

12.

```
class test {
```

```
private:
   int num;
public:
   test();
   int getint() { return num; };
   ~test();
};
test::test() {
   num = 0;
}
test::~test() {
   cout << "Destructor is active" << endl;
}
int main() {
   test x[3];
   cout << "Exiting main" << endl;
}
```

13.分析下面程序的输出结果，并回答其中的 4 个问题：

```
#include <iostream>
#include <string.h>
using namespace std;
struct Worker{
   char name[15];   // 姓名
   int age;         // 年龄
   float pay;       // 工资
};
int main() {
   Worker  x;   //问题1：写出本行起什么作用？
   char *t="liuting";
   int d = 38;
   float f = 493;
   strcpy(x.name, t); //问题2：写出本行起什么作用？
   x.age=d; x.pay=f; //问题3：为什么这2行可以执行，有什么作用？
   cout <<x.name<<' '<<x.age<<' '<<x.pay<<endl; //问题4：写出输出结
果
}
```

六、 改错题。找出下面程序中的语法错误，如果可能予以纠正

1. 程序功能是倒序输出各给定的字符串。

```
#include <stdio.h>
int main() {
   char str[5][] = {"First", "Second", "Third", "Forth", "Fifth" };
   char *cp[ ]={str[4],str[3],str[2],str[1],str[0]};
   int i;
   while(i<=5) {
      printf("%c ",*(cp+i));
      i++;
   }
}
```

2. 程序功能是将各个平方根值放入数组中。

```
#include <stdio.h>
int main(){
   int max,a,i;
   scanf("%d%d",max,a);
   double x[max];
   for (i=0;i<max;i++)
      x[i]=sqrt(a*i);
}
```

3. 程序功能是将某年某月的几号转换成这一年的第多少天。

```
#include <stdio.h>
struct date {int y; int m; int d;}
int daytab[2][ ]={{0,31,28,31,30,31,30,31,31,30,31,30,31},
{0,31,29,31,30,31,30,31,31,30,31,30,31}};
int days (struct date *pd);
int main( ) {
   int n;
   struct date d;
   printf("Input:Year=? Mouth=? Day=? \n");
   scanf("%d%d%d",y,m,d);
   n=days(&d);
   printf("N=%d days\n",n);
}
int days (struct date *pd) {
```

```
    int i,day,leap;
    day=pd.d;
    leap=pd.y%4==0&&pd.y%100!=0 || pd.y%400==0;
    for(i=1;i<pd->m;++i)
        day+=daytab[leap][i];
    return(day);
}
```

4. 程序功能是打印 Object 类型变量的分量 a。

```
#include <iostream>
struct Object{int a;int b;};
int main(){
    Object& MyFunction(int a=20,int b);
    Object& rMyObj = MyFunction(,5);
    cout << "rMyObj.a = " << rMyObj.a << endl;
    delete &rMyObj;
}
Object& MyFunction(int a=20,int b){
    Object* o = new Object;
    o->a = a;
    o->b = b;
    return o;
}
```

5. 下面的函数将浮点型参数 para 的值赋给一个局部指针变量 pFloat，然后输出它的值。

```
#include <iostream>
void func(float para) {
    float * pFloat;
    *pFloat = para;
    cout <<*pFloat;
}
```

6. 下列程序片段对二维数组的每个元素赋值

```
unsigned short SomeArray[5][4];
for (int i = 1; i<=5; i++)
    for (int j = 1; j<=4; j++)
        SomeArray[i][j] = i + j;
```

7. 下列程序中包含三个错误

```
class  MyClass{
public:
   MyClass(int ini) { member = ini; }
   void SetMember(int m) { member = m;}
   int GetMember() const { return member; }
private:
   int member;
};

int main() {
   MyClass obj1;
   MyClass obj2(3);
   obj1.member = 5;
   MyClass.SetMember(10);
}
```

8. 下列有关缺省参数的程序写法是否都合法，不合法的请你改正：

(1)

```
int add(int a = 1, int b = 2);
int main(){cout << add(3);  }
int add(int a, int b){return a + b; }
```

(2)

```
int add(int a, int b);
int main(){cout << add(3);  }
int add(int a = 1, int b = 2){ return a + b;    }
```

(3)

```
int add(int a = 1, int b = 2);
int main(){cout << add(3);  }
int add(int a = 1, int b = 2){ return a + b;    }
```

(4)

```
int add(int a = 1, int b = 2){ return a + b;    }
int main(){cout << add(3);  }
```

七、 程序填空题

1. 下面的程序可以统计命令行第一个参数中出现的字母个数，请填充下面空白，完成程序。

```
#include <stdio.h>
#include <ctype.h>
int main(int argc, ________argv[];) {
   char *str;
   int count=0;
   if(argc<2)exit(1);
   str=__________;
   while(*str)
      if(isalpha(_________)) count++;
   printf("\n字母个数：%d\n",count);
}
```

提示：int isalpha(int ch)函数功能是检查ch是否是字母

2. 完成顺序查找函数 f_seq( )。其过程是：从表头开始，根据给定的模式，逐项与表中元素比较。如果找到所需元素，则查找成功，并打印出它在表中的顺序号。如果查找整个表仍未找到所需对象，则查找失败

```
#include <stdio.h>
#include <string.h>
//list 指针数组，指向字符串 object 模式串 len 表的长度
void f_seq(char *list[],char *object,int len) {
   char **p;
   p=list;
   while (__________________)
      if (strcmp(*p,object)==0)
         break;
      else _________________;
      if (p<list+len)
         printf( "Success! **% d\n",p-list);
      else printf("Unsuccess!\n");
}
int strcmp(char *s,char *t) {
   for (;*s==*t; s++,t++)
      if (*s=='\0') return(0);
   return ___________________;
```

```
}
```

3. 函数 binary（）实现折半查找，即查寻给定的单词 word 是否在关键字表 tab 中（关键字按字典顺序排列），折半查找每次把 word 与 tab 表中相应部分的位于中间位置的关键字进行比较，最终结果：或者与某个关键字相同，或者与所有关键字都不相同。

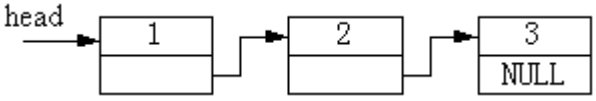
```
#include <string.h>
struct key{char *keyword; int count;};
//word: a searching word
//tab: keyword table
//n: the sum of keywords
struct key *binary(char *word, struct key tab[],int n) {
   int cond;
   struct key *low=tab;
   struct key *high=&tab[n-1];
   struct key *mid;
   while(low<=high){
      mid=___________;
      if((cond=strcmp(word,mid->keyword))<0)
         high=mid-1;
      else if (cond>0)
         __________;
      else return mid;
   }
   return NULL;
}
```

4. 下面程序实现汉诺塔游戏。规则是：三个立柱（分别为 A、B、C），开始 A 上串有 n 个（用户输入值）大小不等的圆盘，大的在下，小的在上。要求借助于 B 把它们从 A 移到 C。每次只能移一个盘，而且三个柱上的盘总是大的在下，小的在上

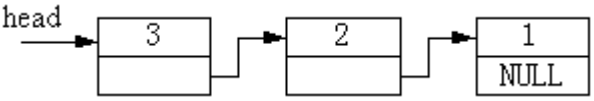
```
#include <stdio.h>
int i=0;
void movetower(int m,char from, char to,char usg);
void movedisk(char source,char destination);
int main( ){
   int n;
```

```
    for (;;){
        printf("input the number of disks of hanoi tower:");
        scanf("%d",&n);
        if (n==0)___________;
        printf("\n\n");
        printf("The moving step is as below:\n");
        movetower(n,'A','C','B');
        printf("\tTotal:%d\n",i);
    }
}
void movetower (int m,char from, char to,char usg) {
    if (__________){
        movetower(m-1,from,usg,to);
        ____________;
        movetower(m-1,____________);
    } else
        movedisk(from,to);
}
void movedisk(char source,char destination) {
    i++;
    printf("%c------->%c\n",source,destination);
}
```

5. 完成使链表逆置函数 reverse，若有链表：



逆置后则变为：



链表结点的结构如下：

```
struct node {
    int num;
    struct node *next;
}
//head 链表头结点
struct node* reverse(struct node *head) {
    struct node *p,*temp1,*temp2;
    if(head==NULL_____________) return head;
    p=head->next;head->next=NULL;
```

```
    while(__________) {
        temp1=head;
        _________;
        temp2=p;
        p=p->next;
        _________________;
    }//Match while statenment
    return head; //返回逆置后的链表的头结点
}
```

八、 编程题

1. 定义一个存放字符的栈类 Stack（包括类的实现)。栈的基本操作为 Push（）和 Pop（)。
2. 完成下面的函数，对有 n 个元素的数组 a，使数组元素按逆序排列。
   void inverse(int *a, int n) {
   }
3. 下面的函数统计子字符串 substr 在字符串 str 中出现的次数，如果 substr 在 str 中不出现，则返回值 0。请完成该函数。
   int str_count(char *substr, char *str){
   }
4. 下列 shape 类是一个表示形状的抽象类，area( )为求图形面积的函数，total( )则是一个通用的用以求不同形状的图形面积总和的函数。请从 shape 类派生三角形类(triangle)、矩形类（rectangle)，并给出具体的求面积函数。

```
class shape{
public:
    virtual float area( )=0;
};

float total(shape *s[ ],int n) {
    float sum=0.0;
    for(int i=0;i<n;i++)
        sum+=s[i]->area( );
    return sum;
}
```

5. 某一个班级有若干同学（假设有 3 人)，每个同学有语文，数学二门功课成绩和总分。 程序的结构如下所示，完成该程序并在上机运行测试。

```
#include<iostream>
using namespace std;
typedef struct studInfo{
   int chinese;
   int math;
   int totalScore;
}STUDINFO;
int main() {
   STUDINFO  stud[3];
   //…
}
```

(1). 编写一个函数输入这 3 个同学的各门功课的成绩；

(2). 编写一个函数求各同学的总分；

(3). 编写一个函数求各同学的名次，并按他们名次的由高到低顺序依次输出这 3 个同学的语文、数学成绩。

6. 下面是一个类的测试程序，设计出能使用如下测试程序的类

```
int main() {
   Test a;
   a.init(68, 55);
   a.print();
}
```

测试结果：输出 68-55=13。

```
class Test {
public:
    void init(int a0, int b0) { a = a0; b = b0; }
    void print() { cout << a <<“-” << b<< “=” <<a - b; }
private:
    int a, b;
};
```

7. 设计并测试一个名为 Circle 的矩形类：

```
class Circle {
public:
   Circle(float r=0)  ;
   Circle(const Circle & a)  ;
   float Area() ;
private:
```

```
    float radius;
};
int main(){
    Circle  A;
    Circle  B(4);
    Circle  C(B);
    cout<<A.Area()<<endl;
    cout<<B.Area()<<endl;
    cout<<C.Area()<<endl;
}
Circle::Circle(float r) :radius(r){
}
Circle::Circle(const Circle &a) : radius(a.r){
}
float Circle::Area() {
      return 3.14 * radius * radius;
}
```

8. 设计并测试一个名为 Cat 类，其定义如下：

```
class Cat {
public:
    Cat();
    Cat(Cat& a )    ;
    static int  GetHowMany() ;
private:
    static int HowManyCats;
};
int main() {
    Cat A;
    cout<<"猫的个数为：  "<<A.GetHowMany()<<endl;
    Cat B(A);
    cout<<"猫的个数为：  "<<A.GetHowMany()<<endl;
    Cat C;
    cout<<"猫的个数为：  "<<A.GetHowMany()<<endl;
}
Cat::Cat() { HowManyCats++; }
Cat::Cat(Cat &a) { HowManyCats++;}
int Cat::GetHowMany(){ return HowManyCats;}
int Cat::HowManyCats = 0;
```

9. 设计并测试一个名为 Rectangle 的矩形类，其定义如下：

```
class Rectangle {
public:
   Rectangle(float length=3,float width=5);
   Rectangle(Rectangle & a);
   float RectangleFerence(); //计算矩形周长
   float Area() ;            //计算矩形面积
   int Compare(Rectangle & a); //比较2矩形的大小，如果大于返回1，小于返回-1，等于返回0
private:
   float  m_length,  m_width;
};
int main() {
   Rectangle  A;
   Rectangle  B(4,5);
   Rectangle  C(B);
   cout<<"矩形A的面积: "<<A.Area()<<endl;
   cout<<"矩形B的面积: "<<B.Area()<<endl;
   cout<<"矩形C的面积: "<<C.Area()<<endl;
   switch(A.Compare(C)) {
   case 1: cout<<"A的面积大于C的面积！！！"; break;
   case 0: cout<<"A的面积等于C的面积！！！"; break;
   case -1: cout<<"A的面积小于C的面积！！！"; break;
   default:break;
   }
}
```

10. 根据下面的要求逐步写出正确的 C++语句，注意：各个步骤之间是有先后顺序的。
    1) 定义两个浮点型变量 f1,f2；
    2) 定义一个指向浮点数变量的指针 pFloat，将该指针初始化为指向 f1；
    3) 将指针 pFloat 改为指向变量 f2；
    4) 通过 pFloat 指针来间接地改变变量 f2 的值为 20.3；
    5) 申请三个连续的浮点数空间，并将申请到空间的首地址赋值给 pFloat;
    6) 用 cout 输出所申请到的首地址值；
    7) 给第二个地址空间赋值 11.3；
    8) 用 cout 输出第二个地址空间的地址；

9) 释放所申请到的三个浮点数空间；
10)将指针 pFloat 设置为不指向任何地址空间；

```
float f1, f2;
float *pFloat = &f1;
pFloat = &f2;
*pFloat = 20.3;
pFloat = new float[3];
cout << pFloat;
pFloat[1] = 11.3;
delete[] pFloat;
pFloat = NULL;
```

11. 编写类 String 的构造函数、析构函数并写出主函数对其进行测试。
已知类 String 的原型为：

```
class String {
public:
   String(const char *str = NULL);   // 普通构造函数
   String(const String &other);       // 拷贝构造函数
   ~ String();              // 析构函数
private:
   char *m_data;                 // 用于保存字符串
};
```

请编写 String 的上述 3 个函数及测试主函数。

```
String::String(const char *str) {
    m_data = new char [strlen(str)+1];
    strcpy(m_data, str);
}
String::String(const String &other) {
    m_data = new char[strlen(other.m_data)+1];
    strcpy(m_data, other.m_data);
}
String::~String() {
      delete[] m_data;
}
```

12. 写一个复数类 Complex，使得如下主程序可以产生后面的输出。

```
#include “Complex.h”
#include <iostream>
int main() {
```

```
    using namespace std;
    Complex c1(4.5),   c2(2,3),   c3;
    c1.add(c2);
    c3.add(c1);
    cout<<"c1="<<c1.real() <<"+"<<c1.imag() <<"i"<<endl;
    cout<<"c2="<<c2.real() <<"+"<<c2.imag() <<"i"<<endl;
    cout<<"c3="<<c3.real() <<"+"<<c3.imag() <<"i"<<endl;
}
```

输出：

c1 = 6.5 + 3i
c2 = 2 + 3i
c3 = 6.5 + 3i

13. 认真读懂课本 P79 介绍的栈（Stack）这个类，请上机按下面的要求完成程序，并回答下列问题。

写头文件 stack.h，完成类 Stack 的声明；
另外写一个源文件 stack.cpp，完成类 Stack 的定义；
编写主文件 main.cpp，完成类 Stack 的测试；
回答为什么将所有的 Stack 成员函数都设计成 inline 类型；
为 Stack 类设计一个 gettop 成员函数，用来返回 Stack 的顶端元素，但不能将该元素移出 Stack。

14. 下面列出了由三个文件 main.cpp、three.h 和 three.cpp 组成的一个程序。文件 main.cpp 中实现了主函数；文件 three.h 中定义了类 CThree；文件 three.cpp 中实现了类的成员函数。类 CThree 储存三个成员变量，成员函数 Min 传回其中的最小值，成员函数 Max 传回其中的最大值。请完成类中的三个成员函数。

```
    //文件 main.cpp
?
    //文件  three.cpp
?
    //文件  three.h
?
```

15.

(1)设计一个基类叫“师院人(person)”，有姓名、出生年月、性别属性（即数据成员），并有一个“师院人卡片”输出函数 PrintOut()。
(2)在“师院人”类下派生出“教职工(staff)”、“一般工人(worker)　”、“学生(student)”子类，在“教职工”下再派生出“教师　(teacher)”和“行政人员

(official)”子类，各子类再添加自己的属性，如教工的职称，学生的学号。在这些子类中分别给出各自的“师院人卡片”以具体实现。
(3)输入 10 个“师院人”(运行时逐个输入也可，在 main()程序直接填好也可)，教职工、学生各色人员都有，再依次输出这 10 张“师院人卡片”。
实验内容或步骤必须包含类的派生关系图、主程序 main 代码、基类及起码一个派生类的构造函数和 PrintOut()代码。
(要求：做完练习后一定要仔细分析程序运行的结果及原因)

16. 已有如下 Vehicle 类，请从此类派生出 Bicycle 类，并加 int Height 成员；再从 Bicycle 类派生出 Motorcycle 类，并加 int SeatNum 成员。并以下面主函数测试之。

```
class Vehicle {
public:
   Vehicle(int m=0,int w=0){MaxSpeed=m; Weight=w;}
   ~Vehicle(){}
   void Print() {cout<<MaxSpeed<<"   "<<Weight<<endl; }
private:
   int MaxSpeed;
   int Weight;
};
int main() {
   Motorcycle a(100,20,30,200);
   a.Print ();       //要求在此输出结果100，20，30，200
}
```

17.给定基类：

```
class Base{
public:
   virtual void Iam() {cout<<"Base "<<endl;}
};
```

(1)请从 Base 类派生两个新类 NewClass1 和 NewClass2 ，每个类定义 Iam()来打印出自己类的名字；
(2) 编写一个以 Base*为参数的函数 fun，函数体内实现通过该指针调用 Iam()；
(3)在主函数中分别创建出这些类的对象，用这些对象分别调用 Iam()；
(4)以这些类对象的地址分别调用 fun 函数。
程序运行的结果如下：Base，NewClass1，NewClass2
Base，NewClass1，NewClass2

18. 编写字符串类 String(注意类名中 S 为大写)。

```
#include <iostream>
#include <string.h>
class String {
public:
   String(const char *str = NULL);      // 普通构造函数
   String(const String &other);          // 拷贝构造函数
   ~String();                  // 析构函数
   String & operate = (const String &other); // 赋值函数
   char* c_str() { return m_data; }
private:
   char *m_data;               // 用于保存字符串
};
inline ostream & operator<< (ostream & os,String & s) {
   return os<<s.c_str();
}
int main() {
   String   a1;
   cout<<a1<<endl<<endl;
   String   a2("YWB");
   cout<<a2<<endl<<endl;
   String a3(a2);
   cout<<a3<<endl<<endl;
   a1 = a3;
   cout<<a1<<endl<<endl;
}
```

(要求：做完练习后一定要仔细分析程序运行的结果及原因)

19. 参考课本 P219 类 Complex 实现重载等号（ = = ）、不等号（ ! = ）、求负（ - ）和输出（ >> ）操作符。

（复数的判等操作定义为：复数 Z1 = a+b i 等于 Z2 = c+d i，当且仅当 a 等于 c 且 b 等于 d。复数求负操作定义为：如果 Z = a+b i 是一个复数则 -Z = -a-b i。）

20. 声明一个 Shape 抽象类，在此基础上派生出 Rectangle 和 Circle 类，二者都有 GetArea ( ) 函数计算对象的面积，按下面的要求完成程序的编制工作。

(1) Rectangle 类有 m_Width, m_Height 属性；

(2) Circle 类有 m_Radius 属性；

(3) 分别编写 Rectangle 类和 Circle 类带默认参数的构造函数；
(4) 编写主函数对 Rectangle、Circle 类进行测试，使它们以统一的操作界面输出面积；（提示：利用 for 循环进行输出）
(5) 回答将 Shape 类声明为抽象类的方法及意义。

九、 综合题

1.

```
#include <iostream>
using namespace std;
class A {
public:
   virtual void func(int data){cout<<"class A:"<<data<<endl;}
   void func(char *str){ cout<<"class A:"<<str<<endl; }
};
class B: public A {
public:
   void func() {cout<<"function in B without parameter! \n";}
   void func(int data) { cout<<"class B:"<<data<<endl; }
   void func(char *str){ cout<<"class B:"<<str<<endl;}
};
int main() {
   B b;
   A *pA=&b;
   pA->func(1);
   pA->func("haha");
}
```

问题 1：在下面写出程序的运行结果：
问题 2： 如下句所示，在函数 main()中通过 pA 调用类 B 中定义的参数表为空的函数 func()：pA->func();是否正确为什么？
问题 3：如果要记录已创建的 A 类的实例（对象）的个数，我们可以借助于类的静态成员。修改上面类 A 的定义，使得它包含一个私有的静态成员 object_count，记录属于该类的对象的个数，然后为类 A 增加必要的成员函数，使得下面的程序：

```
int main() {
    A *pA=new A[3];
```

```
    cout << ”There are” << pA->GetObjectCount() << ” objects” << endl;
    delete []pA;
    cout << ”There are” << A::GetObjectCount() << ” objects” << endl;
}
```

得到的输出为：

There are 3 objects

There are 0 objects

1）在下面写出类 A 的定义（将所有的函数成员实现写在类定义体中）：

2）在下面写出初始化类的静态成员 object_count 的语句

2. 请完成下面 2 个运算符重载函数，并对程序最后一行分析错误原因并改正。

给定整数类：

```
class Integer {
public:
   void Set ( int ii=0 ) { i=ii; }
   Integer operator + ( int c );
   Integer operator + (Integer & c);
   Integer& operator += (Integer & c);
   void Display() {cout<<i<<endl;}
private:
   int i;
};
```

请完成上面 3 个运算符重载函数，并以下面的主函数进行测试。

```
int main() {
   Integer A, B, C;
   A.Set(20);
   C.Set(10);
   B=A+4;
   B.Display();
   B=A+C;
   B+=C;
   B.Display();
   B = 4 + A; // 此行代码有误，请分析错误原因并改正
}
```

3. 下面的程序定义了一个简单的 SmallInt 类，用来表示从-128 到 127 之间的整数。类的唯一的数据成员 val 存放一个-128 到 127（包含-128 和 127 这两个数）之间的整数，为了方便，类 SmallInt 还重载了一些运算符。阅读 SmallInt 的定义，回答题目后面的问题。

```
class SmallInt{
public:
   SmallInt(int i=0); {
      while(i>127) i-=256;
      while(i<-128) i+=256;
      val=i;
   }
   friend ostream&operator<<(ostream&os, const SmallInt&si) {
      os<<(int)si.val; return os;
   };
   friend istream &operator>>(istream &is, SmallInt &si) {
      int tmp; is>>tmp;
      si=SmallInt(tmp); return is;
   }
   SmallInt operator+(const SmallInt &si){return
SmallInt(val+si.val);}
   SmallInt operator-(const SmallInt &si){return
SmallInt(val-si.val);}
   SmallInt operator*(const SmallInt &si){return
SmallInt(val*si.val);}
   SmallInt operator/(const SmallInt &si){return
SmallInt(val/si.val);}
   bool operator==(const SmallInt &si){return (val==si.val);}
private:
   char val;
};
```

问题 1：上面的类定义中，重载的插入运算符和抽取运算符被定义为类的友元函数，能不能将这两个运算符定义为类的成员函数？如果能，写出函数原型，如果不能，说明理由。

问题 2：为类 SmallInt 增加一个重载的运算符'+='，函数原型如下：

```
class SmallInt{
public:
SmallInt &operator +=(const SmallInt &si);
//其它函数……
```

```
private:
    char val;
};
```

该函数将返回对当前对象的引用。如：SmallInt si1(20),si2(13); 则表达式：si1+=si2 将返回对象 si1 的引用，并且 si1 中的的值是 si1.val 和 si2.val 的值之和（但必须正规化为在-128 至 127 之间）。

在下面写出重载的运算符+=的实现：

4. 读懂程序，请对所提的问题做出准确解答。

```
1)  #include <iostream.h>
2)  int CircleArea()
3)  {
4)      double *pd=new double;
5)      if(!pd)
6)      {
7)          cout<<"Error Memory Allocation!"<<endl;
8)          return 1;
9)      }
10)     double &rd=*pd;
11)     cout<<"The radius is: ";
12)     cin>>rd;
13)     cout<<"The Area of Circle is: "<<rd*rd*3.14<<endl;
14)     delete  &rd;
15)     return 0;
16) }
17) int main()
18) {
19)     if(CircleArea())
20)         cout<<"The programn failed!"<<endl;
21)     else
22)         cout<<"The programn successed!"<<endl;
23) }
```

请写出下列问题答案：

第 4 行起何作用？

第 5-9 行可否省去？并说明原因。

第 14 行起何作用？

此程序功能是什么？

解释 14 行中&符号的意义？

5. 下面的程序定义了三个类，Base 是图形对象基类，Point 表示屏幕上的一

个点，ColorPoint 表示带颜色的点，基类中利用 posx 和 posy 记录图形对象的位置。先阅读三个类的定义，然后回答后面的问题。

```
#include <iostream>
class Base{
public:
   Base():posx(0),posy(0){ };
   Base(int px,int py):posx(px),posy(py){};
   virtual void draw();
protected:
   int posx,posy;    //图形对象的位置
};
class Point:public Base{
public:
   Point(int px,int py):Base(px,py){}
   void draw(){std::cout<<"Point::draw";}
};
class ColorPoint:public Point{
public:
   ColorPoint(int px,int py):Point(px,py){color=0;}
   ColorPoint(int px,int py,int c):Point(px,py),color(c){}
   void draw(){Point::draw();}
protected:
   int color;
};
```

问题 1：判断下面语句的正误并说明原因；

```
int main() {
      Base b; Point p1; Point p2(3, 4);
      ColorPoint cp1;
      ColorPoint cp2(5, 6);
      ColorPoint cp3(7, 8, 1);
}
```

问题 2：将类 ColorPoint 的两个构造函数合并为一个构造函数，写出该函数的实现。

问题 3：下面的程序段

```
Base * p;
p = new ColorPoint(5,6);
p->draw()
```

运行后将调用哪个函数，运行结果是多少？