一．单选题 （每小题 3 分，共 30 分，第 11、15、17 小题为加分题）

1—5	D	C	D	D	B
6—10	A	C	A	B	B
11—15	A	A	C	B	C
16—18	D	A	D		

二．填空题（每小题 5 分，共 25 分）

19 题

```
void swap (int& a, int& b)
{     int tmp;
      tmp = a;
      a = b;
      b = tmp;
}
void swap (int* a, int* b)
{     int tmp;
      tmp = *a;
      *a = *b;
      *b = tmp;
}
```

20 题

```
Point& Point::operator ++()   //前置自增操作符
{
      X++;
      Y++;
      return *this;
}
Point Point::operator ++(int)   //后置自增操作符
{
      Point temp=*this;
      ++(*this);
      return temp;
}
```

21 题

（1） int *p; 指向：整型变量

（2） int *pβ]; 指向：int*型指针所构成的数组的首元素

（3） int (*p)[3]; 指向：整型变量构成的数组

（4） int (*p)(); 指向：函数返回值为整型的函数

22 題

// Section 1
B's default constructor
A's default constructor
C's default constructor
// Section 2
B's default constructor
A's default constructor
C's copy constructor
func1
C's destructor
A's destructor
B's destructor
// Section 3
func2
// Section 4
C's destructor
A's destructor
B's destructor

23 題

// Section 1
A's default constructor
A's f
A's default constructor
A's f
B's default constructor
// Section 2
A's copy constructor
A's f
A's f
A's g
A's f
A's g
A's default destructor
// Section 3
B's f
A's g
B's f
A's g

// Section 4

A's default destructor

A's default destructor

三. 简答题 （每小题 5 分，共 20 分）

24 题

指针类型主要用于参数传递和对动态变量的访问。在 C++中，指针类型还用于访问数组元素，以提高访问效率。

引用类型与指针类型都可以实现通过一个变量访问另一个变量，但访问的语法形式不同：引用是采用直接访问形式，指针则采用间接访问形式。在作为函数参数类型时，引用类型参数的实参是一个变量，而指针类型参数的实参是一个变量的地址。

除了在定义时指定的被引用变量外，引用类型变量不能再引用其他变量；而指针变量定义后可以指向其他同类型的变量。因此，引用类型比指针类型要安全。引用类型的间接访问对使用者而言是透明的。

25 题

因为对类成员访问权限的控制，是通过设置成员的访问控制属性实现的。所以访问控制属性有以下三种：public, private 和 protected。

1. 公有类型成员用 public 关键字声明，任何一个来自类外部的访问都必须通过这种类型的成员来访问（“对象.公有成员”）。公有类型声明了类的外部接口。
2. 私有类型成员用 private 声明（若私有类型成员紧接着类名称，可省略关键字），私有类型的成员只允许本类的成员函数来访问，而类外部的任何访问都是非法的。这样完成了私有成员的隐蔽。
3. 在不考虑继承的情况下， 保护类型（protected）的性质和私有类型的性质一致。即保护类型和私有类型的性质相似，其差别在于继承过程中对产生的新类影响不同。

26 题

拷贝构造函数用于在创建对象时用另一个同类的对象对其初始化。一般来说，有三种情况将调拷贝构造函数：

1） 定义对象时
2） 把对象作为值参数传给函数时
3） 把对象作为返回值时

27 题

public 继承方式使得基类的 public 成员可以被派生类的对象访问，它可以实现类之间的子类型关系；protected 继承使得基类的 public 成员不能被派生类的对象访问，但可以被派生类的派生类访问；private 继承使得基类的 public 成员既不能被派生类的对象访问，也不能被派生类的派生类访问。protected 和 private 继承主要用于实现上的继承，即纯粹为了代码复用。

28 题 （每个派生类和 main 函数各 5 分，共 20 分）

```
class Shape {
   public:
      virtual double area() const=0;
};
class Square: public Shape {
   public:
      Square(double s): side(s){}
      double area() const{ return side*side; }
   private:
      double side;
};
class Trapezoid: public Shape {
   public:
      Trapezoid(double i, double j, double k): a(i),b(j),h(k) {}
      double area() const{ return ((a+b)*h/2); }
   private:
      double a,b,h;
};
class Triangle: public Shape {
   public:
      Triangle(double i,double j):w(i),h(j) {}
      double area() const{ return(w*h/2); }
   private:
      double w,h;
};
void main()
{
      Square se(5);   Trapezoid td(2,5,4);   Triangle te(5,8);
      Shape *p[3];
      p[0]=&se;    p[1]=&td;   p[2]=&te;
      double da=0;
      for(int i=0; i<3; i++) { da+=p[i]->area(); }
      cout << "总面积是：" << da << endl;
 }
```