一、F F F T T F F T T F
二、C D B A D B A C
三、1. 一句一分，后两句顺序错的直接扣 2 分

```
int & ra = a;
int * pa = &a;
cin >> a;
cout << "a == " << a;
```

2. 一个 1 分，分 C、D 开头的两组，各组都从第一个错的开始扣分

```
CA                  DA
CAI                 DB
CB                  DA
CAI                 DB
CAI                 DA
CB                  DA
```

3. 红色是出错语句，一个两分,错改位置的倒扣分，直到本题 0 分为止

```
#include <iostream>
using namespace std;
struct a_ST
{
public:
    a_ST( int aVal = 1000 ):_st(aVal){}
    a_ST(const a_ST & aRef)
    {
        this->_st = aRef._st;
    }
    void display() const
    {
        cout << _st << endl;
        cout << KST << endl;
    }
private:
    int _st;
    static int KST;
};
int a_ST::KST = 0;  //添加
int main()
{
    a_ST st;
    st.display();
    a_ST *sp = new a_ST;
    delete sp;  //添加
    return 0;
}
```

四、参考答案 1. 头文件 1 分，析构函数 1 分，其余成员函数 2 分

```
#include <cmath>
#include "circle.h"                                    // 共 1 分
```

```
Circle::Circle( double aX, double aY, double aR )
    : m_iX(aX), m_iY(aY), m_iRadius(aR) {}            // 2分
Circle::~Circle ( ) {}                                 // 1分
void  Circle::setPos ( double aX, double aY )          // 2分
{
    m_iX = aX,   m_iY = aY;
}
void  Circle::getPos ( double & rX, double & rY )const  // 2分
{
    rX = m_iX,   rY = m_iY;
}
double  Circle::getRadius ( ) const                    // 2分
{
    return m_iRadius;
}
double  Circle::getDist ( const Circle & aRef ) const    // 2分
{
    double x1, y1, x2, y2;
    this->aRef.getPos(x1, y1);
    aRef.getPos(x2, y2);
    double r = pow( x1 - x2, 2 );
    r += pow( y1 - y2, 2 );
    return sqrt(r);
}
bool  Circle::isInter ( const Circle & aRef ) const   // 2分
{
    double min_dist = this->getRadius ( ) + aRef.getRadius ( );
    double dist = this->getDist( aRef );
    if ( min_dist >= dist )return true;
    return false;
}
```

2. 六个成员函数各3分，两个成员变量各1分。

```
#ifndef  ARRAY_H
#define  ARRAY_H
#include <iostream>
class Array
{
public:
    Array ( const int aCount = 10 );
    Array ( const Array & aArray );
    ~Array ( );
    void setCount ( const int aCount );
    int getValue ( const int aIndex )const;
    void setValue ( const int aIndex, const int aValue );
private:
    int m_iCount, *m_pArray;
```

```
};
#endif
Array::Array( const int aCount ) : m_iCount(aCount), m_pArray(NULL)
{
    if(aCount<=0)return;
    m_pArray = new int[aCount];
}
Array::Array ( const Array & aArray )
             : m_iCount(aArray.m_iCount), m_pArray(NULL)
{
    if(aArray.m_iCount<=0)return;
    this->m_pArray = new int[aArray.m_iCount];
    //memcpy( this->m_pArray, aArray.m_pArray,
            this->m_iCount * sizeof(int) );
    for(int I = 0;i<m_iCount;i++)
    {
            m_pArray[i]=aArray.m_pArray[i];
    }
}
Array::~Array ( )  {
    delete []m_pArray;
}
void Array::setCount ( const int aCount )
{
    if(!aCount)return;
    int min_count = ( aCount < m_iCount ) ? aCount : m_iCount;
    m_iCount = aCount;
    int * nArray = new int[aCount];
    if(min_count)memcpy( nArray, m_pArray, min_count * sizeof(int) );
    delete []m_pArray;
    m_pArray = nArray;
}
int Array::getValue ( const int aIndex ) const
{
    //可以有错误处理
    return m_pArray[aIndex];
}
void Array::setValue ( const int aIndex, const int aValue )
{
    if(!m_pArray)return;
    if(aIndex>=m_iCount)return; //任意错误处理
    m_pArray[aIndex] = aValue;
}
```