C++第五章和第六章小测试

- 基本信息

姓名：　　学号：

1、下列程序的输出结果是：

```
# include <iostream>
using namespace std;

int x=5,y=7;
void myFunction(){
    int y=10;
    cout<<x<<" "<<y<<"\n";
}
int main(){
    cout<<x<<" "<<y<<"\n";
    myFunction();
    x = 10;
    cout<<x<<" "<<y<<"\n";
    return 0;
}
```

A、5 7
5 10
5 7

B、5 7
5 10
10 10

C、5 7
5 7
10 7

D、5 7
5 10
10 7

2、下列程序的输出结果是：

```
# include <iostream>
using namespace std;

int i = 1;
void other(){
    static int a = 2;
    static int b = 1;
    int c = 10;
    a += 2;i += 4; c+=5;
    cout<<"other\n";
    cout<<i<<" "<<a<<" "<<b<<" "<<c<<endl;
    b = a;
}
int main(){
    static int a = 1;
    int b = 3;
    int c = 0;
    cout<<"main\n";
    cout<<i<<" "<<a<<" "<<b<<" "<<c<<endl;
    c += 4 ;
    other();
    cout<<"main\n";
    cout<<i<<" "<<a<<" "<<b<<" "<<c<<endl;
    i += 5;
    other();
    return 0;
}
```

3、下面对静态数据成员的描述中，正确的是（）
A.类的不同对象有不同的静态数据成员值
B.类的每个对象都有自己的静态数据成员
C.静态数据成员是类的所有对象共享的数据
D.静态数据成员不能通过类的对象调用

4、下列程序的输出结果是：

```
# include <iostream>
using namespace std;

class Point{
    public:
        Point(){
            count++;
        }
        static void showCount(){
            cout<<count<<endl;
        }
        Point(Point &p){
            count+=2;
        }

    private:
        static int count;
};
int Point::count=0;

int main(){
    Point a;
    Point::showCount();
    Point b(a);
    Point::showCount();
    return 0;
}
```

5、下面有关友元函数的描述中，正确的说法是（）
A. 友元函数是独立于当前类的外部函数
B. 一个友元函数不可以同时定义为两个类的友元函数
C. 友元函数必须在类的外部进行定义
D. 在类的外部定义友元函数是必须加上 friend 关键字

6、下列程序的输出结果是（）

```cpp
# include <iostream>
using namespace std;

class A{
    friend class B;
    public:
        void display(){
            cout<<x+2<<endl;
        }
    private:
        int x;
};
class B{
    public:
        void set(int i);
        void display();
    private :
        A a;
};
void B::set(int i){
    a.x=i;
}
void B::display(){
    a.display();
}
int main(){
    B test;
    test.set(5);
    test.display();
    return 0;
}
```

7、下列说法中错误的是　（）

A.常对象必须进行初始化，且不能被更新。

B.常数据成员能被常成员函数和非常成员函数访问，且两函数均不能改变其值。。

C.类中的常数据成员只能通过构造函数的参数初始化表进行初始化。

D.如果既要利用引用提高程序的效率，又要保护传递给函数的数据不在函数中被改变，就应使用常引用。

8、文件作用域中定义的变量，缺省情况下都是外部变量， 但在其他文件中如果需要使用这一变量，需要使用____关键字加以声明。

A.static

B.new

C.extern

D.define

9、预处理命令在程序中都是以 () 符号开头的。

A.*

B.#

C.&

D.@

10、以下能对二维数组 a 进行正确初始化的语句是　()

A.int a[2][3] = {0};
B.int a[][3] = {{1,2},{0}};
C.int a[2][3] = {{1,2},{3,4},{5,6}};
D.int a[][3] = {1,2,3,4,5,6};

11、若有说明：int a[3][4];则对 a 数组元素的非法引用是 （）
A.a[0][2*1]
B.a[1][3]
C.a[4-2][0]
D.a[0][4]

12、下列程序的输出结果是

```
# include <iostream>
using namespace std;

void rowSum(int a[][4],int row){
    for(int i = 0;i < 4;i++){
        a[row][i] += 1;
    }
}

int main(){
    int table[3][4] = {{1,2,3,4},{2,3,4,5},{3,4,5,6}};
    rowSum(table,1);
    for(int i=0;i<3;i++){
        cout<<table[i][0]<<" ";
    }
    return 0;
}
```

13、下列程序的输出结果是：

```
#include<iostream>
using namespace std;

class Point{
public:
    Point();
    ~Point();
    void move(int newX,int newY);
private:
    int x,y;
};

Point::Point():x(0),y(0){
    cout<<"Default Constructor called"<<endl;
}
Point::~Point(){
    cout <<"Destructor called"<<endl;
}
void Point::move(int newX,int newY){
    cout<<"Moving the point to("<<newX<<","<<newY<<")"<<endl;
    x = newX;
    y = newY;
}
int main(){
    Point A[1];
    for(int i = 0;i <1;i++){
        A[i].move(i + 10, i + 20);
    }
    cout<<"Exiting main"<<endl;
    return 0;
}
```

14、有以下程序，请问程序运行后的输出结果是：

```
# include <iostream>
using namespace std;

int main(){
    int m=1,n=2,*p=&m,*q=&n,*r;
    r=p;p=q;q=r;
    cout<<m<<" "<<n<<" "<<*p<<" "<<*q;
    return 0;
}
```

A) 1 2 1 2
B) 1 2 2 1
C) 2 1 2 1
D) 2 1 1 2

15、请指出下列程序中的错误：

```
# include <iostream>
using namespace std;

int main(){
    int a=2;
    const int *p=&a;
    *p=1;
    cout<<*p;
    return 0;
}
```

16、下列程序的输出结果是

```
# include <iostream>
using namespace std;

int main(){
    int a[10] = {1,2,3,4,5,6,7,8,9,10};
    int i = 1;
    cout<<a[i]<<" ";
    i += 3;
    cout<<*(a+i)<<" ";
    int *p = a;
    cout<<p[9]<<" ";
    cout<<*(p+2);
    return 0;
}
```

17、若有定义语句：int a[4][10],*p,*q[4];且 0<=i<=4,则错误的赋值是 ()
A.p=a
B.q[i]=a[i]
C.p=a[i]
D.p=&a[2][1]

18、下列程序的输出结果是

```cpp
# include <iostream>
using namespace std;

int main(){
    int line1[] = {1,2,3};
    int line2[] = {4,5,6};
    int line3[] = {7,8,9};

    int *pline[3] = {line1,line2,line3};
    cout<<*(pline[1]+1);

    return 0;
}
```

19、下列程序的输出结果是

```cpp
# include <iostream>
using namespace std;


int main()
{
    int a[3][4]={{1,2,3,4},{5,6,7,8},{9,10,11,12}};
    int (*p)[4];
    int i=0;
    p=a;
    printf("%d\n",(*p)[11]);
    return 0;
}
```

20、下列程序的运行结果是 （）

```cpp
# include <iostream>
# include <string.h>
using namespace std;

void fun(int n,int *p){
    int f1,f2;
    if(n==1 || n==2){
        *p=1;
    }
    else{
        fun(n-1,&f1);
        fun(n-2,&f2);
        *p=f1+f2;
    }
}

int main(){
    int s;
    fun(3,&s);
    cout<<s;
    return 0;
}
```

A.2

B.3

C.4

D.5

21、有下列程序，程序的运行结果是（）

```
# include <iostream>
# include <string.h>
using namespace std;

void fun(int *a,int n){ //将a所指的数组元素从大到小排序
    int t,i,j;
    for(i=0;i<n-1;i++){
        for(j=i+1;j<n;j++){
            if(a[i]<a[j]){
                t=a[i];
                a[i]=a[j];
                a[j]=t;
            }
        }
    }
}

int main(){
    int c[10]={1,2,3,4,5,6,7,8,9,0};
    int i;
    fun(c+4,6);
    for(i=0;i<10;i++){
        cout<<c[i];
    }
    return 0;
}
```

A.1234567890

B.0987651234

C.0987654321

D.1234987650

22、下列程序的输出结果是

```
# include <iostream>
using namespace std;

class Point{
    public:
        Point(int x=0,int y=0):x(x),y(y){}
        int getX() const {return x;}
        int getY() const {return y;}
    private:
        int x,y;
};

int main(){
    Point a(4,5);
    Point *p1 = &a;
    cout<<p1->getX()<<endl;
    cout<<a.getY()<<endl;
    return 0;
}
```

4

5

23、设有定义语句 int(*f)(int);，则以下叙述正确的是（）

A）f 是基类型为 int 的指针变量

B）f 是指向函数的指针变量，该函数具有一个 int 类型的形参
C）f 是指向 int 类型一维数组的指针变量
D）f 是函数名，该函数的返回值是其类型为 int 类型的地址

24、关于 new 运算符的下列描述中，（）是错误的。
A.它可以用来动态创建对象和对象数组。
B.使用它创建的对象或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。

25、下列程序的输出结果是

```
# include <iostream>
using namespace std;

class Point{
    public:
        Point():x(0),y(0){
            cout<<"Default Constructor called"<<endl;
        }
        Point(int x,int y): x(x),y(y){
            cout<<"Constructor called."<<endl;
        }
        ~Point(){cout<<"Destructor called"<<endl;}
        int getX() const{return x;}
        int getY() const{return y;}
        void move(int newX,int newY){
            x=newX;
            y=newY;
        }
    private:
        int x,y;
};

int main()
{
    Point *ptr=new Point[2];
    ptr[0].move(5,10);
    ptr[1].move(15,20);
    cout<<"Deleting"<<endl;
    delete[] ptr;
    return 0;
}
```

26、下列关于用 vector 创建数组对象的说法中错误的是：
A.与普通数组不同的是，用 vector 定义的数组对象的所有元素都会被初始化。
B.vector 数组对象的名字也是数组的首地址。
C.vector 定义的数组对象具有一个重要的成员函数 size()，它会返回数组的大小。
D.用 vector 创建数组对象，它通过将动态数组封装成一个类，允许在调试状态下访问元素时检查下标越界的错误。

27、下列程序中的对象拷贝是属于深拷贝还是浅拷贝？ （）

```cpp
# include <iostream>
# include <string>
using namespace std;

class Array{
    public:
        Array(){ //构造函数
            m_iCount = 5;
            m_pArr = new int[m_iCount];
        }
        Array(const Array& arr){ //拷贝构造函数
            m_iCount=arr.m_iCount;
            m_pArr = arr.m_pArr;
            for(int i=0;i<m_iCount;++i){
                m_pArr[i] = arr.m_pArr[i];
            }
        }
    private:
        int m_iCount;
        int *m_pArr;
};

int main()
{
    Array arr1; // 调用构造函数
    Array arr2 = arr1; //调用拷贝构造函数
    return 0;
}
```

解析：2 个对象的指针应该指向不同的内存，拷贝的时候应该不是简单的拷贝指针地址，而是将指针指向的内存中的每一个元素依次的拷贝过来。

28、请指出下列程序的错误：

```cpp
# include <iostream>
using namespace std;

int main(){
    char s[5] ="BOOK!";
    return 0;
}
```

29、请问下列程序的输出结果是：

```c
#include <stdio.h>
#include <string.h>
#include <stdlib.h>

void print_string(const char* str)
{
    int len = strlen(str); //求str的长度
    for (int i = 0; i < len; i++)
    {
        if (str[i] == 0)
            break;
        printf("%c ", str[i]);
    }
}

int main()
{
    const char* a = "abcd";
    print_string(a);
    return 0;
}
```

30、关于 string 类的应用，下列程序的输出结果是：

```c
# include <iostream>
# include <string>
using namespace std;

int main()
{
    string s1="ABC";
    string s2 = "DEF";
    s2 += s1;
    cout<<s2<<endl;
    cout<<s2.length()<<endl;
    return 0;
}
```