

一、选择题（每题 2 分，共 40 分）

- (1) 下列关于 C++ 语言类的描述中错误的是 C
- A) 类用于描述事物的属性和对事物的操作
  - B) 类与类之间通过封装而具有明确的独立性
  - C) 类与类之间必须是平等的关系，而不能组成层次结构
  - D) 类与类之间可以通过一些方法进行通信和联络

- (2) 在 C++ 语言中标志一条语句结束的标号是 B
- A) #
  - B) ;
  - C) //
  - D) }

- (3) 以下叙述中正确的是 A
- A) 构成 C++ 语言程序的基本单位是类
  - B) 可以在一个函数中定义另一个函数
  - C) main() 函数必须放在其他函数之前
  - D) 所有被调用的函数一定要在调用之前进行定义

- (4) 已知有定义
- ```
const int D=5;
int i=1;
double f=0.32;
char c=15;
```
- 则下列选项错误的是 B
- A) ++i;
  - B) D--;
  - C) C++;
  - D) --f;

- (5) 以下程序的输出结果是 A
- ```
#include <iostream.h>
void reverse(int a[], int n)
{ int i, t;
  for(i=0; i<n/2; i++)
  { t=a[i]; a[i]=a[n-1-i]; a[n-1-i]=t; }
}
void main()
{ int b[10]={1, 2, 3, 4, 5, 6, 7, 8, 9, 10}; int i, s=0;
  reverse(b, 8);
  for(i=6; i<10; i++) s+=b[i];
  cout<<s;
}
```

- A) 22 B) 10 C) 34 D) 30

- (6) 设有数组定义：char array[]="China";，则数组 array 所占的空间为 C
- A) 4 个字节
  - B) 5 个字节
  - C) 6 个字节
  - D) 7 个字节

- (7) 若已定义：
- ```
int a[]={0, 1, 2, 3, 4, 5, 6, 7, 8, 9}, *p=a, i;
```

其中  $0 \leq i \leq 9$ ，则对 a 数组元素不正确的引用是 D

- A) a[p-a]
- B) \*(&a[i])
- C) p[i]
- D) a[10]

- (8) 以下程序的输出结果是

```
#include <iostream.h>
void main()
{ int x=1, y=3;
  cout<<x++<<" ", " ";
  {
    int x=0; x+=y*2;
    cout<<x<<" ", " <<y<<" ";
  }
  cout<<x<<" ", " <<y;
}
```

- A) 1, 6, 3, 1, 3 B) 1, 6, 3, 6, 3 C) 1, 6, 3, 2, 3 D) 1, 7, 3, 2, 3

- (9) 函数 fun 的返回值是

```
fun(char*a, char*b)
{
  int num=0, n=0;
  while(*(a+num)!='\0') num++;
  while(b[n]) {*(a+num)=b[n]; num++; n++;}
  return num;
}
```

- A) 字符串 a 的长度 B) 字符串 b 的长度  
C) 字符串 a 和 b 的长度之差 D) 字符串 a 和 b 的长度之和

- (10) 下面程序的结果为

```
#include <iostream.h>
void main()
{
  int i;
  int a[3][3]={1, 2, 3, 4, 5, 6, 7, 8, 9};
  for(i=0; i<3; i++)
    cout<<a[2-i][i]<<" ";
}
```

- A) 1 5 9 B) 7 5 3 C) 3 5 7 D) 5 9 1

- (11) 下列描述中哪个是正确的。

- A) 私有派生的子类无法访问父类的成员
- B) 类 A 的私有派生类 C 无法初始化其祖先类 A 对象的属性，因为类 A 的成员对类 C 是不可访问的
- C) 私有派生类不能作为基类派生子类
- D) 私有派生类的所有子孙类将无法继续继承该类的成员

(12) 下列关于构造函数说法不正确的是

- A) 构造函数必须与类同名  
B) 构造函数可以省略不写  
C) 构造函数必须有返回值  
D) 在构造函数中可以对类中的成员进行初始化

(13) 如果友元函数重载一个运算符时，其参数表中没有任何参数则说明该运算符是

- A) 一元运算符  
B) 二元运算符  
C) 选项 A) 和选项 B) 都可能  
D) 重载错误

(14) 在重载一个运算符时，如果其参数表中有一个参数，则说明该运算符是

- A) 一元成员运算符  
B) 二元成员运算符  
C) 一元友元运算符  
D) 选项 B) 和选项 C) 都可能

(15) 下列关于模板的说法正确的是

- A) 模板的实参在什么时候都可以省略  
B) 类模板与模板类所指的是同一概念  
C) 类模板的参数必须是虚拟类型的  
D) 类模板中的成员函数全部都是模板函数

(16) 下面程序段的运行结果是

```
void main()
{
    int t, a, b, c;
    a=1; b=2; c=2;
    while(a<b){c}
    {t=a; a=b; b=t; c--;}
    cout<<a<<" " <<b<<" " <<c;
}
```

- A) 1, 2, 0  
B) 2, 1, 0  
C) 1, 2, 1  
D) 2, 1, 1

(17) 若定义 int k=7, x=12;，则值为 3 的表达式是

- A)  $x\%=(k\%5)$   
B)  $x\%=(k-k\%5)$   
C)  $x\%=k-k\%5$   
D)  $(x\%=k)-(k\%=5)$

(18) 在位运算中，操作数每右移一位，其结果相当于

- A) 操作数乘以 2  
B) 操作数除以 2  
C) 操作数乘以 4  
D) 操作数除以 4

(19) 有以下结构体说明和变量的定义，且指针 p 指向变量 a，指针 q 指向变量 b。则不能把节点 b 连接到节点 a 之后的语句是

```
struct node
{
    char data;
    struct node*next;
}a, b, *p=&a, *q=&b;
```

A)  $a\rightarrow next=q$ ;  
B)  $p\rightarrow next=&b$ ;  
C)  $p\rightarrow next=&b$ ;  
D)  $(*p)\rightarrow next=q$ ;

(20) 一个类的友元函数能够访问该类的

- A) 私有成员  
B) 保护成员  
C) 公有成员  
D) 所有成员

(1) C++语言是从早期的 C 语言逐渐发展演变而来的。与 C 语言相比，它在求解问题方法上进行的最大改进是

( )

- A) 面向过程  
B) 面向对象  
C) 安全性  
D) 复用性

□□

□□ (2) C++语言的跳转语句中，对于 break 和 continue 说法正确的是

□□ A) break 语句只应用与循环体中

□□ B) continue 语句只应用与循环体中

□□ C) break 是无条件跳转语句，continue 不是

□□ D) break 和 continue 的跳转范围不够明确，容易产生问题

□□

□□ (3) for(int x=0,y=0;!x&& y<=5;y++)语句执行循环的次数是

- A) 0  
B) 5  
C) 6  
D) 无次数

□□

□□ (4) 考虑函数原型 void test(int a,int

□□ b=7,char="\*"),下面的函数调用中，属于不合法调用的是

□□ A) test(5); B) test(5,8); C) test(6,"#") D) test(0,0,"\*");

□□

□□ (5) 下面有关重载函数的说法中正确的是

□□ A) 重载函数必须具有不同的返回值类型;

B) 重载函数形参数必须不同;

□□ C) 重载函数必须有不同的形参列表;

D) 重载函数名可以不同;

□□

□□ (6) 下列关于构造函数的描述中，错误的是

□□ A) 构造函数可以设置默认参数;

B) 构造函数在定义类对象时自动执行

□□ C) 构造函数可以是内联函数;

D) 构造函数不可以重载

□□

□□ (7) 下面描述中，表达错误的是

B

- ☐A)公有继承时基类中的 public 成员在派生类中仍是 public 的
- ☐B)公有继承是基类中的 private 成员在派生类中仍是 private 的
- ☐C)公有继承时基类中的 protected 成员在派生类中仍是 protected 的
- ☐D)私有继承时基类中的 public 成员在派生类中是 private 的

☐

☐（ 8 ）应在下列程序划线处填入的正确语句是（ C ）

☐#include

☐class Base

☐{ public:

void fun(){cout<<"Base::fun"<<ENDL;}

☐};

☐class Derived:public Base

☐{ void fun()

☐{ \_\_\_\_\_ //显式调用基类的函数 fun()

cout<<"Derived::fun"<<ENDL;

☐ }

☐};

☐A)fun(); B)Base.fun(); C)Base::fun(); D) Base->fun();

☐

☐（ 9 ）有如下程序：

☐# include

☐class BASE

☐{

☐char c;

☐public:

☐BASE(char n):c(n){}

☐virtual~BASE(){cout<<C;}

☐};

☐class DERIVED:public BASE

☐{

☐char c;

☐public:

☐DERIVED(char n):BASE(n+1),c(n){}

☐~DERIVED(){cout<<C;}

☐};

☐int main()

☐{ DERIVED("X");

☐return 0;

☐}

☐执行上面的程序将输出（ ）

☐A)XY B)YX C)X D)Y

☐

☐（ 10 ）在进行完任何 C++流的操作后，都可以用 C++流的有关成员函数检测流的状态；其中只能用于检测输入流状态的操作函数名称是（ B ）

☐A)fail B)eof C)bad D)good

二、填空题（每题 2 分，共 10 分）

（ 1 ）下面程序的打印结果是 [ ]

☐#include <iostream.h>

☐int f(int);

☐int main()

☐{ int i;

☐for(i=0;i<5;i++)

☐cout<<f(i)<<"";

☐return 0;

☐}

☐int f(int i)

☐{ static int k=1;

☐for(i>0;i- -)

☐k += i;

☐return k;

☐}

（ 1 ）下列程序中画线处应填入的语句是

class Base

{

```
        public:
            void fun() {cout<<" Base of fun" <<endl;}
};
class Derived: public Base
{
    void fun()
    {
        [                ] //调用基类的成员函数 fun
        cout<<" Derived of fun" <<endl;
    }
};
```

(2) 在用 class 定义一个类时，数据成员和成员函数的默认访问权限是

[ ]

□□ (3) 含有纯虚函数的类称为 [ ]

□□ (4) 已知 int DBL(int n){return n + n;} 和 long DBL(long n){return n + n;} 是一个函数模板的两个实例，则该函数模板的定义是

[ ]

□□ (5) 在下面程序的横线处填上适当的语句，使该程序执行结果为 10.

```
□□ #include
□□ class MyClass
□□ {    public:
□□     MyClass(int a){x = a;}
□□     [                ] //取 x 值
□□     private
□□     int x;
□□ };
□□ int main()
□□ {    MyClass my(10);
□□     cout<<MY.GetNum()<<ENDL;
□□     return 0;
□□ }
```

三、阅读程序（每题 5 分，共 40 分）

1、下面程序的输出结果是

```
#include<iostream. h>
#include<string. h>
void main()
{
    char p1[10], p2[10];
    strepy(p1, " abc" );
```

```
    strcpy(p2, " ABC" );
    char str[50]=" xyz" ;
    strcpy(str+2, strcat(p1, p2));
    cout<<str;
}
```

运行结果是：

2、下面程序的结果为

```
#include<iostream. h>
int c;
class A
{
    private:
        int a;
        static int b;
    public:
        A() {a=0;c=0;}
        void seta() {a++;}
        void setb() {b++;}
        void setc() {c++;}
        void display() {cout<<a<<" " <<b<<" " <<c;}
};
int A:: b=0;
void main()
{
    A a1, a2;
    a1.seta();
    a1.setb();
    a1.setc();
    a2.seta();
    a2.setb();
    a2.setc();
    a2.display();
}
```

运行结果是：

3.写出下列程序的运行结果.

```
#include<iostream.h>
void main()
```

```
{
    int a[2][2]={1,2,3,4},*p;
    p=a[0]+1;
    cout<<*p<<endl;
}
```

运行结果是：

4.写出下列程序的运行结果.

```
#include<iostream.h>
void main()
{
    int x;
    int &p=x;
    x=10;
    p=x+10;
    cout<<x<<","<<p<< ;<endl;
}
```

运行结果是：

5.写出下列程序的运行结果

```
#include<iostream.h>
int a=100;
void fun()
{
    int a=0;
    a++;
    ::a=200;
    cout<<"The a of fun is"<<a<<endl;
    cout<<"::a="<<::a<< endl;
}
void main()
{
    int a=10;
    fun();
    a++;
    ::a+=1;
    cout<<"The a of main is"<<a<<endl;
    cout<<"::a="<<::a<< endl;
}
```

运行结果是：

6.写出下列程序的运行结果.

```
#include<iostream.h>
int f(int a)
{
    return ++a;
}
int g(int &a)
{
    return ++a;
}
void main()
{
    int m=0,n=0;
    m+=f(g(m));
    n+=f(f(n));
    cout<<"m="<<m<<end l;
    cout<<"n="<<n<<end l;
}
```

运行结果是：

7. 写出下列程序的运行结果.

```
#include<iostream.h>
class MyClass
{
public:
    void SetValue(int val);
    MyClass();
    ~MyClass();
private:
    int i;
};
MyClass::MyClass()
{
    i=0;
    cout<<"This is a constructor!i="<<i<<endl;
}
void MyClass::SetValue(int val)
{
    i=val;
```

```
        cout<<"i="<<i<<endl;
    }
    MyClass::~MyClass()
    {
        cout<<"This is a destructor!i="<<i<<endl;
    }
    void main()
    {
        MyClass * myl[3];
        int k;
        for(k=0;k<3;k++)
            myl[k]=new MyClass;
        for(k=0;k<3;k++)
            delete myl[k];
    }
    运行结果是:
```

8. 写出下列程序的运行结果.

```
#include<iostream.h>
class Base
{
public:
    Base(int i=0):x(i){}
    virtual int sum() const{ return x;}
private:
    int x;
};
class Derived:public Base
{
public:
    Derived(int i=0,int j=0):Base(i),y(j){}
    int sum() const { return Base::sum()+y;}
private:
    int y;
};
void Call(Base b)
{
    cout<<"sum="<<b.sum()<<" ";<<endl;
}
void main()
```

```
{
    Base b(10);
    Derived d(10,40);
    Call(b);
    Call(d);
}
运行结果是:
9.写出下列程序的运行结果.
#include<iostream.h>
class MyClass
{
public:
    MyClass(int a)
    {
        X=a;
        cout<< "This is"<<X<<"'s constructor."<<endl;
    }
    ~MyClass()
    {
        cout<< "This is "<<X<<"'s destructor."<<endl;
    }
private:
    int X;
};
MyClass globalObj(0);
void main()
{
    MyClass commonObj(1);
    static MyClass staticObj(2);
}
运行结果是:
```

10. 写出下列程序的运行结果.

```
#include<iostream.h>
class CStatic
{
public:
    CStatic(){val++;}
    static int val;
};
int CStatic::val=0;
void main()
{
    cout<<"CStatic::val="<<CStatic::val<<endl;
    CStatic cs1;
    cout<<"cs1.val="<<cs1.val<<endl;
    CStatic cs2;
    cout<<"cs2.val="<<cs2.va l<<endl;
    CStatic cs3,cs4;
    cout<<"cs1.val="<<cs1.va l<<endl;
    cout<<"cs2.val="<<cs2.va l<<endl;
}
```

11. #include<iostream.h>

```
class Data
{
public:
    Data(int x)
    {
        Data::x=x;
        cout<< "Data cons."<<endl;
    }
    ~Data(){ cout<<"Data des."<<endl; }
private:
    int x;
};
class Base
{
```

```
public:
    Base(int x):d1(x)
    {cout<<"Base cons."<<endl;}
    ~Base(){cout<<"Base des."<<endl;}
private:
    Data d1;
};
class Derived:public Base
{
public:
    Derived(int x):Base(x),d2(x){cout<<"Derived cons."<<endl;}
    ~Derived(){cout<<"Derived des."<<endl;}
private:
    Data d2;
};
void main()
{
    Derived obj(5);
}
```

运行结果是:

12 写出下列程序的运行结果.

```
#include<iostream.h>
template<class T1,class T2>
void fun(T1 &x,T2 &y)
{
    if(sizeof(T1)>sizeof(T2))
        x=(T1)y;
    else
        y=(T2)x;
}
void main()
{
    double d;
    int i;
```

9999 8.8.

fun( 99.99 , 88

x = 88.0

```
d=99.99;
i=88;
fun(d,i);
cout<<"d="<<d<<"i= "<<i<<endl;
d=8.8;
i=9999;
fun(i,d);
cout<<"d="<<d<<"i= "<<i<<endl;
}
```

88 88

8888 9999

运行结果是：

五、编程题（每题 5 分，共 10 分）

- 1.设计一个类 **CRectangle**,要求如下所述.  
(1)该类中的私有成员变量存放 **Rectangle** 的长和宽,并且设置它们的默认值是 **1**.  
(2)通过成员函数设置其长和宽,并确保长和宽都在**(0,50)**范围之内.  
(3)求周长 **Perimeter**.
- 2.定义 **Point** 类,有数据成员 **X** 和 **Y**,重载**++**和**-**运算符,要求同时重载前缀方式和后缀方式.