

一、单选题

1. 关于 C++, 正确的是 (**B**)

- A) 它是基于函数式的
- B) 它是面向对象的
- C) 它是过程式的
- D) 以上皆不对

2. 在 C++ 中, 指针类型是一种 (**B**)

- A) 基本数据类型
- B) 构造数据类型
- C) 抽象数据类型
- D) 整数类型

3. C++ 中变量的生存期类型不包括 (**B**)

- A) 静态生存期
- B) 全局生存期
- C) 自动生存期
- D) 动态生存期

4. 下面说法正确的是 (**A**)

- A) 内联函数在编译时是将该函数的目标代码展开到该函数调用点
- B) 内联函数在运行时是将该函数的目标代码展开到该函数调用点
- C) 内联函数在连接时是将该函数的目标代码展开到该函数调用点
- D) 以上皆不对

5. 以下哪些函数可以构成重载函数 (**B**)

- (1) `int function(int a, int b, int c);`
- (2) `double function(int a, int b, double c);`
- (3) `int function(int a, int b);`
- (4) `double function(int a, int b, int c);`

- A) (1)(2)(4)
- B) (1)(2)(3)
- C) (1)(3)(4)
- D) (1)(4)

6. 能够把引用传递给函数的方式是 (**D**)

- A) 形参是指针, 实参是地址
- B) 形参是数组名, 实参是数组名
- C) 形参和实参都是变量
- D) 形参是引用, 实参是变量

7. 以下几种引用的使用方法, 正确的是 (**C**)

- A) `int &a = &n;`
- B) `int &a;`
- C) `int &a = n;`
- D) `int &a = 10;`

8. 关于 `this` 指针的说法不正确的是 (**D**)

- A) 不能在程序中修改 `this` 指针
- B) `this` 指针可以给其他指针赋值, 但不能修改 `this` 指针
- C) 静态成员函数中没有 `this` 指针
- D) `this` 指针的类型为: `<类名> const *this`

正确答案:
`<类名>* const this`

9. 以下对拷贝构造函数的说法, 不正确的是 (**D**)

- A) 利用拷贝构造函数, 可以用同类型的另一个对象来初始化新对象
- B) 当把对象作为值参数传给函数时会调用拷贝构造函数
- C) 隐式拷贝构造函数会默认调用成员对象类的拷贝构造函数来实现成员对象的初始化
- D) 隐式拷贝构造函数不会初始化指针成员变量

10. 下面关于静态成员的描述中，正确的是 (D)
- A) 静态数据成员不能通过类名来访问
 - B) 类的每个对象都有自己的静态数据成员
 - C) 类的不同对象有不同的静态数据成员值
 - D) 静态数据成员是类的所有对象共享的数据
11. 在下列运算符中，不能重载的是 (A)
- A) sizeof
 - B) ->
 - C) new
 - D) delete
12. 下列描述中，表达错误的是 (A)
- A) 公有继承时，基类中的 `private` 成员在派生类中仍是 `private` 的
 - B) 公有继承时，基类中的 `protected` 成员在派生类中仍是 `protected` 的
 - C) 公有继承时，基类中的 `public` 成员在派生类中仍是 `public` 的
 - D) 私有继承时，基类中的 `public` 成员在派生类中是 `private` 的
13. 设有基类的定义如下：
- ```
class Base{
 private: int a, c;
 protected: int b;
};
```
- 派生类采用何种继承方式可以使成员变量 `b` 变成自己的私有成员 ( C )
- A) 公有继承
  - B) 保护继承
  - C) 私有继承
  - D) 保护继承或私有继承
14. 在 C++ 中，用于实现动态绑定的是 ( C )
- A) 重载函数
  - B) 内联函数
  - C) 虚函数
  - D) 模板函数
15. 要将类 A 说明是类 B 的虚基类，正确的描述是 ( C )
- A) `virtual class B: public A`
  - B) `class virtual B: public A`
  - C) `class B: virtual public A`
  - D) `class B: public A virtual`
16. 关于使用模板的意义，以下说法正确的是 ( B )
- A) 为了提高代码的运行效率
  - B) 为了提高代码的复用性
  - C) 加强类的封装性
  - D) 以上皆不对
17. 下列函数模板定义中，不合法的是 ( A )
- A) `template(class T) void fun(T a, T b)`
  - B) `template<class T> void fun(T a, T b)`
  - C) `template<class T> T fun(T a, float *p)`
  - D) `template<class T> float fun(float *p)`
18. 定义字符数组 `char *p = "abcd"`，能正确输出字符数组内存首地址的是 ( D )
- A) `cout << &p;`
  - B) `cout << p;`
  - C) `cout << &p[0];`
  - D) `cout << (void*)p;`
19. 当创建一个 `ifstream` 类的对象，并与外部文件建立联系，则文件默认的打开方式为 ( A )
- A) `ios::in`
  - B) `ios::out`
  - C) `ios::in | ios::out`
  - D) `ios::binary`

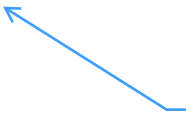
20. 设已有 double 型变量 data, 以二进制方式把 data 的值写入输出文件流对象 outfile 中去, 正确的语句是 ( C )

- A) outfile.write((double\*)&data, sizeof(double));
- B) outfile.write((double\*)&data, data);
- C) outfile.write((char\*)&data, sizeof(double));
- D) outfile.write((char\*)&data, data);

## 二、填空题

1. 定义类 date 的带默认值 (为 2019 年 1 月 1 日) 的构造函数与拷贝构造函数。

```
class date {
 private:
 int year, month, day;
 public:
 date(____[1]____) { year=y; month=m; day = d; }
 答: int y=2019, int m=1, int d=1
 date(____[2]____) { year=d1.year; month=d1.month; day=d1.day; }
 答: const date &d1
};
void main()
{
 date d1(2019, 6, 14);
 date d2 = d1;
}
```



此处可以不加const

2. 写出以下程序的运行结果。

```
#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() { cout << "constructing B2" << endl; }
 ~B2() { cout << "destructing B2" << endl; }
};
```

```

class C: public B2, virtual public B1 {
 int j;
public:
 C(int a, int b, int c): B1(a), memberB1(b), j(c) {}
private:
 B1 memberB1;
 B2 memberB2;
};
void main() {
 C obj(1, 2, 3);
}

```

该程序的运行结果如下：

```

_____ [3] _____
_____ [4] _____
_____ [5] _____
_____ [6] _____

```

答：

constructing B11

constructing B2

constructing B12

constructing B2

destructing B2

destructing B1

destructing B2

destructing B1

### 三、简答题

1. 为什么在 C++ 中需要有 **this** 指针？它的作用是什么？

答：C++ 中类定义中声明的非静态数据成员对该类的每个对象都有一个拷贝，而成员函数是被该类的每个对象共享的，为了确定成员函数中用到的数据成员是属于哪一个对象的，C++ 采用了一个 **this** 指针解决这个问题。

每个成员函数都有一个缺省的形式参数 **this**，其类型为指向该类对象的指针；调用一个对象的成员函数时，实现系统会把该对象的地址传给成员函数。在成员函数中访问类中定义的数据成员时，实际访问的是 **this** 指向的对象的数据成员。

2. 虚函数、纯虚函数的作用分别是什么？

答：虚函数的主要作用是实现“动态绑定”。

包含纯虚函数的类称为抽象类，抽象类为派生类的提供了一个基本框架和一个公共的对外接口。

3. 在多继承中，什么情况下会出现命名冲突，怎样解决命名冲突？什么情况下会出现重复继承，怎样解决重复继承？

**答：**在多继承中会出现两个问题：命名冲突和重复继承。在多继承中，当多个基类中包含同名的成员时，它们在派生类中就会出现命名冲突问题；在多继承中，如果直接基类有公共的基类，就会出现重复继承，这样，公共基类中的数据成员在多继承的派生类中就有多个拷贝。

在 C++ 中，解决命名冲突的方法是采用基类名受限；解决重复继承问题的手段是采用虚基类。

## 四、编程题

1. 定义一个复数类 `Complex`，在构造函数中初始化复数的实部 `real` 和虚部 `imag`。然后，为 `Complex` 类重载以下两个运算符：

(1) 重载运算符 `==`，判断两个复数是否相等；

(2) 重载运算符 `-`，实现两个复数的减法。

**答：**

```
class Complex{
private:
 double real, imag;
public:
 Complex(double r=0, double i=0): real(r), imag(i) {};
 friend bool operator == (const Complex &c1, const Complex &c2);
 friend Complex operator - (const Complex &c1, const Complex &c2);
};
```

```
bool operator == (const Complex &c1, const Complex &c2)
{
 return (c1.real == c2.real) && (c1.imag == c2.imag);
}
```

```
Complex operator - (const Complex &c1, const Complex &c2)
{
 Complex temp(c1.real-c2.real, c1.imag-c2.imag);
 return temp;
}
```

2. 定义一个函数模板，它带有两个参数：(1) 元素类型可变的数组 `x`；(2) 该数组的长度 `n`。要求：返回数组 `x` 中的最大值。

答:

```
template <class T>
```

```
T Max(T x[], int n)
```

```
{
```

```
 int i_max = 0;
```

```
 for (int i = 1; i < n; i++)
```

```
 if (x[i] > x[i_max])
```

```
 i_max = i;
```

```
 return x[i_max];
```

```
}
```

如果T是我们自己定义的类，则需要为它重载<运算符。