1. 选择题
2. 下列哪个选项描述了C++相对于C语言的主要优势？
3. 更简单的语法
4. 更高效的性能
5. 更强大的面向对象特性
6. 更广泛的应用领域
7. 在C++中，用于动态分配内存的关键字是？
8. malloc
9. allocate
10. new
11. create
12. 请问下列关于构造函数的说法中，哪一种不正确？
13. 构造函数可以为const函数
14. 构造函数不能为虚函数
15. 默认构造函数不会初始化复合类型成员变量
16. 类的成员变量在构造函数中初始化
17. 下面程序段的输出结果是( )。

#include<stdio.h>

**void** fun(**int** px, **int** pa, **int** \*pb){

     \*pb = pa + px;

     pa = \*pb + px;

}

**int** main(){

**int**  x = 4, a = 2, b = 2;

     fun(x, a, &b);

**printf**("%d,%d", a, b);

}

1. 2, 2
2. 2, 6
3. 6, 2
4. 6, 6
5. 计算机假定要对类AB定义加号操作符重载成员函数，实现两个AB类对象的加法，并返回相加结果，则该成员函数的声明语句最好为（）
6. AB operator+(AB& A,AB& B)
7. AB &operator+(AB A)
8. AB operator+(AB &A)
9. AB \*operator+(AB A)
10. 填空题
11. C++ 中，用于将一个类的声明（包括成员函数和成员变量的声明）放在头文件，而将成员函数的定义放在实现文件的是 '类的模块化'。
12. 在C++ 中，你可以在函数声明或定义时为参数提供默认值，这样在调用函数时如果没有提供这些参数的值，编译器将使用默认值。这个特性称为函数的默认参数
13. 在C++ 中，用于实现类的成员函数访问成员变量的指针是 "\_\_this\_\_"。
14. 在C++ 中，用于表示一个类不能被继承的关键字是 "\_\_final\_\_"。
15. 在C++ 中，用于在函数声明或定义中指定默认参数的符号是 \_=\_。
16. 简答题
17. 简要解释C++ 中的浅拷贝和深拷贝的概念，并举例说明。

答案：浅拷贝是指简单地复制对象的成员变量的值，两个对象共享同一块内存。深拷贝是指创建一个新的对象，并将原对象的成员变量的值复制到新对象中，两个对象拥有独立的内存空间。例如，在类中如果有指针类型的成员变量，浅拷贝只是复制指针的值，而深拷贝会创建新的内存空间，并复制指针指向的内容。

1. 什么是菱形继承？它在C++ 中可能导致什么问题？

答案：菱形继承是指一个类同时继承了两个类，而这两个类又都继承了同一个父类，形成了一个菱形的继承关系。这种继承关系可能导致数据冗余和虚函数的二义性问题，因为派生类会继承两份相同的基类数据。C++中通过虚继承（virtual inheritance）来解决这个问题。

1. 在完成某项工作时，你认为领导要求的方式不是最好的，自己还有更好的方法，你应该怎么做？

回答提示：①.原则上我会尊重和服从领导的工作安排；同时私底下找机会以请教的口吻，婉转地表达自己的想法，看看领导是否能改变想法；

②如果领导没有采纳我的建议，我也同样会按领导的要求认真地去完成这项工作；

③.还有一种情况，假如领导要求的方式违背原则，我会坚决提出反对意见；如领导仍固执己见，我会毫不犹豫地再向上级领导反映。

1. 解释C++中的引用（Reference）是什么，以及与指针有何不同？

答案：引用是一个已存在变量的别名。与指针不同，引用在创建时必须被初始化，并且不能重新指向其他变量。引用更安全、更易读，并且通常用于传递函数参数。

1. 什么是析构函数？为什么它在C++中很重要？

答案：析构函数是类的特殊成员函数，用于释放对象使用的资源。它与构造函数相反，当对象被销毁时自动调用。在C++中，析构函数非常重要，因为它确保了对象在不再使用时释放资源，避免内存泄漏和资源浪费。

1. 编程题
2. 编写一个C++程序，定义一个简单的类 `Rectangle`，具有私有成员变量 `width` 和 `height`，公有成员函数 `getArea()` 用于计算矩形的面积，并提供构造函数和析构函数。

```cpp

// Rectangle.h

#ifndef RECTANGLE\_H

#define RECTANGLE\_H

class Rectangle {

private:

int width;

int height;

public:

Rectangle(int w, int h);

~Rectangle();

int getArea();

};

#endif // RECTANGLE\_H

```

```cpp

// Rectangle.cpp

#include "Rectangle.h"

Rectangle::Rectangle(int w, int h) {

width = w;

height = h;

}

Rectangle::~Rectangle() {}

int Rectangle::getArea() {

return width \* height;

}

```

1. 编写一个C++程序，模拟实现一个简单的图书馆系统，包括图书的借阅和归还功能。使用类和对象来表示图书和用户，并在程序中实现相应的操作。

```cpp

// Library.h

#ifndef LIBRARY\_H

#define LIBRARY\_H

#include <string>

class Book {

private:

std::string title;

bool available;

public:

Book(std::string t);

void borrowBook();

void returnBook();

};

class User {

private:

std::string name;

public:

User(std::string n);

void borrowBook(Book &book);

void returnBook(Book &book);

};

#endif // LIBRARY\_H

```

```cpp

// Library.cpp

#include "Library.h"

Book::Book(std::string t) {

title = t;

available = true;

}

void Book::borrowBook() {

if (available) {

available = false;

std::cout << "Book \"" << title << "\" has been borrowed." << std::endl;

} else {

std::cout << "Book \"" << title << "\" is not available." << std::endl;

}

}

void Book::returnBook() {

available = true;

std::cout << "Book \"" << title << "\" has been returned." << std::endl;

}

User::User(std::string n) {

name = n;

}

void User::borrowBook(Book &book) {

book.borrowBook();

}

void User::returnBook(Book &book) {

book.returnBook();

}

```