中南大学考试试卷

2021 -- 2022 学年 2 学期 时间 100 分钟 2021 年 5 月 8 日

面向对象编程(C++) 课程 48 学时 3 学分 考试形式： 闭 卷

专业年级： 2021 级自动化与电气类 总分 100 分

注：此页不作答题纸，请将答案写在答题纸上

# 一、填空题（每空 2 分，共 20 分）

1. 将一个函数声明为一个类的友元函数，必须使用关键字 （1） 。
2. 若已经定义了一个int类型的变量a，现要求定义a的引用变量ref\_a，ref\_a的定义方式为 （2） 。
3. 假定A为一个类，则执行语句：A a(2), b[3], \*p[4];时，调用该类构造函数 （3） 次。
4. 假设ip为已经定义好的整型指针，为动态得到一个包含10个整数的数组，并由ip指向该数组，应使用的语句为 （4） ，当要释放ip指向的动态数组对象时，使用的语句为： （5） 。
5. 当一个函数的代码较少且需要频繁调用时，可以将其定义为内联函数，此时需在函数前面使用关键字 （6） 说明该函数为内联函数。
6. 一个抽象类的派生类可以实例化的必要条件是实现了所有的 （7） 。
7. 假定类student中有一个公用属性的静态数据成员static float score；在类外不通过对象名给该成员

score赋值为90的语句为： （8） 。

1. 假设程序中用如下的语句定义了常量PI和变量a：

const float PI = 3.1415926; float a;

现要求定义一个指向常量的指针p1和常指针p2，分别指向PI和a，p1和p2的数据类型均为float， 则应该使用的语句分别为 （9） 和 （10）

# 二、程序阅读题（每小题 4 分，共 16 分）

|  |  |
| --- | --- |
| 1. 阅读下列程序写出运行结果  #include<iostream> using namespace std; class point  {  int x, y; public:  point()  {  x=0;y=0;cout<<"子对象构造函数"<< endl;  }  ~point()  {  cout<<"子对象析构函数"<< endl;  }  };  //（右侧后续） | class line  {  point origin\_point; int delta\_x, delta\_y;  public:  line()  {  delta\_x=1; delta\_y =1;  cout <<"构造函数" << endl;  }  ~ line (){ cout <<"析构函数" << endl; }  };  int main()  { line line1; return 0;  } |

|  |  |
| --- | --- |
| 2. 阅读下列程序写出运行结果  #include<iostream> using namespace std; class counter  { static int count; public:  void static setcount(int i)  { count = i; } void showcount( )  { cout << count << endl; }  };  //（右侧后续） | int counter::count = 10; int main()  {  counter a, b; a.setcount(20); a.showcount(); b.showcount(); counter::setcount(40); a.showcount(); b.showcount();  return 0;  } |
| 3. 阅读下列程序写出运行结果  #include <iostream> using namespace std; class A  {public:  int a, b;  A() {a=0;b=0;}  void set(int x, int y){a = x; b = y;} void print( )  {cout<<"A.a="<<a<<", A.b ="<<b<< endl;}  };  class B: public A  {public:  int b, c;  B() {b=0; c=0;}  void set( int x, int y){b = x; c = y;} void print( )  {cout<<"B.b="<<b<<", B.c ="<<c<<endl; }  };  int main( )  { B objB; objB.A::set(1,2); objB.print();  objB.set(2,3);  objB.A::print(); return 0;  } | 4. 阅读下列程序写出运行结果  #include <iostream> using namespace std; class Location  { int X, Y; public:  void set(int X1, int Y1)  { X += X1; Y += Y1; }  int GetX()  { return X; } int GetY()  { return Y; }  };  void display(Location& rL)  {  cout << rL.GetX() << ", " << rL.GetY() << "\n";  }  int main()  { Location A[5] = {{1,1},{2,2},{3,3},  {4,4},{5,5}};  Location \*rA = A;  for (int i=0;i<5;i++,i++) (rA+i)->set(2,2);  A[3].set(3,3);  for (int i=0;i<5;i++) display(\*(rA++));  } |

**三、程序补全题：下列程序缺少若干语句，在空白处填上合适的代码（每空 2 分，共 16 分）**

|  |  |
| --- | --- |
| 1. 假设存在抽象基类 Shape，由它派生出 3 个层级的派生类： Point( 点) 、Circle( 圆形) 和  Cylinder(圆柱体)。要求：类中计算面积的函数为 area(抽象基类中的纯虚函数)，在 main 函数中利用基类指针输出派生类对象的面积。  #include<iostream> using namespace std; Class Shape  {  （1） void area() const=0;  …  } //（右侧后续） | int main()  {  Point point1(0, 0);  Circle circle1(point1, 15);  Cylinder cylinder1(circle1, 9);  Shape \*pt[3] = {&point1, &circle1, &cylinder1}; for(int i=0;i<3;i++)  cout<< （2） <<endl; return 0;  } |

|  |  |  |
| --- | --- | --- |
| 2. 假设存在类 X，要求对 X 类以**成员函数**的形式重载后缀自增运算符。  #include<iostream> using namespace std;  … class X  {  int a, b; public:  X() {a=0; b=0;}  X operator++( int );  } //（右侧后续） | X operator++( int )  {  X temp= （4） ; a++;  b++;  return temp;  } | |
| 3. 假设有 Text(文字)和 Picture(图片)两个类，要求：以 Text 和 Picture 为基类，以公有继承的方式派生出表情包类(Emoji)。  #include <iostream> using namespace std;  class Emoji : （5） , （6）  {  … public:  Emoji( ){cout<<"多重派生类构造函数";}  …  }; | | 4. 求解一元二次方程，要求判断方程是否有解， 无解时用标准出错信息流(标准错误信息流)向屏幕输出无解错误信息，有解时用标准输出流输出方程的解。  #include <iostream>  #include <cmath> int main()  {  float a, b, c, disc; cout<<"请输入 a, b, c"; cin>>a>>b>>c;  if (a==0)||((disc=b\*b-4\*a\*c)<0)  （7） <<"无解"<<endl; else  （8） <<"x1="<<(-b+sqrt(disc))/(2\*a)<<endl  <<"x2="<<(-b-sqrt(disc))/(2\*a)<<endl;  } |

**四、程序改错题，找到错误，分析原因并改正（每小题 4 分，共 16 分）**

|  |  |
| --- | --- |
| 1、标有行号的代码有两处错误，找出并改正  #include <iostream> using namespace std;  (1) class Test  (2) {   1. int x, y; 2. void fun(int i, int j) (5) {x = i; y = j;}   (6) void show()  (7) { cout << "x = " << x;   1. if(y) cout << ", y = " << y << endl; 2. cout << endl;   (10) } (11)};  int main()  { Test a;  a.fun(1); a.show();  a.fun(2,4); a.show();  } | 2、下面的程序定义了一个表示复数的类 Com，对加法运算符进行重载，其中两行代码有错。  (1) class Com  (2) {   1. double Re = 0, Im = 0; 2. public: 3. Com( double r = 0, double i = 0) 4. { Re = r; 5. Im = i;   (8) }  (9) Com operator+(Com, Com); (10)};  (11)Com operator+(Com c1, Com c2) (12){   1. Com tmp; 2. tmp.Re = c1.Re + c2.Re; 3. tmp.Im = c1.Im + c2.Im; 4. return tmp; (17)} |

|  |  |
| --- | --- |
| 3、标有行号的代码有两处错误，找出并改正  (1) class base  (2) {  (3) public: int b;  (4) base() { b = 0; } (5) };  (6) class derived1: public base  (7) {   1. public: 2. int d1;   (10) derived1() { d1 = 0; } (11)};  (12)class derived2: public base (13){   1. public: 2. int d2;   (16) derived2() { d2 = 0; } (17)};  class derived: public derived1, derived2  {  public:  void set() { b = 1; d1 = 1; d2 = 2;}  }; | 4、已知如下基类和派生类定义，main 函数中有两处错误，分析错误原因或说明改正方法。  class Base  {  int base\_int; public:  Base() { base\_int = 1; }  void print\_base() { cout << base\_int << endl; }  };  class Derived : public Base  { int derived\_int; public:  Derived() { derived\_int = 1; }  void print\_derived() { cout << derived\_int << endl; }  };   1. int main() 2. { Base \*bp, Bobj; 3. Derived \*dp, Dobj; 4. bp=&Dobj; 5. bp->print\_derived(); 6. dp = &Bobj; 7. dp->print\_derived();   (8) } |

**五、 简答题（每小题 4 分，共 12 分）**

1、简述友元函数重载运算符和成员函数重载运算符的异同。

2、简述 new 运算符和 delete 运算符的作用及其关系。

3、简述结构化程序设计方法与面向对象程序设计方法的基本思想。

# 六、 编程题（第 1 题 12 分，第 2 题 8 分，共 20 分）

1、关于交通工具的程序设计

（1）设计一个表示交通工具的抽象基类 vehicle，该类中至少包括速度、颜色两个属性（私有成员）， 以及设置这些属性的成员函数（这些函数均为虚函数，其中至少有一个为纯虚函数）。

（2）由 vehicle 公有派生出 car、plane、ship 三个类，每个类中新增最大载客数（私有成员），并对基类中的虚函数重新定义；新增一个设置参数的成员函数，设置各交通工具的基本属性（速度、颜色、最大载客数）；新增一个 print 函数，输出该交通工具的基本属性：速度、颜色、最大载客数。

（3）main 函数中，对不同类型的交通工具，进行参数设置与输出，要求实现动态多态性。

2、关于时间 Time 的程序设计

（1）设计一个表示时间的类 Time，其私有数据成员包括三个 int 类型的数：h, m, s，分别表示小时、分钟和秒钟。

（2）设计一个普通构造函数和一个拷贝构造函数，实现时间的初始化。

（3）重载运算符“-”和“<<”，实现时间的减法运算和时间的输出。注：时间的减法运算是指两个时间之间差的秒数，以非负整数表示。