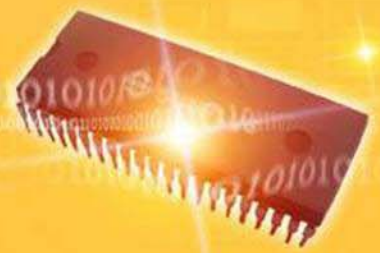
嵌入式系统工程师



动态联编与虚函数

课程内容

- 静态联编
- 虚函数
- 纯虚函数和抽象类
- 虚析构函数

定义

- 联编是指一个计算机程序自身彼此关联的过程
- 静态联编：编联工作出现在编译连接阶段，这种联编又称为早期联编，因为联编过程是在程序开始之前完成的
- 动态联编：编译程序在编译阶段并不能确切知道将要调用的函数，只有在程序运行时才能确定将要调用的函数，这要求联编工作要在程序运行时进行，这种在程序运行时进行联编工作称为动态联编

静态联编

例5.1静态联编

```
#include <iostream.h>
class Point
{
   double x, y;
public:
   Point(double i, double j) { x=i; y=j; }
   double Area() const { return 0.0; }
};
class Rectangle:public Point
{
public:
   Rectangle(double i, double j, double k, double l);
   double Area() const { return w*h; }
private:
   double w, h;
};
```

静态联编

```
Rectangle::Rectangle(double i,double j,double k,
                     double l):Point(i, j)
{
   w=k; h=l;
}
void fun(Point &s)
{
   cout<s.Area()<<endl;
}
int main()
{
   Rectangle rec(3.0, 5.2, 15.0, 25.0);
   fun(rec);
}
```

运行结果: 0

问题

- 如何得到我们想要的结果呢？

课程内容

- 静态联编
- 虚函数
- 纯虚函数和抽象类
- 虚析构函数

虚函数

- C++规定动态联编是在虚函数的支持下实现的
- 虚函数是动态联编的基础。虚函数是非static的成员函数
- 说明虚函数的方法如下：

  virtual <类型说明符><函数名>(<参数表>);

动态联编

例5.2 动态联编

```
#include <iostream.h>
class Point
{
 public:
   Point(double i, double j) { x=i; y=j; }
   virtual double Area() const { return 0.0; }
 private:
   double x, y;
};
class Rectangle:public Point
{
 public:
   Rectangle(double i, double j, double k, double l);
   virtual double Area() const { return w*h; }
 private:
   double w, h;
};
```

动态联编

```
Rectangle::Rectangle(double i, double j, double k,
   double l):Point(i, j)
{
   w=k; h=l;
}
void fun(Point &s)
{
   cout<s.Area()<<endl;
}
int main()
{
   Rectangle rec(3.0, 5.2, 15.0, 25.0);
   fun(rec);
}
```

虚函数替换的条件

- 从上面的例子可以看到，派生类中对基类的虚函数进行替换时，要求两者满足如下条件:
  - 与基类虚函数有相同的参数个数
  - 其参数的类型与基类的虚函数的对应的参数类型相同
  - 其返回值与基类函数的相同
- 满足上述条件的派生类成员函数，自然是虚函数，可以不必加virtual说明

课程内容

- 静态联编
- 虚函数
- 纯虚函数和抽象类
- 虚析构函数

纯虚函数

- 纯虚函数是一种特殊的虚函数，它的一般格式如下：

```
class <类名>
{
    virtual<类型><函数名>(<参数表>)=0;
}
```

抽象类

- 带有纯虚函数的类称为抽象类
- 抽象类是不能定义对象的，在实际中为了强调一个类是抽象类，可将该类的构造函数声明为保护的控制权限
- 抽象类只能作为基类来使用，其纯虚函数的实现由派生类给出。如果派生类中没有重定义纯虚函数，则这个派生类仍然还是一个抽象类

抽象类

例5.3抽象类

```
class Vehicle
{
public:
  Vehicle(float speed,int total)
  {
       this->speed = speed;
       Vehicle::total = total;
   }
 virtual void ShowMember()=0;//纯虚函数的声明
protected:
  float speed;
  int total;
};
```

抽象类

```
class Car:public Vehicle
{public:
  Car(int aird,float speed,int total):Vehicle(speed,total)
   {Car::aird = aird;
  }
  virtual void ShowMember(){  //派生类成员函数重载
   cout<<speed<<"|"<<total<<"|"<<aird<<endl;
  }
 protected:
   int aird;
};
int main(){
    //Vehicle a(100,4);//错误,抽象类不能创建对象
    Car b(250,150,4);
    b.ShowMember();
    system("pause");
}
```

课程内容

- 静态联编
- 虚函数
- 纯虚函数和抽象类
- 虚析构函数

虚析构函数

- 在析构函数前面加上关键字virtual进行声明，称该析构函数为虚析构函数
- 构造函数不能是虚函数
- 如果一个基类的析构被声明为虚析构函数，则它的派生类中的析构函数也是虚函数
- 声明虚析构函数的目的在于使用delete运算符删除一个对象时，能确保析构函数被正确的执行。因为设置虚析构函数后，可以采用动态连编的方式选择析构函数

虚析构函数

例5.4虚析构函数

```
class ClxBase
{public:
    ClxBase() {cout<<"construct Base class!"<<endl;}

    virtual ~ClxBase() {cout<<"in destructor of class
                                ClxBase"<<endl;}

    virtual void DoSomething() { cout <<"Do something in
                                    class ClxBase!"<<endl;}
};
class ClxDerived : public ClxBase
{public:
    ClxDerived() {cout<<"construct Sub class!"<<endl;}

    ~ClxDerived() { cout<<"Output from the destructor of
                        class ClxDerived!"<<endl;}

    void DoSomething()  {cout<<"Do something in class
                                ClxDerived!"<<endl;}
};
```

虚析构函数

```
int main ( )
{
    ClxBase *pTest = new ClxDerived;
    pTest->DoSomething();
    delete pTest;
}
```

➢ 写出执行结果:

练习 1

凌阳教育
www.sunplusedu.com

值得信赖的教育品牌

Tel: 400-705-9680 , Email: edu@sunplusapp.com , BBS: bbs.sunplusedu.com