C#中的抽象类的功能与C++中的抽象类类似。

在C#中，使用abstract修饰符，可以声明抽象方法（类似于C++中的纯虚函数），没有方法体，同时，属性和索引器也可以是抽象的。

抽象方法不能声明为virtual（抽象方法默认为virtual），static方法不能为抽象方法（static方法不依赖对象，需要有实现，否则通过类名来访问时就会报错）。

包含一个或多个抽象方法的类必须声明为abstract类。

当派生类继承抽象类时，它必须实现全部的抽象方法、属性和索引器，否则也必须声明为abstract类，abstract特性将一直继承，直到全部的抽象方法、属性和索引器被全部实现。

派生类中实现abstract方法时，也必须使用override修饰符。

可以声明一个抽象类的引用，但不能创建一个抽象类的对象。

例：程序inherit\_class\_test4

// Copyright 2016.刘珅珅

// author：刘珅珅

// 抽象类

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace inherit\_class\_test4

{

abstract class TwoDShape

{

double width\_;

double heigth\_;

public TwoDShape(double w, double h)

{

Width = w;

Height = h;

}

public double Width

{

get { return width\_; }

set { width\_ = value < 0 ? -value : value; }

}

public double Height

{

get { return heigth\_; }

set { heigth\_ = value < 0 ? -value : value; }

}

public void ShowDim()

{

Console.WriteLine("Width and height are " + Width + " and " + Height);

}

// 抽象方法

public abstract double Area();

}

class Triangle : TwoDShape

{

string style\_;

public Triangle(string s, double w, double h) : base(w, h)

{

style\_ = s;

}

// 实现抽象方法

public override double Area()

{

return Width \* Height / 2;

}

}

class InheritClassTest

{

static void Main(string[] args)

{

// 抽象类不能定义对象，只能定义引用

TwoDShape[] shapes = new TwoDShape[2];

shapes[0] = new Triangle("right", 8.0, 12.0);

shapes[1] = new Triangle("left", 4.0, 5.0);

for (int i = 0; i < shapes.Length; ++i)

{

Console.WriteLine("Area is " + shapes[i].Area());

Console.WriteLine();

}

}

}

}