ctrl+z撤销

ctrl+s保存

ctrl+K+c注释当前选中代码

ctrl+K+U取消对所选代码的注释

单精度类型 值后面要加f ,小数点后面有七位。

前++ 先给这个变量自身加1，然后带着这个加1后的值去参与运算

后++ 先拿原值去参与运算，运算完毕后，再将这个变量自身加1

无论是哪种++，最终的结果都是给这个变量加1

一元运算符（++，--）的优先级要高于二元运算符，如果在一个表达式中既有一元运算符又有二元运算符，我们首先计算一元运算符。

\n 表示换行

\t 表示制表键，使得文字对齐

\” 表示半个双引号

\b 会把前面一个字符删掉

@ 用在字符串的前面有两种用途

1. 取消“\”在字符串中的转义作用，使其单纯的就表示一个斜线{多用于路径}。
2. 如果代码超过一行到另外一行可以加一个使其还是一个整体。

double - int

double -- float 强制类型转换

小的转大的 隐式类型转换

两份int类型的加加减减仍然是int类型

若其中有一个double类型则值变为double

类型不兼容 需要用convert进行转换

Convert.ToInt32();

Convert.ToDouble(）；

if else 语句

if()

{

}

else

{

if ()

{

}

}

switch语句

switch()

{

case1:

要执行的代码

break;

case2:

要执行的代码

break；

default:

要执行的代码；

break；

}

break语句 ：立刻结束循环体 多层循环只退出所在那一层循环

continue语句: 遇到立刻返回判断条件，若符合判断条件则继续循环，若不符合跳出循环

常量：一旦声明不可重新赋值

语法： const 数据类型 常量名（全大写）= 值

枚举类型：用来规范代码用法,把重要的字段放在枚举里面，谁用谁就从里面拿出来用

放在namespace下

语法：pubic enum 枚举名(首字母大写）

{

值1，

值2，

值n

}

使用枚举类型

枚举类型 变量名 = 枚举类型·值

结构体：可以声明多个不同类型的变量

声明结构体：

public struct 结构体名称（首字母大写）

{

public 数据类型 字段名；

public 数据类型 字段名；

public 数据类型 字段名；

（字段名一般以下划线开头）

}

数组：存储相同数据类型元素的集合

声明初始化二合一格式：

数据类型[] 数组名 = new 数据类型[数组长度]；

int[] number = new int[5];

int[] number ={0,1,2,3,4,5};

int[] c = new int[] {0,1,2,3,4,5};

null: 意思为根本没在内存中开辟空间

空：开了空间但是没有储存值

bool的初始值都为false

string的初始值都为null

方法的声明：

public static 返回值类型 函数名（[参数列表]）

{

函数代码体

}

声明全局变量：public static 返回值类型 \_字段名

out参数：

一个函数中如果要返回多个函数的值，那就需要

函数外可以不为变量赋值，函数内必须赋值

形参和实参前都必须加out关键字

ref参数：

将一个变量传入传入一个函数中进行处理，处理完成后，再将处理后的值带出函数。

函数的重载：对于不同类型的数据使用同一个函数，就把这个函数名不变，数据类型换下即可

类的语法：

[访问修饰符] class 类名

{

public 数据类型 字段名；

}

类中实例化一个对象：

类名 对象名 = new 类名（）；

当类中的字段数据的访问修饰符发生变化时，就不会不可访问，我们此时可以通过属性间接操作访问字段。

属性的语法

public 数据类型 属性名

{

get{return 字段名；}

set{字段名=value；}

}

数据类型：和保护的字段的数据类型一样

属性名：和字段名一样，只不过首字母要大写

get：我们从字段中国取值的时候调用GET方法

set :我们给字段赋值的时候调用set方法

value:系统关键字，代表赋给属性或字段的值

get和set本质上是两个函数，在其中仍然可以加判断条件

快捷键 ctrl+r+e

静态与非静态的区别：

1. 在一个非静态的类中，既可以有实例成员，也可以有静态成员。
2. 在调用实例成员时，需要使用对象名·实例成员
3. 在调用静态成员时，需要使用类名·静态成员

静态函数中只能访问静态成员

实例函数两者都能使用

静态类中只能有静态成员

构造函数：当重复输入某个格式时使用

public 名字（与当前类名一样）（参数类型 参数名）

{ this· 属性 =参数名

}

值先传给属性在传给字段

命名空间：可以看做是类的文件夹，使用类的时候，软件的最上方必须要出现的此类的来源

常用字符串方法：

1. ToUpper()

将字符转换成大写形式，仅对字母有效。

使用：字符串变量.方法名（）；

1. ToLower()

将字符转换成小写形式

同上

1. Equals()

作用：比较两个字符串是否相同。相同返回真，不同返回假。

使用：字符串变量.方法名（要比较的字符串变量）；

1. Split()

分割字符串。

使用：字符串变量.Split(用于分割的字符数组)；

例如：+abcd,.cdf 清除中间无用字符

（5）Subsrting()

作用：截取字符串，返回截取后的字符串。

使用：字符串变量.Substring(开始截取的位置)；

包括自己往后截取。

（2,2）第一个2表示从哪里开始截取，第二个2表示截取的长度

1. Contains()

作用：判断某个字符串中是否包含指定的字符串。 如果包含返回真，否则返回假。

使用：字符串变量. Contains( 子字符串);

1. Replace()

作用：将字符串的某个字符串替换成一个新的字符串。返回新的字符串。

使用：字符串变量.方法名（旧字符串，新字符串）；

（7）IndexOf()

作用：查找某个字符串在字符串中第一次出现的位置。

使用：字符串变量.IndexOf(子字符串)；

位置从0开始

1. LastIndexOf()

作用：查找某个字符串在字符串中最后一次出现的位置。

使用：字符串变量.LastIndexOf(子字符串)；

1. StartsWith()

作用：判断是否以某个字符串开始。如果是，返回真；否则返回假

使用：字符串变量.StartsWith(子字符串)；

1. EndsWith()

作用：判断是否以某个字符串结束。如果是，返回真；否则返回假

使用：字符串变量.EndsWith(子字符串)；

1. Trim()

作用：去掉字符串中前后空格。返回处理后的字符串。

使用：字符串变量.Trim();

1. TrimEnd()

作用：去掉字符串结束后的空格。返回处理后的字符串。

使用：字符串变量.TrimEnd()；

1. TrimStart()

作用：去掉字符串开始前的空格。返回处理后的字符串。

使用：字符串变量.TrimStart()；

（15）IsNullOrEmpty()

作用：判断一个字符串是否为Null或者空。如果为null或者空，返回真。否则返回假。

使用：String.IsNullOrEmpty(字符串变量)；

值类型的值存储在内存的栈中

引用类型的值存储在内存的堆中

值类型：int,double,bool,char,struct,enum

引用类型：string,数组

StringBuilder简介

StringBuilder是一个类，其类型的“字符串变量”，一直操作同一块内存空间，不会产生垃圾数据，且执行效率远远高于String类型的字符串变量。

使用：

1. 创建StringBuilder类型的变量

StringBuilder sb = new StringBuilder（）；

//创建一个对象

1. 往sb中追加数据

sb.Append(i);

sb.ToString();//将sb转成字符串形式

1. 清空sb中的数据

sb.Clear();

StringBuilder效率测试

1. Stopwatch类

Stopwatch，秒表计时器，用来记录程序运行的时间

1. 创建Stopwatch类型对象

Stopwatch sw = new Stopwatch();

sw.Start();//计时器开始

sw.Stop();//计时器结束

sw.Elapsed;//记录开始到结束的时长

继承：

一个子类只能有一个父类

语法格式：

class 子类：父类

{

}

子类继承父类到底继承了什么？

首先，子类继承了父类的属性和方法，但是子类并没有继承父类的私有字段

子类没有继承父类的构造函数，但是子类会默认调用父类无参数的构造函数。

如何调用父类的构造函数？

使用关键字:base()

里式转换原则

object类是所有类型的父类（基类）

1 子类对象可以直接赋值给父类

2 如果父类对象中装的是子类对象，可以将这个父类对象强转为子类对象：

father f= new Ming();

Ming s=(Ming)f;

3 子类对象可以调用父类的成员，但是父类对象永远只能调用自己的成员

is和as

if（f is Ming）

{

console.writeline(“转换成功”)；

}

else

{

console.writeline(“转换失败”)；

}

is返回true 或者false

as 返回对应的对象或者null

list泛型集合：省略了命名空间,长度可变

List<int>list=new List<int>();

之后加入的数组必须是INT类型的

什么是装箱？

把值类型转换为引用类型

（int，double，float） （string,自定义类，集合）

拆箱：

把引用类型转换为值类型

string str = “123”;

int n = convert.ToInt32(str);

看两种类型是否发生了拆装箱，要看这两种类型是否存在继承关系。如果存在继承关系则可能发生装箱或者拆箱，如没有则不可能发生。

面向对象开发有三大特性：封装，继承，多态

在继承关系的前提下，实例化不同的对象，这些对象调用相同的方法却表现出不同的行为这就叫多态。

在c#语言中体现多态有三种方式：虚方法，抽象类，接口

当父类的方法与子类的方法相同时，子类继承父类只能用父类的方法，但是如果子类坚持用自己的方法

实现： 在父类方法前加virtual

在子类方法前加override

父类： public virtual 返回值类型 方法名（）

{

}

子类：public override 返回值类型 方法名（）

{

}

父类里面用关键字virtual关键字修饰的方法叫做虚方法，子类可以使用 override 重写该虚方法，也可以不重写。

虚方法还是有方法体的，当我们父类的这个方法无法确定方法体的时候，就可以使用另外一种形式来表现，这种形式叫抽象方法

抽象方法的返回值类型前用关键字 abstract 修饰，且无方法体。抽象方法必须存在于抽象类中

抽象类语法：

在定义类的关键字class前面加abstract修饰的类就是抽象类，子类继承抽象类，使用override关键字重写父类中所有的抽象方法

使用方法：

当父类中的方法不知道如何实现的时候，可以考虑将父类写成抽象类，将方法写成抽象方法

访问修饰符

public：公开的，公共的

private：私有的，只能在当前类的内部访问

protected：受保护的，只能在当前类的内部以及该类的子类中访问

internal:只能在当前项目中访问，在同一个项目中，相当于public

平常新建的一个类 它的访问权限默认是internal

能够修饰类的访问修饰符只有两个：public，internal

接口:

接口中只能写没有方法体的函数，该方法可以有返回值但不能添加访问修饰符。接口不可以实例化

语法格式要求：

接口使用interface关键字定义，没有class关键字，接口名一般以I开头这种方式进行书写，在一堆脚本中通过名字判断，I开头的都是接口。

一个类可以继承多个接口，接口也可以继承接口，但是类继承接口时，必须要把接口中的方法全部实现

什么时候用抽象类来实现多态？

如果几个类中，你能提取出来一个父类并且父类当中必须写上几个子类共有的方法然后你还不知道怎么写这个方法。那就是使用抽象类

什么时候使用虚方法 来实现多态

如果这个父类的方法你知道怎么写并且还需要创建这个父类的对象那就用虚方法实现

什么时候用接口来实现多态？

如果这几个类中根本就提取不出来父类，但是这几个类都有一个共同的行为或能力。那就用接口实现