控制台输入一个数 X 求 1(2)+2(2)+3(2)+...X（2）的结果

```
        public int Fun(int num)
        {
            if (num == 1)
                return 1;
            else
            {
                int result;
                result=Fun(num - 1) + num * num;
                return result;
            }
        }
```

递归实现：

一个猴子摘了一堆桃子 当即吃了一半 觉得不过瘾 又多吃了一个

第二天 猴子吃了第一天剩下桃子的一半 再多吃一个

以后每天吃前一天剩下的桃子的一半 多一个

到了第10天 发现就只有1个桃子了

问 第一天摘了多少个桃子

公式：(x+1)*2

```
        public int Fun(int day)
        {
            int count;
            if (day == 10)
                return 1;
            else
            {
                count = (Fun(day + 1) + 1) * 2;
                return count;
            }
        }
```

万物皆对象

世界是由什么组成的：

动物 植物 物品 名胜古迹等

分类是认识世界的一个自然过程 在日常生活中会不直觉的进行分类

类是一个抽象的概念 仅仅是模板 比如 人类

对象是一个可以看得见也摸得着的实体

我们每一个人都是一个人类的对象

每一个类 都具备特征和行为

例如：

人类

{

特征(属性)：

年龄 姓名 体重 身高等

行为(方法)：

衣食住行

}

所以说 类是模子 确定了对象将会拥有的特征和行为

类的成员 包括成员变量和成员方法

成员变量包含：

常量,字段

成员方法包括：

方法(函数) 属性 事件 索引器 构造方法和析构方法

例：

在一个超市中 可以大致分为两类人

1 顾客

属性：姓名 年龄 体重 （ 变量 ）

行为：购买商品 付款 （ 方法 ）

2 收银员

属性： 员工号 姓名 部门（ 变量 ）

行为：收款 打印账单 开发票 刷卡(方法)

类 Class

类的定义

```
    //学校类
    class School
    {
        /// <summary>
        /// 学校类的成员变量
        /// </summary>
        //学校名称
        public string name;
        //学校里教室的数目
        public int classNum;
        //学校的学生数量
        public int stuNum;


        //学校类的成员方法
        //展示信息
        public void Show()
        {
            Console.WriteLine(name + "学校共有" + classNum + "个教室,共有" + stuNum + "个学生");
        }

```

```
    }
```

创建对象 调用成员

```
        static void Main(string[] args)
        {
            //实例化对象  s是对象名 创建对象关键字是 new
            School s=new School();

            //获取成员变量并赋值
            s.name = "博思中国";
            s.classNum = 8;
            s.stuNum = 100;

            //调用成员方法
            s.Show();
        }
```

在一个类中 成员变量不赋值也可以使用

不同的数据类型拥有不同的默认值

引用类型：数组 string 默认值null

```
          School s=new School();
            School s2 = null;

            if (s != null)
            {
                Console.WriteLine("s不为空");
            }
            if (s2 == null)
            {
                Console.WriteLine("s2为空");
            }

```

```
            //使用引用数据类型对象前 先判定是否为null 否则使用null对象 报错
             if(s.array!=null)
                 Console.WriteLine(s.array.Length);

             Console.WriteLine(s.name);
```

类的访问权限

public 公共的 共有的 外部可以通过对象名.来访问 (最高的访问权限)

private 私有的 不允许外部访问 值允许自己内部访问 （ 最低的访问权限 ）

protected 保护的 只允许自己和自己的子类进行访问

internal 声明内部成员 内部成员只有在一个程序集中的文件才能访问

类中的成员 在没有标识访问权限时 默认是Private

C#是面向对象的程序设计语言

面向对象的第一个特征就是封装

好处：

1 把人类的思维转化为程序可以理解的过程

2 信息隐藏 提高程序的可维护性和安全性

3 提高了程序的重用性 提高开发效率

例：

创新游人类 包含姓名和年龄属性

控制台输入游人的属性 根据年龄判断购买门票的价格

```
    class visitor
    {
        public string name;
        public int age;
    }

```

```
7
8         static void Main(string[] args)
9         {
10            while (true)
11            {
12                visitor v = new visitor();
13                Console.WriteLine("请输入游客的姓名");
14                v.name = Console.ReadLine();
15
16                Console.WriteLine("请输入游客的年龄");
17                v.age = int.Parse(Console.ReadLine());
18
19                if (v.age < 18 || v.age > 70)
20                    Console.WriteLine("免费入园");
21                else
22                    Console.WriteLine("门票价格20元");
23            }
24        }
```

创建Admin类 包含用户名和密码两个成员变量

实例化对象 并给 成员变量赋值

实现密码修改的功能

先输入旧密码

重复输入两次新密码

如果新密码两次一致 修改成功

```
1     class Admin
2     {
3         public string username;
4         public string password;
5     }
6
7         static void Main(string[] args)
8         {
9             Admin admin = new Admin();
10            admin.username = "user";
```

```
            admin.password = "123456";

            while (true)
            {
                Console.WriteLine("请输入密码");
                string s = Console.ReadLine();

                if (s.Equals(admin.password))
                {
                    Console.WriteLine("请输入新密码");
                    string b = Console.ReadLine();
                    Console.WriteLine("请再次输入新密码");
                    if (b.Equals(Console.ReadLine()))
                    {
                        Console.WriteLine("两次结果输入一致 密码修改成功");
                        admin.password = b;
                        break;
                    }
                    else
                    {
                        Console.WriteLine("两次密码输入不一样 修改失败");
                    }
                }
                else
                    Console.WriteLine("密码错误");
            }
        }
```

类的构造方法

类的构造方法 是一个特殊的类的成员 当创建类的对象时执行(出生方法)

注意：构造方法的方法名和类名相同 不能写返回值 即使是Void也不能写

```
    class Admin
    {
        public string username;
        public string password;
```

```
5
6           //这是默认构造方法
7           public Admin()
8           {
9               username = "user";
10              password = "123456";
11          }
12      }
13
14      class Program
15      {
16          //Main方法 程序入口 代码需要在Main方法中实现
17          static void Main(string[] args)
18          {
19              //new Admin()调用默认构造方法
20              Admin admin = new Admin();
21              Console.WriteLine(admin.password);
22          }
23      }
```

构造方法存在的意义：

用于给成员变量赋初值(完成初始化)

很多在实例化对象时就需要赋值的数据 可以写在构造方法里

构造方法重载 参数化构造方法

```
1       class Admin
2       {
3           public string username;
4           public string password;
5
6           //这是默认构造方法
7           public Admin()
8           {
9               username = "user";
10              password = "123456";
11          }
12
```

```
        //重载构造方法 参数列表可以随意定义 但是依然不能写void 方法名和类名相同
        public Admin(string name,string word)
        {
            username = name;
            password = word;
        }
        public Admin(string name)
        {
            username = name;
            password = "11111";
        }
    }

    class Program
    {
        //Main方法 程序入口 代码需要在Main方法中实现
        static void Main(string[] args)
        {
            //new Admin()调用默认构造方法
            Admin admin1 = new Admin();
            Admin admin2 = new Admin("用户2","game2");
            Admin admin3 = new Admin("用户3");
            Console.WriteLine(admin3.password);
        }
    }
```

析构方法

和构造方法正好相反 是死亡方法 在对象被释放时调用

不能加public访问权限 ~类名（ ）

开发中使用的很少 了解即可

```

        ~Admin()
        {
            Console.WriteLine("我被释放了");
        }
```

类的this关键字用法

当类中的成员变量和形参的变量名相同时

参数的优先级高于成员变量

区分成员变量和参数 使用this

```
class Admin
    {
        public string username;
        public string password;
        public int x;

        //这是默认构造方法
        public Admin()
        {
            username = "user";
            password = "123456";
        }

        //重载构造方法 参数列表可以随意定义 但是依然不能写void 方法名和类名相同
        public Admin(string username,string password)
        {
            //username=username  不能使用 都是代表参数
            //this 代表 当前调用此方法的实例对象 所以this.username是对象的成员
            this.username = username;
            this.password = password;
        }

        public void fn(int x)
        {
            x = 100;
            this.x = 100;
        }
    }
    class Program
    {
        //Main方法 程序入口 代码需要在Main方法中实现
```

```
        static void Main(string[] args)
        {
            Admin a = new Admin("game","sss");
            int b=0;
            a.fn(b);
        }
    }
```

关联构造方法

```
    class Admin
    {
        public string username;
        public string password;

        //这是默认构造方法
        public Admin()
        {
            Console.WriteLine("默认构造");
        }

        //this关联构造方法  先调用默认构造方法 后调用 有参构造方法
        public Admin(string username,string password):this()
        {
            Console.WriteLine("有参构造方法");
        }
    }
```

练习：

编写一个矩形类（Rect）,私有数据成员为矩形的长(length)和宽(width)

无参构造函数奖length和width设置为0 有参构造函数传参赋值长和宽的值

另外 类还包括：求矩形的周长 ，面积 ，取长度，取宽度，修改矩形长度和宽度的公有方法