# 变量类型

int double string char bool decimal

* 变量的使用规则：先声明再赋值最后使用

int number;

number=10;

number=20;//直接赋值

Console.WriteLine(number);

# Camel Pascal

命名方式

# 运算符

* 赋值运算符：=
* 复合赋值运算符：+= -= \*= /= %= sum+=age; sum=sum+age;
* 算数运算符： + - \* / % ++ --
* 关系运算符： > < >= <= == !=
* 逻辑运算符： &&（全真为真） ||（一真为真） !

# C#中的语法结构

* 分支结构：
* if if-else
* 选择结构：
* while do-while for（知道次数）

# 常量

* 声明的常量的语法：

const 变量类型 变量名=值;

* 什么时候会用到常量？

不想变的东西

|  |
| --- |
| int number = 10;  number = 20;//体现了变量可以重新赋值  const int number2 = 20;//常量 不能被重新赋值 |

# 枚举

* 语法：

[public] enum 枚举名

{

值1,

值2,

值3,

........

}

* public：访问修饰符。公开的公共的，哪都可以访问。
* enum：关键字，声明枚举的关键字
* 枚举名：要符合Pascal命名规范
* 和用字符串比起来，用枚举的好处就是限定了变量的取值范围，程序处理起来更方便。
* 将枚举声明到命名空间的下面，类的外面，表示这个命名空间下，所有的类都可以使用这个枚举。
* 枚举就是一个变量类型 ，int--double string decimal.
* 只是枚举声明、赋值、使用的方式跟那些普通的变量类型不一样。

|  |
| --- |
| //将枚举声明到命名空间的下面，类的外面，表示这个命名空间下  //所有的类都可以使用这个枚举。  //语法：  //[public] enum 枚举名  // {  // 值1,  // 值2,  // 值3,  // ........  //}  //声明了一个枚举Gender  public enum Gender  {  男,  女  }  public enum Season  {  春,  夏,  秋,  冬  }  public enum QQstate  {  Online,  Offline,  Leave,  Busy,  Qme  }  class Program  {  static void Main(string[] args)  {  #region 导言  //为什么会有枚举这个东西  //xx大学管理系统  //姓名 性别 年龄 系别 年级  //特别是性别，写性别的时候有很多写法  //枚举可以规范代码  //char gender = '男';  //string s1 = "female";  //string ss1 = "爷们";  #endregion  #region 举例  ////变量类型 变量名=值  //int n = 10;  //Gender gender = Gender.男;  #endregion  #region 练习  Season s = Season.夏;  QQstate state = QQstate.Offline;  #endregion |

* 我们可以将一个枚举类型的变量跟int类型和string类型互相转换。
* 枚举类型默认是跟int类型相互兼容的，所以可以通过强制类型转换的语法互相转换。
* 当转换一个枚举中没有的值的时候，不会抛异常，而是直接将数字显示出来。
* 枚举同样也可以跟string类型互相转换，如果将枚举类型转换成string类型，则直接调用ToString().
* 如果将字符串转换成枚举类型则需要下面这样一行代码：

(要转换的枚举类型)Enum.Parse(typeof(要转换的枚举类型),"要转换的字符串");

* 如果转换的字符串是数字，则就算枚举中没有，也会不会抛异常。
* 如果转换的字符串是文本，如果枚举中没有，则会抛出异常。
* 所有的类型都能够转换成string类型，调用ToString()。

|  |
| --- |
| public enum QQstate  {  OnLine = 1,  OffLine,  Leave,  Busy,  QMe  }  class Program  {  static void Main(string[] args)  {  #region 将枚举类型强转成int类型  QQstate state = QQstate.OnLine;  //枚举类型默认可以跟int类型相互转换 枚举类型跟int类型是兼容的  int n = (int)state;  Console.WriteLine(n);  //输出是0，说明枚举是从0开始的  Console.WriteLine((int)QQstate.OffLine);  Console.WriteLine((int)QQstate.Busy);  #endregion  #region 将int类型强转为枚举类型  //int n1 = 3;  int n2 = 8;  //输出3号busy  //QQstate state = (QQstate)n1;  //但是枚举中没有8号，不会报错，直接输出原来的值  QQstate state = (QQstate)n2;  Console.WriteLine(state);  #endregion  #region 将枚举类型转换成字符串类型  #region 所以的类型都可以转成string类型  //所以的类型都可以转成string类型  int n1 = 10;  double n2 = 3.14;  decimal n3 = 5000m;  string s1 = n1.ToString();  string s2 = n2.ToString();  string s3 = n3.ToString();  Console.WriteLine(s1);  Console.WriteLine(s2);  Console.WriteLine(s3);  #endregion  QQstate state = QQstate.OnLine;  string s = state.ToString();  //输出OnLine  Console.WriteLine(s);  #endregion  #region 将字符串类型转化为枚举类型  //如果枚举中没有，则会抛出异常  string s = "0";  //将s转化为枚举类型 不一样  //string转int的方法  //Convert.ToInt32 int.Parse() int.TryParse()  //调用Parse()方法的的就是为了让它们帮助我们把一个字符串转换为对应的枚举类型  //首先需要获得枚举的类型  QQstate state = (QQstate)Enum.Parse(typeof(QQstate), s);  Console.WriteLine(state);  #endregion  #region 枚举类型  //提示用户选择一个在线状态，我们接受，并将用户输入类型转换成枚举类型  //再次打印到控制台中  //默认第一个是0 若要第一个是1则要在前赋1  Console.WriteLine("请选择你的QQ在线状态\n1--OnLine 2--OffLine 3--Leave 4--Busy 5--QMe");  string input = Console.ReadLine();  //多条件的定值判断  switch (input)  {  case "1":  QQstate s1 = (QQstate)Enum.Parse(typeof(QQstate), input);  Console.WriteLine("您选择的在线状态是{0}", s1);  break;  case "2":  QQstate s2 = (QQstate)Enum.Parse(typeof(QQstate), input);  Console.WriteLine("您选择的在线状态是{0}", s2);  break;  case "3":  QQstate s3 = (QQstate)Enum.Parse(typeof(QQstate), input);  Console.WriteLine("您选择的在线状态是{0}", s3);  break;  case "4":  QQstate s4 = (QQstate)Enum.Parse(typeof(QQstate), input);  Console.WriteLine("您选择的在线状态是{0}", s4);  break;  case "5":  QQstate s5 = (QQstate)Enum.Parse(typeof(QQstate), input);  Console.WriteLine("您选择的在线状态是{0}", s5);  break;  }  #endregion  Console.ReadKey();  }  } |

# 结构

* 可以帮助我们一次性声明多个不同类型的变量。
* 语法：

[public] struct 结构名

{

成员;//字段

}

变量在程序运行期间只能存储一个值，而字段可以存储多个值。

|  |
| --- |
| public struct Person  {  //其他位置读不到要加public  public string \_name;//字段 区别变量 变量只能存一个 字段可以存很多  public int \_age;//区别变量前面加\_  public Gender \_gender;//变成枚举  }  public enum Gender  {  男,  女  }  class Program  {  static void Main(string[] args)  {  ////xx大学管理系统  ////姓名、年龄、性别、系别、年级  ////如果人很多这些变量会很多  //string zsName = "张三";  //int zsAge = 21;  //char zsGender = '男';  //int zsGrade = 3;  Person zsPerson;  zsPerson.\_name = "张三";  zsPerson.\_age = 21;  zsPerson.\_gender = Gender.男;  Person lsPerson;  lsPerson.\_name = "李四";  lsPerson.\_age = 22;  lsPerson.\_gender = Gender.女;  Console.WriteLine(zsPerson.\_name);  Console.WriteLine(lsPerson.\_name);  Console.ReadKey();  }  } |

练习：

1、定义一个结构叫MyColor,有三个成员,分别定义为int类型的red,green,blue

声明一个 MyColor类型的变量,并对其成员赋值.使MyColor可以表示成一个红色.

2、定义一个结构类型Person,有三个成员,分别为姓名,性别,年龄 性别用枚举类型

声明两个Person类型的变量,分别表示 张三 男 18岁/ 小兰 女 16岁

|  |
| --- |
| public struct MyColor  {  public int \_red;  public int \_green;  public int \_bule;  }  public enum Gender  {  男,  女  }  public struct Person  {  public string \_name;  public int \_age;  public Gender \_gender;  }  class Program  {  static void Main(string[] args)  {  //1、定义一个结构叫MyColor,有三个成员,分别定义为int类型的red,green,blue  //声明一个 MyColor类型的变量, 并对其成员赋值.使MyColor可以表示成一个红色.  MyColor mc;//红色  mc.\_red = 255;  mc.\_green = 0;  mc.\_bule = 0;  //2、定义一个结构类型Person,有三个成员,分别为姓名,性别,年龄 性别用枚举类型  //声明两个Person类型的变量,分别表示 张三 男 18岁 / 小兰 女 16岁  Person zsPerson;  zsPerson.\_name = "张三";  zsPerson.\_age = 18;  zsPerson.\_gender = Gender.男;  Person xlPerson;  xlPerson.\_name = "小兰";  xlPerson.\_age = 16;  xlPerson.\_gender = Gender.女;  Console.WriteLine("我叫{0}，我今年{1}岁，我是{2}生", xlPerson.\_name, xlPerson.\_age, xlPerson.\_gender);  Console.WriteLine("我叫{0}，我今年{1}岁，我是{2}生", zsPerson.\_name, zsPerson.\_age, zsPerson.\_gender);  Console.ReadKey();  }  } |

# 数组

一次性存储多个相同类型的变量。

语法：

数组类型[] 数组名=new 数组类型[数组长度];

int[] nums = new int[10];

* 当你写了上面这样的代码之后，就在内存中开辟了连续的10块空间，我们替每一个块称为这个数组的元素，初始是0
* 如果想访问到数组中的某一块元素，需要通过这个元素的下标或者索引去访问，从0开始

\*\*\*数组的长度一旦固定了，就不能再被改变了

|  |
| --- |
| //数组类型[] 数组名=new 数组类型[数组长度];  //初值0  int[] nums = new int[10];  //初值null 没有开空间 ""空表示开空间了但是里边是空的  string[] str = new string[10];  //初值false  bool[] bools = new bool[10];  //数组的赋值方式  int[] numsTwo = { 1, 2, 3, 4, 5, 6 };  //不推荐  int[] numsThree = new int[3] { 1, 2, 3 };  int[] numFour = new int[] { 1, 2, 3, 4, 5 };  //给数组赋值 一个一个很麻烦 用for  nums[0] = 1;  nums[1] = 2;  nums[2] = 3;  nums[3] = 4;  nums[4] = 5;  nums[5] = 6;  nums[6] = 10;  nums[7] = 8;  nums[8] = 9;  nums[9] = 10;  //nums[10] = 11; 报错溢出了  //我们通过一个循环个数组赋值，也通过一个循环对一个数组进行取值  for (int i = 0; i < nums.Length; i++)  {  nums[i] = i;  }  for (int i = 0; i < nums.Length; i++)  {  Console.WriteLine(nums[i]);  }  Console.ReadKey(); |

练习1：从一个整数数组中取出最大的整数,最小整数,总和,平均值

练习2：计算一个整数数组的所有元素的和。

练习3:数组里面都是人的名字,分割成:例如:老杨|老苏|老邹…”

(老杨,老苏,老邹,老虎,老牛,老蒋,老王,老马)

练习4：将一个整数数组的每一个元素进行如下的处理：如果元素是正数则将这个位置的元素的值加1，如果元素是负数则将这个位置的元素的值减1,如果元素是0,则不变。

练习5：将一个字符串数组的元素的顺序进行反转。{“我”,“是”,”好人”} {“好人”,”是”,”我”}。第i个和第length-i-1个进行交换。

|  |
| --- |
| #region 示例  //数组类型[] 数组名=new 数组类型[数组长度];  //初值0  int[] nums = new int[10];  //初值null 没有开空间 ""空表示开空间了但是里边是空的  string[] str = new string[10];  //初值false  bool[] bools = new bool[10];  //数组的赋值方式  int[] numsTwo = { 1, 2, 3, 4, 5, 6 };  //不推荐  int[] numsThree = new int[3] { 1, 2, 3 };  int[] numFour = new int[] { 1, 2, 3, 4, 5 };  //给数组赋值 一个一个很麻烦 用for  nums[0] = 1;  nums[1] = 2;  nums[2] = 3;  nums[3] = 4;  nums[4] = 5;  nums[5] = 6;  nums[6] = 10;  nums[7] = 8;  nums[8] = 9;  nums[9] = 10;  //nums[10] = 11; 报错溢出了  //我们通过一个循环个数组赋值，也通过一个循环对一个数组进行取值  for (int i = 0; i < nums.Length; i++)  {  nums[i] = i;  }  for (int i = 0; i < nums.Length; i++)  {  Console.WriteLine(nums[i]);  }  #endregion  #region 练习1  //练习1：从一个整数数组中取出最大的整数,最小整数,总和,平均值  //声明一个int类型的数组 并且随意的赋值  int[] nums = { 1, -2, 3, 4, -5, 6, -7, 8, 9, 0 };  //声明两个变量用来存储最大值和最小值  ////还可以这样写(不太好用)  //int max = int.MinValue;  //int min = int.MaxValue;  int max = nums[0];//最大值也是  int min = nums[0];//但是当原数组当中没有0的时候它不成立，会导致最小值是0，所以赋一个数组的数字给它  int sum = 0;  //循环让数组的每个元素和我的最大值最小值进行比较  for (int i = 0; i < nums.Length; i++)  {  //关于在循环中nums[i]的理解方式  //1、代表数组中当前循环到的元素  //2、代表数组中的每个元素  //如果数组中当前循环的这个元素比我的max还要大，则把当前这个元素赋值给我的max  if (nums[i] > max)  {  max = nums[i];  }  if (nums[i] < min)  {  min = nums[i];  }  sum += nums[i];  }  Console.WriteLine("这个数组的最大值是{0}，最小值{1}，总和是{2}，平均值{3}", max, min, sum, sum / nums.Length);  #endregion  #region 练习3  //练习3:数组里面都是人的名字,分割成:例如:老杨|老苏|老邹…”  //(老杨, 老苏, 老邹, 老虎, 老牛, 老蒋, 老王, 老马)  string[] names = { "老杨", "老苏", "老邹", "老虎", "老牛", "老马" };  //老杨|老苏|老邹…  //思路：通过循环，获得字符串数组中的每一个元素  //然后，将这个每个元素加到一个字符串，以|分割  string str = null;  for (int i = 0; i < names.Length - 1; i++)  {  //str += names[i] + "|";//但是老马后面还有|，少循环一次  str += names[i] + "|";  }  Console.WriteLine(str + names[names.Length - 1]);  #endregion  #region 练习4  //练习4：将一个整数数组的每一个元素进行如下的处理：  //如果元素是正数则将这个位置的元素的值加1，如果元素是负数则将这个位置的元素的值减1,  //如果元素是0,则不变。  int[] nums = { 1, -2, 3, -4, 5, 6, 0 };  //思路：循环获取每个数  //对每个数进行判断  for (int i = 0; i < nums.Length; i++)  {  if (nums[i] > 0)  {  nums[i] += 1;  }  else if (nums[i] < 0)  {  nums[i] -= 1;  }  }  for (int i = 0; i < nums.Length; i++)  {  Console.WriteLine(nums[i]);  }  #endregion  #region 练习5  //练习5：将一个字符串数组的元素的顺序进行反转。  //{“我”,“是”,”好人”} {“好人”,”是”,”我”}。第i个和第length-i-1个进行交换。  string[] names = { "我", "是", "好人" };  //第一个和最后一个换位置  //5个元素换2次，6个元素换3次，n个元素化n/2  //1、0和5换 i=0 length-1-0  //2、1和4换 i=1 length-1-1  //3、2和3换 i=2 length-1-2  for (int i = 0; i < names.Length/2; i++)  {  string temp = names[i];  names[i] = names[names.Length - 1 - i];  names[names.Length - 1 - i] = temp;  }  for (int i = 0; i < names.Length; i++)  {  Console.WriteLine(names[i]);  }  #endregion |

# 冒泡排序

* 就是将一个数组中的元素按照从大到小或者从小到大的顺序进行排列。

int[] nums={9,8,7,6,5,4,3,2,1,0}; 0 1 2 3 4 5 6 7 8 9

第一趟比较：8 7 6 5 4 3 2 1 0 9 交换了9次 i=0 j=nums.Length-1-i

第二趟比较：7 6 5 4 3 2 1 0 8 9 交换了8次 i=1 j=nums.Length-1-i

第三趟比较：6 5 4 3 2 1 0 7 8 9 交换了7次 i=2 j=nums.Length-1-i

第四趟比较：5 4 3 2 1 0 6 7 8 9 交换了6次 i=3 j=nums.Length-1-i

第五趟比较：4 3 2 1 0 5 6 7 8 9 交换了5次

第六趟比较：3 2 1 0 4 5 6 7 8 9 交换了4次

第七趟比较：2 1 0 3 4 5 6 7 8 9 交换了3次

第八趟比较：1 0 2 3 4 5 6 7 8 9 交换了2次

第九趟比较：0 1 2 3 4 5 6 7 8 9 交换了1次

for(int i=0;i<number.Length-1;i++)

{

for(int j=0;j<nums.Length-1-i;j++)

{

if(nums[j]>nums[j+1])

{

int temp=nums[j];

nums[j]=nums[j+1];

nums[j+1]=temp;

}

}

}

# 方法

* 函数就是将一堆代码进行重用的一种机制。函数就是一段代码，这段代码可能有输入的值（参数），可能会返回值。一个函数就像一个专门做这件事的人，我们调用它来做一些事情，它可能需要我们提供一些数据给它，它执行完成后可能会有一些执行结果给我们。要求的数据就叫参数，返回的执行结果就是返回值。
* string s=Console.ReadLine()就是一个有返回结果的函数；Console.WriteLine("hello")就是一个有执行参数的函数，只有告诉WriteLine被打印的数据它才知道如何打印；int i=Convert.ToInt32("22")则是一个既有参数又有返回值的函数。
* 有了函数写代码就像拼积木，C#中的各种各样的技术其实就是通过for、if等这些基础的语法将不同的函数按照一定的逻辑组织起来。
* 函数就是将一堆代码进行重用的一种机制。
* 函数的语法：

[public] static 返回值类型 方法名([参数列表])

{

方法体；

}

* public:访问修饰符，公开的，公共的，哪都可以访问。
* static：静态的
* 返回值类型：如果不需要写返回值，写void
* 方法名：Pascal 每个单词的首字母都大些。其余字母小写
* 参数列表：完成这个方法所必须要提供给这个方法的条件。如果没有参数，小括号也不能省略。
* 方法写好后，如果想要被执行，必须要在Main()函数中调用。
* 方法的调用语法：

类名.方法名([参数]);

* \*\*\*在某些情况下，类名是可以省略的，如果你写的方法跟Main()函数同在一个类中，这个时候，

类名可以省略。

|  |
| --- |
| class Program  {  static void Main(string[] args)  {  #region 概念  //模拟代码 理解方法的概念  Console.WriteLine("超级玛丽走啊走，跳啊跳，顶啊顶");  Console.WriteLine("超级玛丽走啊走，跳啊跳，顶啊顶");  Console.WriteLine("超级玛丽走啊走，跳啊跳，顶啊顶");  Console.WriteLine("超级玛丽走啊走，跳啊跳，顶啊顶");  Console.WriteLine("顶到了无敌");//到无敌的时候代码就会重复 引入方法  Console.WriteLine("屏幕闪烁");  Console.WriteLine("播放无敌音效");  Console.WriteLine("屏幕停止");  Console.WriteLine("超级玛丽走啊走，跳啊跳，顶啊顶");  Console.WriteLine("超级玛丽走啊走，跳啊跳，顶啊顶");  Console.WriteLine("顶到了无敌");  Console.WriteLine("屏幕闪烁");  Console.WriteLine("播放无敌音效");  Console.WriteLine("屏幕停止");  #endregion  #region 调用最大值方法  //计算两个整数的最大值  int max = Program.GetMax(1, 3);  Console.WriteLine(max);  Console.ReadKey();  #endregion  #region 示例  Program.PlayGame();  Program.invincible();  Program.PlayGame();  Console.ReadKey();  #endregion  }  /// <summary>  /// 计算两个整数之间最大值并且将最大值返回  /// </summary>  /// <param name="n1">第一个整数</param>  /// <param name="n2">第二个整数</param>  /// <returns>将最大值返回</returns>  public static int GetMax(int n1, int n2)  {  return n1 > n2 ? n1 : n2;  }    /// <summary>  /// 正常玩游戏  /// </summary>  public static void PlayGame()  {  Console.WriteLine("超级玛丽走啊走，跳啊跳，顶啊顶");  Console.WriteLine("超级玛丽走啊走，跳啊跳，顶啊顶");  Console.WriteLine("超级玛丽走啊走，跳啊跳，顶啊顶");  Console.WriteLine("超级玛丽走啊走，跳啊跳，顶啊顶");  }    /// <summary>  /// 无敌  /// </summary>  public static void invincible()  {  Console.WriteLine("顶到了无敌");  Console.WriteLine("屏幕闪烁");  Console.WriteLine("播放无敌音效");  Console.WriteLine("屏幕停止");  }  } |

# return

1、在方法中返回要返回的值。

2、立即结束本次方法。

|  |
| --- |
| static void Main(string[] args)  {  while (true)  {  Console.WriteLine("Hello World");  //break; //跳出  //continue; //死循环  return;//直接跳出这个方法 下面不执行  }  Console.WriteLine("Hello .Net");  Console.ReadKey();  } |