楚广明C#简明教程

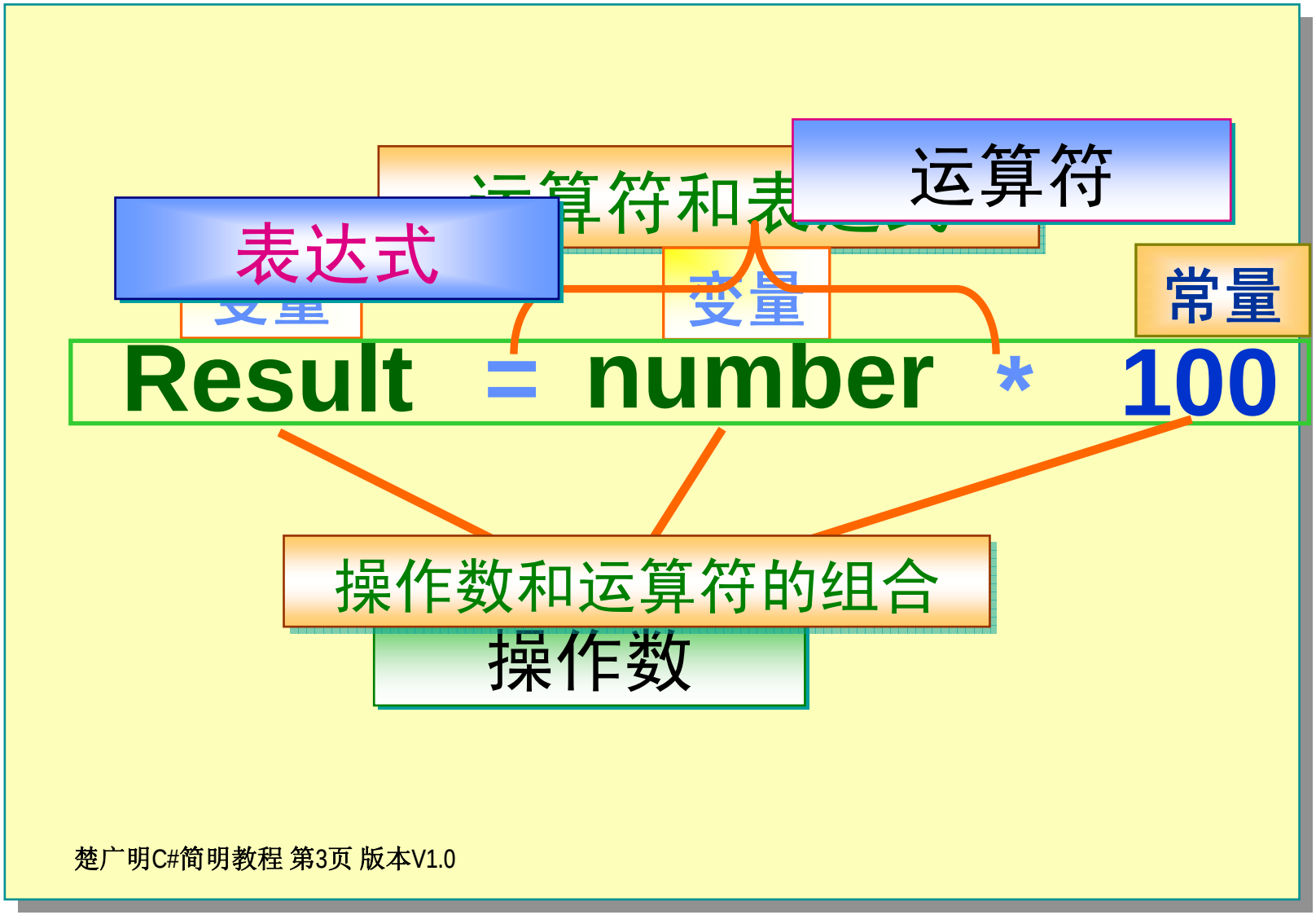
Email:chu888chu888@Gmail.com

Blog:http://www.cnblogs.com/chu888chu888

第四节 表达式

- 操作符
- 算术表达式
- 自增与自减表达式
- 位运算符
- 赋值表达式
- 关系表达式
- 条件逻辑表达式
- 其他特殊表达式

运算符和表达式



运算符

1. 一元运算符。一元运算符带一个操作数并使用前缀表示法（如-x）或后缀表示法（如x++）。
2. 二元运算符。二元运算符带两个操作数。并且全都使用中缀表示法（如x+ y）。
3. 三元运算符。只有一个三元运算符?:存在，它带三个操作数并使用中缀表示法（c? x: y)。

算术运算符

类别	运算符	说明	表达式
算术运算符	+	执行加法运算（如果两个操作数是字符串，则该运算符用作字符串连接运算符，将一个字符串添加到另一个字符串的末尾）	操作数1 + 操作数2
	-	执行减法运算	操作数1 - 操作数2
	*	执行乘法运算	操作数1 * 操作数2
	/	执行除法运算	操作数1 / 操作数2
	%	获得进行除法运算后的余数	操作数1 % 操作数2
	++	将操作数加 1	操作数++ 或++操作数
	--	将操作数减 1	操作数-- 或--操作数
	~	将一个数按位取反	~操作数

运算符和表达式

赋值运算符 (=)

变量 = 表达式;

例如：

身高 = 177.5;
体重 = 78;
性别 = “m”;

运算符和表达式

一元运算符（++/--）

Variable ++;

相当于

Variable = Variable + 1;

Variable --;

相当于

Variable = Variable - 1;

运算符和表达式

运算符	计算方法	表达式	求值	结果（假定 X = 10）
+=	运算结果 = 操作数1 + 操作数2	X += 5	X = X + 5	15
-=	运算结果 = 操作数1 - 操作数2	X -= 5	X = X - 5	5
*=	运算结果 = 操作数1 * 操作数2	X *= 5	X = X * 5	50
/=	运算结果 = 操作数1 / 操作数2	X /= 5	X = X / 5	2
%=	运算结果 = 操作数1 % 操作数2	X %= 5	X = X % 5	0

运算符和表达式

类别	运算符	说明	表达式
比较运算符	>	检查一个数是否大于另一个数	操作数1 > 操作数2
	<	检查一个数是否小于另一个数	操作数1 < 操作数2
	>=	检查一个数是否大于或等于另一个数	操作数1 >= 操作数2
	<=	检查一个数是否小于或等于另一个数	操作数1 <= 操作数2
	==	检查两个值是否相等	操作数1 == 操作数2
	!=	检查两个值是否不相等	操作数1 != 操作数2

运算符和表达式

类别	运算符	说明	表达式
成员访问运算符	.	**用于访问数据结构的成员**	数据结构.成员
赋值运算符	=	**给变量赋值**	操作数1 = 操作数2
逻辑运算符	&&	**对两个表达式执行逻辑“与”运算**	操作数1 && 操作数2
	||	**对两个表达式执行逻辑“或”运算**	操作数1 || 操作数2
	!	**对两个表达式执行逻辑“非”运算**	! 操作数
	( )	**将操作数强制转换为给定的数据类型**	(数据类型) 操作数

运算符 条件运算符

x	y	!x	x || y	x && y
true	true	false	true	true
true	false	false	true	false
false	true	true	true	false
false	false	true	false	false

运算符 逻辑运算符

- & 逻辑AND
  - 为整型和 bool 类型预定义了二进制 & 运算符。对于整型，& 计算操作数的逻辑按位“与”。对于 bool 操作数，& 计算操作数的逻辑“与”；也就是说，当且仅当两个操作数均为 true 时，结果才为 true。

```
1. Console.WriteLine(true & false); // logical and
2. Console.WriteLine(true & true); // logical and
3. Console.WriteLine("0x{0:x}", 0xf8 & 0x3f); // bitwise and
```

- | 逻辑OR
  - 二元 | 运算符是为整型和 bool 类型预定义的。对于整型，| 计算操作数的按位“或”结果。对于 bool 操作数，| 计算操作数的逻辑“或”结果；也就是说，当且仅当两个操作数均为 false 时，结果才为 false。
  - Console.WriteLine(true | false); // logical or Console.WriteLine(false | false); // logical or Console.WriteLine("0x{0:x}", 0xf8 | 0x3f); // bitwise or

位运算

- 位运算是高级语言中的”低级”运算，其操作的对象是整型数，在机器内部二时制表示的每一位(bit).
- C#提供如下六种位运算符：
  - 双目位运算符： & | ^ >> <<
  - 单目位运算符：~

接位与运算符 &|^

- A & B的值是37,计算过程是按位进行逻辑与运算

39	0	0	1	0	0	1	1	1
45	0	0	1	0	1	1	0	1
37	0	0	1	0	0	1	0	1

- A |B的值是47,计算过程是按位进行逻辑或运算

39	0	0	1	0	0	1	1	1
45	0	0	1	0	1	1	0	1
37	0	0	1	0	1	1	1	1

- A ^B的值是10,计算过程是按位进行逻辑异或运算

39	0	0	1	0	0	1	1	1
45	0	0	1	0	1	1	0	1
37	0	0	0	0	1	0	1	0

按位取反运算 ~

- A的原值是　00100111
- A~ 的的值是 11011000
- 按位左移运算<<:
  - 则a<<1等于01001110
  - 则a<<3等于00111000
- 按位右移运算>>:
  - 则a>>1等于00010011
  - 则a>>3等于00000100

前置和后置自加/自减运算符

表达式	类型	计算方法	结果（假定 num1 的值为 5）
num2 = ++num1;	前置自加	num1 = num1 + 1; num2 = num1;	num2 = 6; num1 = 6;
num2 = num1++;	后置自加	num2 = num1; num1 = num1 + 1;	num2 = 5; num1 = 6;
num2 = --num1;	前置自减	num1 = num1 - 1; num2 = num1;	num2 = 4; Num1 = 4;
num2 = num1--;	后置自减	num2 = num1; num1 = num1 - 1;	num2 = 5; Num1 = 4;

运算符和表达式

类别	运算符	说明	表达式
三元运算符（条件运算符）	?:	检查给出的第一个表达式 expression 是否为真。如果为真，则计算 operand1,否则计算 operand2。这是唯一带有三个操作数的运算符	表达式? 操作数1: 操作数2

运算符 三元运算符

- 三元运算符
  - if...else结构的简化形式。其名称的出处是它带有三个操作数。它可以计算一个条件，如果条件为真，就返回一个值，如果条件为假，则返回另一个值。其语法如下：
  - 条件?为真:为假
  - Console.WriteLine(X>=0?”我是真”:”我是假”);

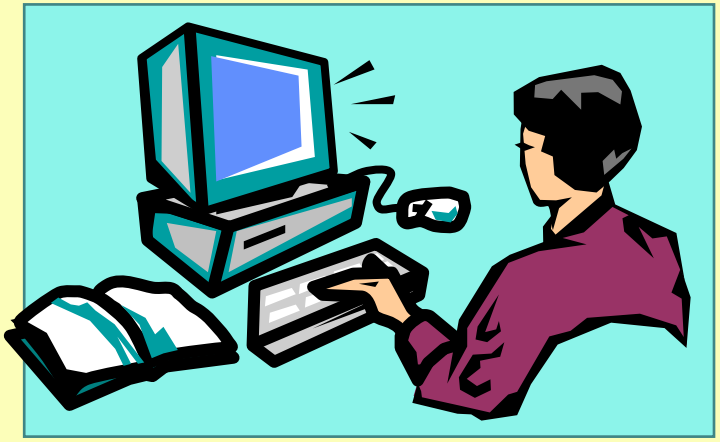
C# 运算符的优先级

优先级（1 最高）	说明	运算符	结合性
1	括号	()	从左到右
2	自加/自减运算符	++/--	从右到左
3	乘法运算符 除法运算符 取模运算符	* / %	从左到右
4	加法运算符 减法运算符	+ -	从左到右
5	小于 小于等于 大于 大于等于	< <= > >=	从左到右
6	等于 不等于	= !=	从左到右 从左到右
7	逻辑与	&&	从左到右
8	逻辑或	||	从左到右
9	赋值运算符和快捷运算符	= += *= /= %= -=	从右到左

运算符 运算符-总结

运算符类别	运算符
算术	+ - * / %
逻辑（布尔型和按位）	& | ^ ! ~ && || true false
字符串串联	+
递增、递减	++ --
移位	<< >>
关系	== != < > <= >=
赋值	= += -= *= /= %= &= |= ^= <<= >>=
成员访问	.
索引	[]
转换	()
条件	?:
委托串联和移除	+ -
创建对象	new
类型信息	as is sizeof typeof
溢出异常控制	checked unchecked
间接寻址和地址	* -> [] &

Lab 实验 操作符



- 请求用户输入两个整数，判断第一个整数是否是第二个整数的倍数，即：
  - 第一个整数：15
  - 第二个整数：3
  - 输出“第一个整数是/不是第二个整数的倍数”。
  - 提示：用%操作符

Lab 实验-温度转换

- 编一个华氏温度与摄氏温度之间转换的程序，运行界面如下：
  - 提示：摄氏温度=(华氏温度-32)*5/9
  - {i,+/-x:yn}
    - i代表变量的索引
    - x代表显示的宽度，+/-代表右对齐或左对齐
    - y代表格式
    - n代表精度
  - 注意：如果按照默认的或者n所指定的精度显示的数字长度超过x，则x将不起作用

常量

- 常量非常类似于静态只读字段。在声明和初始化变量时，在变量的前面加上关键字const，就可以把该变量指定为一个常量。顾名思义，常量是其值在使用过程中不会发生变化的变量
- const int a=100;
- 常量与只读字段有4个方面的区别：
  - 局部变量和字段可以声明为常量
  - 常量必须在声明时初始化，不能声明为类级后，再在构造函数中给它指定一个值。指定了其值后，就不能再修改了
  - 常量的值必须能在编译时用于计算。因此，不能用从一个变量中提取的值来初始化常量，如果需要这么做，应命名用只读字段
  - 常量总是静态的，但注意不允许在常量声明中包含修饰符static
- 在程序中使用常量(或只读变量至少有三个好处)
  - 常量用易于理解的清楚的名称代替了"含义不明确的数字或字符串"，使程序更易于阅读
  - 常量使程序更易于修改
  - 常量更容易避免程序出现错误

3-9 checked和unchecked运算符

- 当对整数类型执行操作，而其值超过该数据类型的范围时，checked和uncheck运算符可以指定CLR如何处理堆栈溢出。例如，考虑下面的代码:

```
1.  byte b=255;
2.  b++;
3.  Console.WriteLine(b.ToString());
```

```
1.  byte b=255;
2.  checked
3.  {
4.     b++;
5.  }
6.  Console.WriteLine(b.ToString());
```

3-9 checked和unchecked运算符 Overflow

- checked关键字
  - 用于检查是否会出现溢出
- /checked编译开关：最好打开
  - 用于检查是否会出现溢出
- 特别要注意每种数据类型允许的值范围

3-10 is运算符

- is运算符可以检查对象是否与特定的类型兼容。例如，要检查变量是否与object类型兼容

```
1.  int i=10;
2.  if(i is object)
3.  {
4.     Console.WriteLine(“这是一个object对象”);
5.  }
```

3-11 sizeof运算符

- 使用sizeof运算符可以确定堆栈中值类型需要的长度(单位是字节)

```
1.  string s=“A string”;
2.  unsafe
3.  {
4.      Console.WriteLine(sizeof(int));
5.  }
```

第六节 控制结构

- 选择语句
- 循环语句
- 挑转语句

流控制

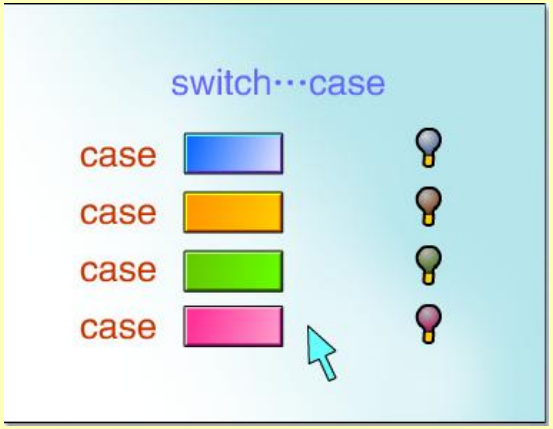
- if语句

```
1.    if(condition)
2.         statement(s)
3.    else
4.        statement(s)
```

```
int intA=5, intB = 6;
if (intA > intB)
  Console.WriteLine("没有错intA大于intB");
else
  Console.WriteLine("这种事怎么能出现哪？");
```

流控制-switch语句

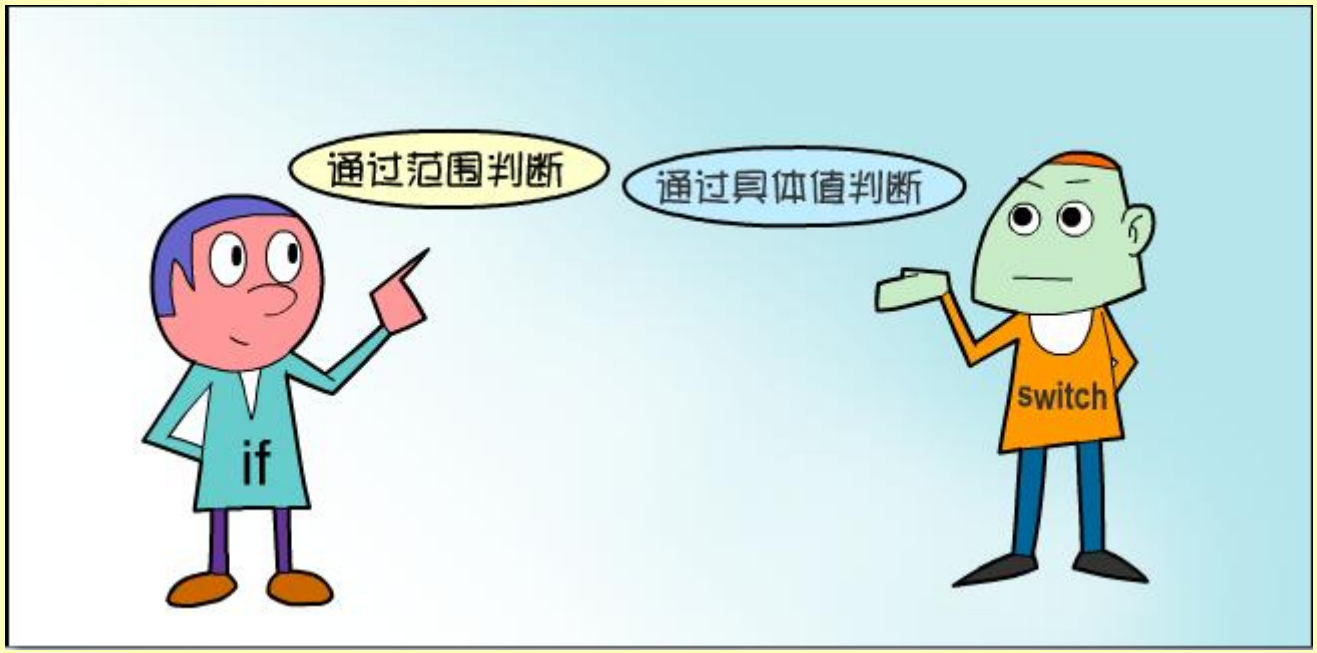
```
1.    switch (intA)
2.                {
3.                    case 5:
4.                        Console.WriteLine("没有错就是等于");
5.                        break;
6.                    case 6:
7.                        Console.WriteLine("有点邪了");
8.                        break;
9.                    default:
10.                       Console.WriteLine("都不是那就是我的了！");
11.                       break;
12.               }
```



流控制-switch语句

```
1.   string myCountry = Console.ReadLine();
2.              switch (myCountry)
3.              {
4.                  case "中国":
5.                      Console.WriteLine("您选择的是中国");
6.                      break;
7.                  case "美国":
8.                      Console.WriteLine("您选择的是美国");
9.                      break;
10.                 default:
11.                     Console.WriteLine("您选择的是中国");
12.                     break;
13.             }
```

流控制 Switch与if语句的差别



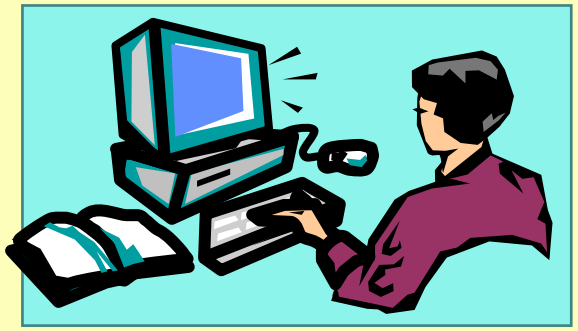
流控制 循环-for语句

```
1.      for (int a = 0; a < 100; a++)
2.                  {
3.                      Console.WriteLine("您输出的是{0}",a.ToString());
4.                  }
```

```
1.                  for (int i = 1; i <= 9; i++)
2.                  {
3.                      for (int j = 1; j <= i; j++)
4.                      {
5.                          Console.Write("{0}*{1}={2} ",i,j,i*j);
6.                      }
7.                      Console.WriteLine();
8.                  }
```

Lab 实验-九九表

- 打印小九九表



流控制 循环-while语句

```
1.              bool mywhile = true;
2.              while (mywhile)
3.              {
4.                  Console.WriteLine("这是一个死循环哪！");
5.              }
```

流控制 循环语句-do

```
1.    do
2.        {
3.            Console.WriteLine("我是先执行再判断!");
4.        } while (mywhile);
```

流控制 循环语句-foreach语句

- foreach循环是我们讨论的最后一种C#循环机制。其他循环机制都是C和C++的最早期版本，而foreach语句是新增的循环机制(借用于VB).也是非常受欢迎的一种循环

```
1.  int[] ints ={ 1,2,3};
2.              foreach (int temp in ints)
3.              {
4.                  Console.WriteLine(temp);
5.              }
```

Lab 实验-学生成绩

- 请求用户输入一个百分制的成绩(整数)，输出相应的成绩等级。等级如下：
  - >100或<0：非法成绩
  - >=85：优秀
  - >=70并且<85：中
  - >=60并且<70：及格
  - <60：不及格

Lab 实验-循环测试

- 利用while语句写一个计算1到100之和的C#程序，并输出计算结果。
- 利用do语句写一个计算1到100之和的C#程序，并输出计算结果。
- 利用for语句写一个计算1到100之和的C#程序，并输出计算结果。

流控制 跳转语句-goto语句

- goto语句可以直接跳转到程序中用标签指定的别一行(标签是一个标识符，后跟一个冒号):

```
1.  goto Label1:
2.       Console.Writeline(“this won’t be executed”);
3.  Label1:
4.       Console.WriteLine(“from here”);
```

流控制 跳转语句-break语句

- 实际上，break也可以用于退出for、foreach、while或do...while循环，循环结束后，立即执行后面的语句:

```
1.  for (int i = 0; i < 10; i++)
2.              {
3.                  Console.Write("请输入一个语句(输入end结束):");
4.                  string s = Console.ReadLine();
5.                  if (s == "end")
6.                  {
7.                      break;
8.                  }
9.                  Console.WriteLine("您输入的单词:"+s);
10.             }
```

跳转语句-continue语句 return语句 using语句

- Continue语句类似于break,也必须用于for/foreach/while或do..while循环中，它只从循环的当前迭代中退出，然后在循环的下一次迭代开始重新执行，而不是退出循环
- return语句用于退出类的方法，把控制返回方法的调用者，如果方法有返回类型，return语句必须返回这个类型的值，如果方法没有返回类型，该语句就不能用于表达式
- using语句可以确保在使用完资源密集型的对象后，就处理它们，其语法如下：

```
1.  using(object)
2.  {
3.      //code using object
4.  }
```

注释

- C#使用传统的C风格注释：单行注释使用//,多行注释使用/* ...*/
- //这是一段代码注释
- /*这是一段代码注释*/

注释 XML注释

标识符	说明
<c>	把一行中的文本标记为代码,例如<c>int i=10;</c>
<code>	把多行标记为代码
<example>	标记为一个代码示例
<exception>	说明一个异常类(编译器要验证其语法)
<include>	包含其他文档说明文件的注释
<list>	把列表插入到其文档说明中
<param>	标记为方法的参数
<paramref>	表示一个单词是方法的参数
<returns>	文档方法的返回值
<summary>	提供类或成员的简短小结
<value>	描述属性

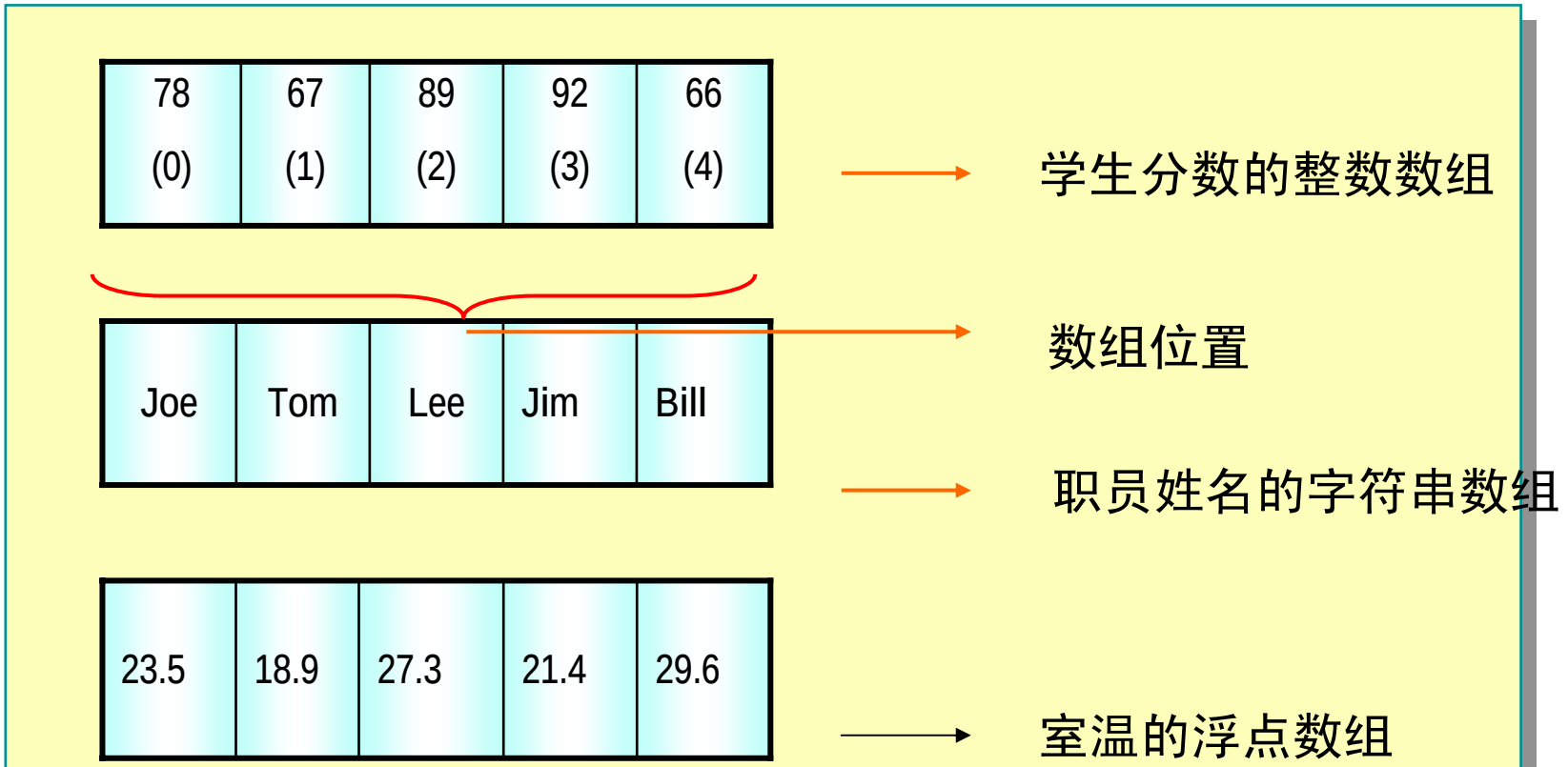
C#数组

- 如果需要用到很多个同一类型的变量，如要定义”int a,a1,a2……a100”，这样定义就很麻烦。
- 数组就可以解决这个问题，这样就行：
  - `int [] intArray`

数组

- 数组是同一数据类型的一组值
- 数组属于引用类型，因此存储在堆内存中
- 数组元素初始化或给数组元素赋值都可以在声明数组时或在程序的后面阶段中进行

数组


78
(0)
67
(1)
89
(2)
92
(3)
66
(4)
学生分数的整数数组
Joe
Tom
Lee
Jim
Bill
数组位置
职员姓名的字符串数组
23.5
18.9
27.3
21.4
29.6
室温的浮点数组

C#数组 声明

- C# 数组下标为0
- 可以不指定数组的大小

```
int[] numbers; // declare numbers as
               // an int array of any size
```

```
int[] numbers = new int[5]; // declare and create
```

```
int[] numbers = new int[5] {1, 2, 3, 4, 5};
```

```
int[] numbers = {1, 2, 3, 4, 5};
int LengthOfNumbers = numbers.Length;
```

C#数组概述

- **数组具有以下属性：**
  - 数组可以是一维、多维或交错的。
  - 数值数组元素的默认值设置为零，而引用元素的默认值设置为 null。
  - 交错数组是数组的数组，因此，它的元素是引用类型，初始化为 null。
  - 数组的索引从零开始：具有 n 个元素的数组的索引是从 0 到 n-1。
  - 数组元素可以是任何类型，包括数组类型。
  - 数组类型是从抽象基类型 Array 派生的引用类型。由于此类型实现了 IEnumerable 和 IEnumerable，因此可以对 C# 中的所有数组使用 foreach 迭代。

C#数组-Demo

```
1.  class TestArraysClass
2.  {
3.      static void Main()
4.      {
5.          // Declare a single-dimensional array
6.          int[] array1 = new int[5];

7.          // Declare and set array element values
8.          int[] array2 = new int[] { 1, 3, 5, 7, 9 };

9.          // Alternative syntax
10.         int[] array3 = { 1, 2, 3, 4, 5, 6 };

11.         // Declare a two dimensional array
12.         int[,] multiDimensionalArray1 = new int[2, 3];

13.         // Declare and set array element values
14.         int[,] multiDimensionalArray2 = { { 1, 2, 3 }, { 4, 5, 6 } };

15.         // Declare a jagged array
16.         int[][] jaggedArray = new int[6][];

17.         // Set the values of the first array in the jagged array structure
18.         jaggedArray[0] = new int[4] { 1, 2, 3, 4 };
19.     }
20. }
```

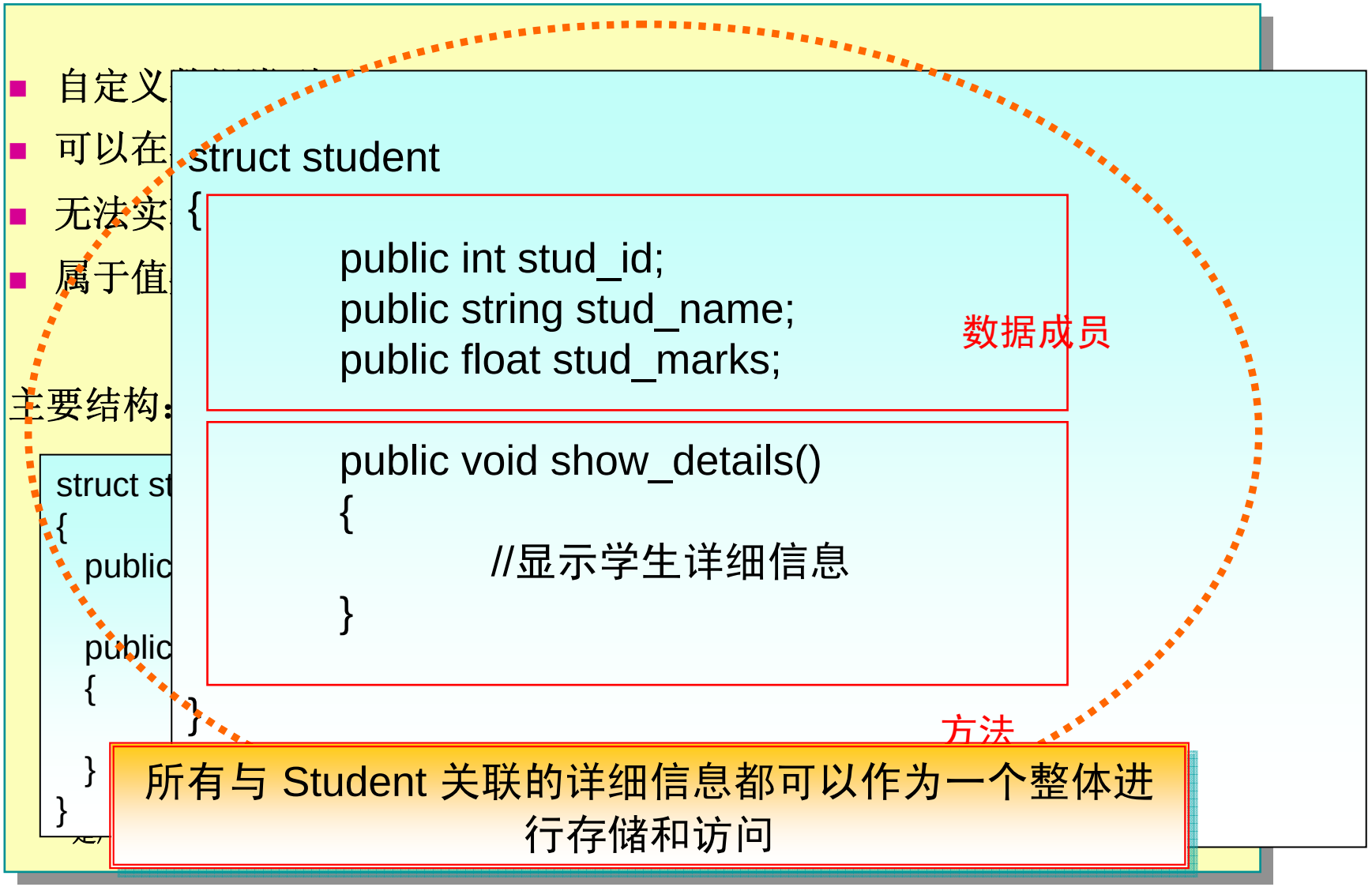
C#数组的维数

```
1.  class TestArraysClass
2.  {
3.      static void Main()
4.      {
5.          // Declare and initialize an array:
6.          int[,] theArray = new int[5, 10];
7.          System.Console.WriteLine("The array has {0} dimensions.",
    theArray.Rank);
8.      }
9.  }
```

C#数组使用foreach

```
1.  int[] numbers = { 4, 5, 6, 1, 2, 3, -2, -1, 0 };
2.  foreach (int i in numbers)
3.  {
4.      System.Console.WriteLine(i);
5.  }
6.  int[,] numbers2D = new int[3, 2]
7.  { { 9, 99 }, { 3, 33 }, { 5, 55 } };
8.  foreach (int i in numbers2D)
9.  {
10.     System.Console.Write("{0} ", i);
11. }
```

结构


自定义
可以在
无法实
属于值
主要结构：
struct st
{
public
public
{
}
}
struct student
{
public int stud_id;
public string stud_name;
public float stud_marks;
数据成员
public void show_details()
{
//显示学生详细信息
}
}
方法
所有与 Student 关联的详细信息都可以作为一个整体进行存储和访问

枚举

- 枚举（Enum，Enumerator 的缩写）是一组已命名的数值常量
- 用于定义具有一组特定值的数据类型
- 枚举以 enum 关键字声明

```
public class Holiday
{
  public enum WeekDays
  {
            Monday,
            Tuesday,
            Wednesday,
            Thursday,
            Friday
  }
  public void GetWeekDays (String EmpName, WeekDays DayOff)
  {
     //处理工作日
  }
  static void Main()
  {
    Holiday myHoliday = new Holiday();
    myHoliday.GetWeekDays ("Richie", Holiday.WeekDays.Wednesday);
  }
}
```

Wednesday = 2

枚举（续）

- C# 中的枚举包含与值关联的数字
- 默认情况下，将 0 值赋给枚举的第一个元素，然后对每个后续的枚举元素按 1 递增
- 在初始化过程中可重写默认值

```
public enum WeekDays
{
        Monday=1,
        Tuesday=2,
        Wednesday=3,
        Thursday=4,
        Friday=5
}
```

关于实验

- 不要怕犯错，只有不断犯错，不断修正错误才能不能进步、增长实践经验
- 一定要诚实，会做就会做，不会做可以请教老师和同学，但一定要搞懂，不要仅仅是为了应付，否则是害了自己

Lab 实验-打印三角形

- 根据输入的行数，打印三角形

Lab 实验-递归求阶乘

- 根据输入的数，求1~n的阶乘

Lab 实验-冒泡排序

- 冒泡算法

附录

- 常用名称空间

常用名称空间.txt

Lab-1答案

```
static void Main(string[] args)
{

        Console.Write("请输入圆的半径：");
        string answer=Console.ReadLine();
        double radius=double.Parse(answer);

        double area=Math.PI*radius*radius;

         double circum=2*Math.PI*radius;

        Console.WriteLine("该圆的周长是{0}， 面积是{1}",circum,area);
}
```

Lab-2答案

```
int first=int.Parse(this.textBox1.Text);
int second=int.Parse(this.textBox2.Text);
int i=first%second;
if (i==0)
        MessageBox.Show("第一个整数是第二个整数的倍数");
else
        MessageBox.Show("第一个整数不是第二个整数的倍数")
```