int和char的对应关系

```
for (int i = 0; i < 128; i++)
{
      Console.Write((char)i);
      if (i % 10 == 0)
           Console.WriteLine();
}
```

国际通用编码表

ascii码对照表

代码	字符	代码	字符	代码	字符	代码	字符	代码	字符
32		52	4	72	H	92	\	112	p
33	!	53	5	73	I	93	]	113	q
34	"	54	6	74	J	94	^	114	r
35	#	55	7	75	K	95	_	115	s
36	$	56	8	76	L	96	`	116	t
37	%	57	9	77	M	97	a	117	u
38	&	58	:	78	N	98	b	118	v
39	'	59	;	79	O	99	c	119	w
40	(	60	<	80	P	100	d	120	x
41	)	61	=	81	Q	101	e	121	y
42	*	62	>	82	R	102	f	122	z
43	+	63	?	83	S	103	g	123	{
44	,	64	@	84	T	104	h	124	|
45	-	65	A	85	U	105	i	125	}
46	.	66	B	86	V	106	j	126	~
47	/	67	C	87	W	107	k		
48	0	68	D	88	X	108	l		
49	1	69	E	89	Y	109	m		
50	2	70	F	90	Z	110	n		
51	3	71	G	91	[	111	o		

记住几个关键数据

‘0’ =48

‘A’ =65

'a' =97

所以char也可以比较大小 比对方式是通过ASCII码表为参照的

练习：

输入一个英文字母 输出是大写还是小写 并转换大小写

```
char x =(char)Console.Read();
if (x >= 65 && x <= 90)
{
     Console.WriteLine("是大写");
     Console.WriteLine((char)(x + 32));
}
else if (x >= 97 && x <= 122)
{
    Console.WriteLine("是小写");
    Console.WriteLine((char)(x - 32));
}
```

随机数

Random

在程序开发中 很多数据是随机得来的

比如 彩票号码的生成 棋牌游戏的发牌 游戏中的抽奖 怪物巡逻位置等

新建一个随机数

```
Random ran = new Random();
int a =ran.Next();
```

Next方法返回一个非负正整数 如果需要随机负数 在前加负号

返回一个0-99的正整数

```
Random ran = new Random();

for (int i = 0; i < 100; i++)
{
    int a =ran.Next(100);
    Console.WriteLine(a);
}
```

根据输出结果 Next(100)会得到0-99的随机数 不包括100

所以 50-100的随机数写法：

int a =50+ran.Next(51);

返回一个0.0-1.0之间的随机数

double a =ran.NextDouble();

例：

一件装备当前强化等级是7级

7级强化到8级的成功率是40%

强化成功 装备强化等级+1

强化失败 装备强化等级重置为零

```
Random ran = new Random();
int level = 7;
double x=0.4;
Console.WriteLine("启示强化等级为" + level + "    开始强化********");
double r = ran.NextDouble();
if (r <= x)
{
    Console.WriteLine("强化成功");
    level++;
}
else
{
    level = 0;
    Console.WriteLine("强化失败");
}
Console.WriteLine("当前等级为"+level);
```

习题：

一键强化功能

强化一次装备需要消耗 100点金钱

默认初始装备等级是0 强化成功率为60%
玩家手动输入拥有的金钱数 和 希望强化到的等级数

输出每次强化的结果和剩余金钱

如果金钱不足 退出强化系统 并提示金钱不足 显示剩余金钱

如果强化目标完成 退出强化系统 显示剩余金钱

```
            Random ran = new Random();
            Console.WriteLine("手动输入金钱");
            int money = int.Parse(Console.ReadLine());
            Console.WriteLine("输入目标等级");
            int targetLevel = int.Parse(Console.ReadLine());

            int level = 0;

            while (level < targetLevel)
            {
                Console.WriteLine("*********开始强化**********");
                if (money < 100)
                {
                    Console.WriteLine("金钱不足" + money + "强化结束");
                    break;
                }
                money -= 100;
                double r = ran.NextDouble();
                if (r < 0.6)
                {
                    level++;
                    Console.WriteLine("恭喜强化成功");
                    Console.WriteLine("当前等级" + level);
                }
                else
                {
                    Console.WriteLine("强化失败");
                    Console.WriteLine("当前等级" + level);
                }
                Console.WriteLine("剩余金钱" + money);
            }

```

习题：

商场举行消费反馈活动

当用户购物满1000元 可以参与抽奖环节

当用户购物不足1000元 不能参与抽奖

每位用户每满1000元可以抽奖一次
最多抽奖3次

抽奖的奖品为：

1 电视机一台

2 微波炉一台

3 毛绒玩具一个

4 牙签一盒

要求：手动输入玩家消费金额 判断是否可以参与抽奖

随机抽取5个奖品中的一个

最后输出用户的奖品

```
            Console.WriteLine("请输入消费金额");
            int cost = int.Parse(Console.ReadLine());

            if (cost>=1000)
            {
                Console.WriteLine("消费满足抽奖要求 开始抽奖");
                Random random = new Random();

                int count = cost /1000;

                if (count > 3)//最多抽奖3次
                    count = 3;

                for (int i = 1; i <= count; i++)
                {
```

```
                    Console.WriteLine("第" + i + "次抽奖");

                    int a = 1 + random.Next(5);

                    switch (a)
                    {
                        case 1:
                            Console.WriteLine("恭喜你获得电视机的奖励");
                            break;
                        case 2:
                            Console.WriteLine("恭喜你获得微波炉的奖励");
                            break;
                        case 3:
                            Console.WriteLine("恭喜你获得毛绒玩具的奖
励");
                            break;
                        case 4:
                            Console.WriteLine("恭喜你获得换购卫生纸的奖
励");
                            break;
                        case 5:
                            Console.WriteLine("恭喜你获得免费的牙签");
                            break;
                    }
                }
            }
            else
            {
                Console.WriteLine("消费金额不足1000 不能参与抽奖活动");
            }
```

数组 Array

数组的存在意义：

例：

一个班级一共30个人 考试结束 需要计算全班的平均成绩

那么我们就需要定义30个变量 来存储全班学员的成绩 变量太多太繁琐不易管理

那么解决办法 就是把这多个学员的成绩都存储在一个数组中

数组也时一个变量 是用来存储相同类型数据的集合

声明变量时是在内存空间划出一块合适的空间

声明数组时时在内存空间划出一串合适的空间

```
int[] array;    //声明一个int类型的数组 数组名array （标识符）
array = new int[5]; //分配空间：设定数组的长度是5 即数组能存储的数据个数 注意：数组长度一旦确定是不会改变的
array[0] = 100;  //0是数组的索引 程序中的索引值默认都是从0开始 给数组元素赋值
array[1] = 200;
array[2] = 300;
array[3] = 400;
array[4] = 500; //最大索引值是4
//array[5] = 600; 报错 数组越界
Console.WriteLine(array[0]);//使用数组 根据索引获取数据
```

创建数组并赋值的其他写法：

```
//边声明边赋值
int[] array1 = new int[]{ 100, 200, 300, 400, 500 };
int[] array2 = { 200, 300, 400, 400 };

//使用for循环给数组赋值
int[] array3 = new int[5];

//Length是数组的长度
for (int i = 0; i < array3.Length; i++)
{
    array3[i] = i * 100;
}
```

数组的特征：

1 相同数据类型的数据才可以存储在一个数组中

2 当数组的长度一旦确定 不可更改

3 数组中的元素拥有连续内存地址

4 数组索引从0开始 要小心数组越界的错误

例：

计算全班学员平均成绩：

```
//共5个学员
int[] scores = new int[5];

Console.WriteLine("请输入5名学员的成绩");

int sum=0;
for (int i = 0; i < scores.Length; i++)
{
    scores[i] = int.Parse(Console.ReadLine());
    sum+=scores[i];
}

Console.WriteLine("全班的平均分是" + sum / scores.Length);
```

练习：

猜数游戏：

从键盘上输入一个数据 判断数组中是否包含这个数 输出包含或者不包含

```
int[] scores = {100,200,300,400};

int x = int.Parse(Console.ReadLine());

```

```
bool it = false;//记录结果
for (int i = 0; i < scores.Length; i++)
{
        if (scores[i] == x)
        {
            Console.WriteLine("包含");
            it = true;//当找到时 把bool设置为true
            break;
        }
}

if(!it)//如果不需要bool值 就需要判断是不是最后一个元素
    Console.WriteLine("不包含");
```

练习：

求数组的最大值

打擂台方式：

假定第一个元素最大 用一个max变量来存储

然后for循环逐个比较 当出现比max更大的数据 就把这个值赋值给max

```
int[] scores = {100,200,300,400};

int max = scores[0];

for (int i = 0; i < scores.Length; i++)
{
    if (scores[i] > max)
    max = scores[i];
}
Console.WriteLine("最大值是" + max);
```

引用数据类型和基本数据类型

数组是引用数据类型 int，double，float等属于基本数据类型

例：

```
int[] scores = {100,200,300,400};
int[] x = scores;
x[1] = 600;
Console.WriteLine(scores[1]);//scores[1]的数据也随之改变了 这是和基本类型不同的地方
```

当我们用一个int类型的变量给另一个int类型的变量赋值时 这是值赋值

两个int在不同的内存空间 所以改变一个不影响另一个

但如果时引用类型 比如数组 使用等于号让一个数组给另一个数组赋值 这是引用赋值

两个数组变量虽然名字不同 但是都是指向同一个内存区域

所以改变其中一个会影响另一个

所以 如果仅仅是想把一个数组的值copy到另一个数组 需要使用for循环逐个赋值

写法：

```
int[] scores = {100,200,300,400};
int[] x = new int[scores.Length];

for (int i = 0; i < x.Length; i++)
    x[i] = scores[i];

x[1] = 600;
Console.WriteLine(scores[1]);
```

练习：

1 把一个本来长度为4的int数组 添加一个新数据

注意：数组的长度不能修改 增加长度必须重新开辟长度

```
            int[] x = { 100, 200, 90, 29 };

            //防止原数组的数据丢失 先再次创建一个数组记录
            int[] y = x;

            x = new int[y.Length + 1];

            Console.WriteLine(x[0]);

            for(int i=0;i<y.Length;i++)
            {
                x[i]=y[i];
            }

            //添加新数据
            x[x.Length-1]=500;


            Console.WriteLine(x[0]);
            Console.WriteLine(x[4]);
```

2 在一个长度为4的int数组中插入新数据

3 把1个Int类型的数组从小到大排序

冒泡排序算法

每次外层循环一次 确定一个位置的数

即 第一次大循环确定最小值 放在数组的第一个元素（交换变量的值）

每次确定的数不参与下次大循环的比较

```
            int[] x = { 20, 17, 22, 90, 7 };

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

```
            {
                for (int j = i; j < x.Length; j++)
                {
                    if (x[j] < x[i])
                    {
                        int a = x[i];
                        x[i] = x[j];
                        x[j] = a;
                    }
                }
            }

            for (int i = 0; i < x.Length; i++)
                Console.WriteLine(x[i]);
```