# Cha3-认识 Unity3d 的好基友 C# 05

在上一章的内容中，我们了解了 C#中的运算符和表达式。考虑到计算机相比人类最强大的地方就是在于其恐怖的计算能力，学习这些知识是非常有用的。

而在这一课的内容中，我们将学习 C#的另外一个重要语法基础，也就是流程控制。

让我们开始吧~

个人微信号：iseedo

微信公众号：vrlife

**06 认识 Unity3d 的好基友 C#-C#中的流程控制**

让我们想让计算机帮我们完成某项任务的时候，需要告诉它先做什么，后做什么。此外，我们也会遇到一些逻辑判断的问题。

比如，如果法师的法力值大于 0，那么就可以释放法术。但如果法师的法力值等于 0，就不得不躺地板了~

简单来说，所谓的流程控制，就是对某个条件进行判断，如果判断通过，就执行某语句。

C#中的流程控制方法和其他语言基本相同。支持 if…else，while，do…while，for，foreach，switch 这些流程控制语句。

## 1. if…else（如果…那么…）

if…else 是最基本也是最常用的流程控制，表示在满足某种特定的条件情况下，将执行某种操作，否则将执行 else 后面的操作。

例如：

```
    //定义一个布尔变量
  bool willTommorrowRain = false;


 //根据明天的天气来决定要做什么事情
    if (willTommorrowRain)
    {
       Debug.Log("在家看权力的游戏第 8 季~");
    }
```

在以上的代码中，我们首先定义了一个 **布尔**类型变量保存明天的天气状态，然后设置了一个初始值为 **false** ，也就是不下雨。随后进行判断，willTommorrowRain 的值是否为 true。如果判断成立，则执行花括号中的语句。

在这里，**因为** willTommorowRain 的值是 false，所以不会执行下面的这条语句。

如果我们想让判断更清晰，那么逻辑可以进一步优化。

如果条件成立，就去做第一件事，否则就去做第二件事：

```
//定义一个布尔变量
    bool willTommorrowRain = false;

//根据明天的天气来决定要做什么事情
        if (willTommorrowRain)
        {
            Debug.Log("在家看权力的游戏第 8 季~");
        }
        else
```

**Commented [1]:**
已修改

**Commented [2]:**
改为：以下语句的含义是：
原句不通顺
可直接在电子档上修改

```
        {
            Debug.Log("去深圳湾公园捡贝壳~");
        }
```

根据**我们刚才的判断，因为明天不下雨，所以会输出“去深圳湾公园捡贝壳~”。**

## 2.while

while 和 if…else 语句相似，表示当满足某个条件的情况下将执行某些操作，但是它没有 else 部分。

例如：

```
    //定义两个整数变量，保存支付宝账户余额以及本月订单数
    int myAlipayBalance = 100000;
    int amountOforders = 0;

//根据支付宝的余额决定是否买买买~
        while(myAlipayBalance > 99)
        {
            //如果余额足够，当然是买买买了。
            Debug.Log("买买买！");

            //从支付宝余额中扣款~
            myAlipayBalance -= 100;

            //本月订单数量增加 1
            amountOforders++;
        }

        Debug.Log("本月一共买买买了 " + amountOforders + "件东西，还敢这么剁手吗？");
        Debug.Log("余额宝中的余额还有" + myAlipayBalance + "月光不是梦，还敢这么剁手吗？~");
```

在以上的代码中，我们首先定义了两个整数变量，分别用来保存支付宝账户余额以及本月订单数。

接下来我们根据余额宝中的余额决定是否买买买~

假定我们要买的东西每次都是 100 元（太理想了点），那么只要余额宝中的余额大于 99 元，显然是可以买买买的。

每次购买后，我们首先输出一行信息表示已经买买买了，然后从支付宝余额中扣款 100 元，再将本月订单数量增加 1.

最后再输出两条信息，警示一下自己，以后还是要断舍离，否则就成了真正的月光族了。

### 3. do…while

do…while 和 while 的区别在于，do...while 语句并不会首先进行判定，它会先执行一次括号中的语句，再进行判定，如果判定成立，再继续执行括号中的语句。

接着上面的例子。

虽然我们已经月光了，但是还有花呗啊，还有各种分期贷啊~

于是更作的事情来了。

```
//不管支付宝里面的余额够不够，先买了再说，当然得用花呗了。
      do {
          //如果余额足够，当然是买买买了。
          Debug.Log("买买买！");

          //从支付宝余额中扣款~
          myAlipayBalance -= 100;

          //本月订单数量增加 1
          amountOforders++;
      } while(myAlipayBalance > 99)
```

在以上的代码中，不管余额还有多少，先买买买再说，这就有点杯具了，从月光转眼成了首负。

## 4.for

for 语句同样用来循环执行某些指令，不过跟 while 和 do…while…不同，for 循环通常用来执行某些固定次数的循环。

它的形式一般是下面这样的：

for(初始值;判断语句;值变化语句){}

比如：

```
//for 循环通常用于执行固定次数的循环指令
for(int i = 0; i<=10; i++)
{
   Debug.Log("热烈欢迎北境之王和龙母莅临临冬城考察指导工作");
}
```

## 5.foreach

foreach 并不算严格意义上的流程控制语句，它的作用是依次遍历集合中的数据。

问题来了，什么是集合？这个名词听起来很唬人，其实很好理解，当我们把一组数据用某种方式组织起来，就是一个集合。当然，集合有很多种形式，最常见的就是数组了。

什么是数组？首先数组是一种特定形式的集合。其次，它是一种有序的元素集合。也就是说数组中的数据是有固定的排列顺序的。

这么说还是很抽象，我们用具体的示例来说明~

比如：

冰与火之歌里面的英雄战斗力各不相同，我们可以把主要英雄的战力保存在一个数组中，方便“风暴降生丹妮莉丝，安达尔人、罗伊纳人及先民的女王，七国统治者兼全境守护，大草海的卡丽熙，不焚者，镣铐破碎者，弥撒，弥林的女王”大人进行查看，并遣兵调将来对抗衣柜。

例如：

```
 //定义一个数组，用来保存英雄的战力值
        int[] heroPowers = new int[] { 358, 762, 1903, 10, 998,
12, 13 };

        //遍历数组中的每个元素，输出每个英雄的战力值
        foreach(var power in heroPowers)
        {
            Debug.Log(power);
        }
```

在上面的示例中，我们首先定义了一个 int 数组，用来保存英雄的战力值。然后通过 foreach 遍历数组中的每一个元素。foreach 会按顺序依次取出数组中的每一个元素。

foreach 语句的通常写法是：

foreach(类型 变量名 in 集合){}

其中 var 是变量类型，num 为变量名，numbers 为集合名。foreach 支持任何类型集合的遍历，不仅仅只限于数组。

foreach 的性能开销很大，在 Unity 开发中应尽量避免使用。

## 6.Switch

Switch 语句是一个判断选择语句。当我们想要将单个变量的值与一系列常量进行比较时，就会用

到 Switch。

还是拿女王和她的臣子们举例，假定女王对每个手下都有自己的一番评价。为了方便管理，就可以用下面的代码来输出信息，而不用记下来长长的台词~

```
        //定义一个变量，用来保存臣子的姓名
        string heroName ="Jon Snow";

        switch (heroName)
        {

            case "Jon Snow":
                Debug.Log("He is my love，可惜know nothing");
                break;
            case "Sansa Stark":
                Debug.Log("小姑子，很厉害的北方女王，不好惹~");
                break;
            case "Jamie Lannister":
                Debug.Log("曾经的弑君者杀父仇人，眼不下这口气啊");
                break;
            case "Tyrion Lannister":
                Debug.Log("我的国王之手，但是最近智商全面下线，还敢相信他吗？");
                break;
            case "Cersei Lannister":
                Debug.Log("必须除掉的敌人，还有什么好说的");
                break;
            default:
                break;
        }
```

以上代码很好理解，我们定义了一个变量，来保存臣子的姓名。

接下来就是做一个判断，看他/她的姓名对应哪一个，然后给出相应的评价。

小练习：

1.创建一个新的项目，命名为 FlowControl

2.创建一个新的游戏对象，命名为 FlowControlObject

3.创建一个新的脚本 FlowControlScript，并将其关联到 FlowControlObject 上

4.把以上的代码手动输入到 FlowControlScript 中去，然后编译运行，在 Console 视图中预览输出的信息，是否跟自己的猜测相符。

如果有问题，可以在论坛中提问，或是参考本课对应的示例项目~

好了，本课的内容就到此结束了。

在这一课的内容中，我们主要认识了 C#中的流程控制语句。

在下一课的内容中，我们将了解函数的概念。

让我们下一课再见~