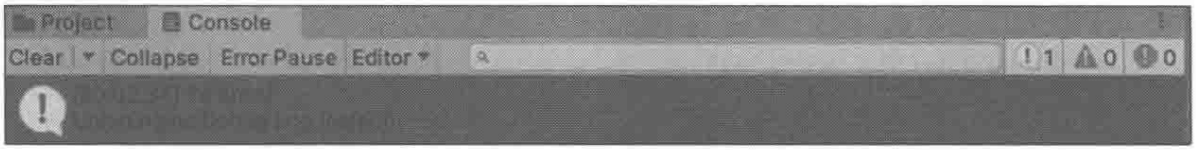

图 4-4 控制台中 switch 语句的输出

提示说明:

将 CharacterAction 更改为"Heal"或一个未定义的动作，以查看第一个 case 分支语句或 default 分支语句的结果。

有时需要多个(但不是全部的)switch 条件执行相同的动作，这被称为贯穿(fall-through)，在下一节我们将介绍它。

2. 贯穿

switch 语句可以在多个 case 分支语句下执行相同的操作，类似于在单个 if 语句中指定多个条件的方式。术语上称为贯穿。有了贯穿便可以为多个 case 分支定义同一组动作。如果 case 分支语句为空且其中没有 break 关键字，它将直接跳转到它下面的 case 分支。

case 和 default 可以按任何顺序编写，因此创建贯穿可以大大提高代码的可读性和效率。

让我们利用 switch 语句和贯穿模拟一个玩桌游的情景，它通过掷骰子决定特定动作的结果。

(1) 创建一个名为 DiceRoll 的 public int 变量，并将其赋值为 7。

```
public int DiceRoll = 7;
```

(2) 创建一个 public 的方法，没有返回值，取名为 RollDice。

```
public void RollDice()
{
}
```

(3) 添加一个 switch 语句，并使用 DiceRoll 作为匹配表达式。

```
switch(DiceRoll)
{
}
```

(4) 作为掷骰子可能的结果，添加三个 case 分支，它们分别是：7、15 和 20，并在最后添加一个 default 分支。

(5) case 15 和 case 20 应有各自的调试日志和中断语句，而 case 7 应贯穿至 case 15。

```
case 7:
case 15:
    Debug.Log("Mediocre damage, not bad.");
    break;
case 20:
    Debug.Log("Critical hit, the creature goes down!");
    break;
default:
    Debug.Log("You completely missed and fell on your face.");
    break;
```

(6) 在 Start 内调用 RollDice 方法。

```
void Start()
{
    RollDice();
}
```

(7) 保存文件并在 Unity 中运行它。

提示说明：

如果想查看贯穿的实际效果，尝试将调试日志添加到 case 7，但不要使用 break 关键字。

当将 DiceRoll 设置为 7 时，switch 语句将与第一个 case 分支进行匹配，由于其缺少可执行的代码块和 break 关键字，因此第一个 case 分支将贯穿并执行 case 15。如果将 DiceRoll 更改为 15 或 20，控制台将显示它们各自对应的消息，任何其他的值都将触发 switch 语句末尾的默认情况，如图 4-5 所示。

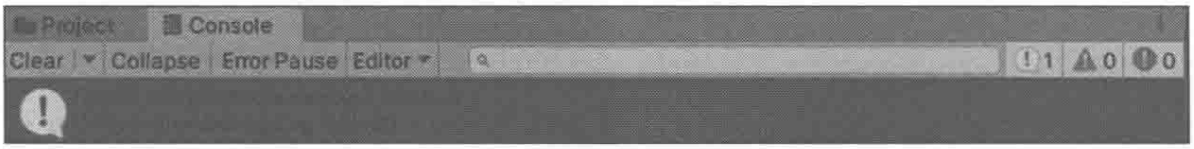

图 4-5　贯穿 switch 语句的代码结果

提示说明：

switch 语句非常强大，它甚至可以化简最复杂的决策逻辑。如果想要更深入地了解 switch 语句的模式匹配，可访问：https://docs.microsoft.com/en-us/dotnet/csharp/language-reference/keywords/switch。

这就是有关条件逻辑目前所需知道的全部内容，如有需要，可以随时重温