Homework Two：case 结构

颜铂林
数学与应用数学 3210101536

2022 年 6 月 28 日

导言：case 结构较之其他结构稍微复杂，具备匹配多个模式然后执行多条相关语句的能力，这使得它非常适合处理用户的输入。于是我尝试了 case 结构的三种使用方法，并作出以下报告。

1 case 语句简介

以下是 case 的语法：

```
case variable in
 pattern [ | pattern] ...) statements;;
 pattern [ | pattern] ...) statements;;
 ...
esac
```

虽然看起来复杂，但是 case 结构允许我们通过一种比较复杂的方式将变量的内容和模式进行匹配，然后再根据匹配的模式去执行不同的代码。这比使用多条 if，elif 和 else 语句来执行多个条件检查要简单的多。
注意，每个模式行都以双分号（;;）结尾。因为我们可以在前后模式之间放置多条语句，所以需要使用一个双分号来标记前一个语句的结束和后一个模式的开始
以下是三个示例。

2 示例一：用户输入

我们输入以下脚本程序：

```
#!/bin/sh
echo "Is it moring? Please answer yes or no"
read timeofday

case "$timeofday" in
    yes)  echo "Good Morning";;
    no )  echo "Good Afternoon";;
    y  )  echo "Good Morning";;
    n  )  echo "Good Afternoon";;
    *  )  echo "Sorry, answer not recognized";;
esac

exit 0
```

测试：
运行后提示 **Is it moring? Please answer yes or no**

输入yes 或者输入y 返回：Good Morning

输入no 或者输入n 返回：Good Afternoon

输入其他字符返回：Sorry, answer not recognized

解析：
当 case 语句被执行时，它会把变量 timeofdy 的内容与个字符串依次比较。一旦某个字符串与输入匹配成功，case 命令就会执行紧随右括号后面的代码，然后结束。
case 命令会对用来做比较的字符串进行正常的通配符扩展。这里只使用一个单独的 * 表示匹配任何可能的字符串，这样可以确保即使没有任何字符串得到匹配，case 语句也会执行某个默认指令。
因为 case 语句是按顺序比较每一个字符串，它不会去查找最佳匹配，而仅仅是查找地一个匹配，所以用 * 对脚本程序的调试很有帮助。

问题：
这样使用的 case 结构效果单一，与 if，elif，else 的结构相比并没有太大的优势，仅仅是看起来精致了许多。但下面的合并匹配模式，则体现的 case 结构的优越之处。

3 示例二：合并匹配模式

我们输入以下脚本程序：

```
#!/bin/sh

echo "Is it morning? Please answer yes or no"
read timeofday

case "$timeofday" in
    yes | y | Yes | YES )  echo "Good Morning";;
    n* | N* )              echo "Good Afternoon";;
    * )                    echo "Sorry, answer not recognized";;
esac

exit 0
```

测试：
运行后提示 **Is it moring? Please answer yes or no**

输入yes，y,Yes,YES 中的任意一个返回：Good Morning

输入 n 或 N 开头的字符串, 返回：Good Afternoon

输入其他字符返回：Sorry, answer not recognized

解析：
这个脚本程序在每个 case 条目中都使用了多个字符串，case 将对每个条目的多个字符串进行测试，以决定是否执行相应语句。它不仅再次使用了 * 通配符，并且多个字符串的判断使用替代了多个 if 语句的使用大大减少了脚本程序的代码量。

问题：
既然有了多个字符串的匹配，那么是否能够返回多条语句？答案是肯定的，下面的示例体现了 case 结构可以同时执行多条语句的特点。

4 示例三：执行多条语句

我们输入以下脚本程序：

```
#!/bin/sh

echo "Is it morning? Please answer yes or no"
read timeofday

case "$timeofday" in
    yes | y | Yes | YES )
           echo "Good Morning"
   echo "Up bright and early this morning"
   ;;
    [nN]*)
           echo "Good Afternoon"
   ;;
    *)
           echo "Sorry, answer not recognized"
   echo "Please answer yes or no"
   exit 1
esac
```

测试：
运行后提示 **Is it moring? Please answer yes or no**

输入yes，y,Yes,YES 中的任意一个返回：Good Morning
和Up bright and early this morning 两条语句.

输入以 n 或 N 开头的字符串，返回：Good Afternoon

输入其他字符串，返回：Sorry, answer not recognize
和Please answer yes or no 两条语句。

解析：
这段代码不仅改动了加入了多条语句的返回，同时还改变了 no 情况下的匹配方法，并且给出了另外一个退出码，让脚本程序具有可重复性。

通过以上三条示例，从简单到复杂依次展示了 case 结构的特点，并且进行了测试。我们发现尽管 case 结构的语法看起来稍显复杂，但是其效果却十分强大，可以替代大量的 if，elif，else 等判断语句，使得脚本程序的代码量大大缩减。* 通配符的使用，也让程序的容错性提高了很多。