Covariant Script 编程语言

基础教程

零：简介

Covariant Script 诞生于2017年初，是一门动态类型脚本语言。

第一部分：语言核心

一、标准输入与输出流

1、Hello，World！

#例 1.1.1

```
system.out.println("Hello, World!")
```

作业：下载 Covariant Script GUI，运行示例程序观察效果。

我们可以看到示例程序在窗口中输出了文字"Hello, World!"

system.out.println 被称为"标准格式化输出函数"

"Hello, World!"是一个字符串常量，在 Covariant Script 中将双引号括起的任意长度文字称为字符串常量，简称为字符串。

提示：在程序中 Hello 和 "Hello" 有什么区别？

若文字未被双引号括起，那么这段文字就是程序指令的一部分；

但被双引号括起后，这段文字就不会被视为指令，从而能够用于

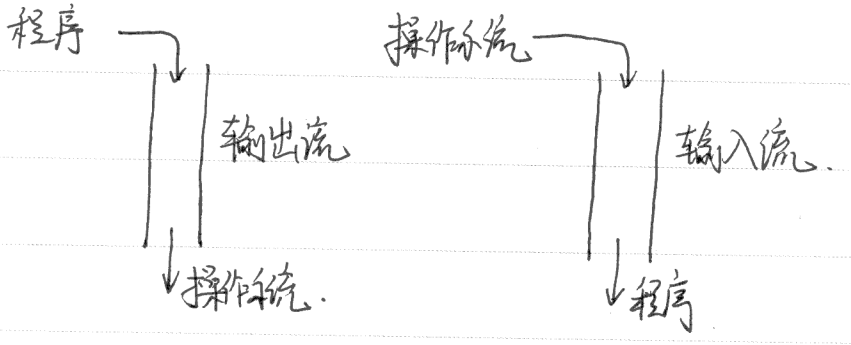
表示一段数据——我们称之为字符串常量

在例1.1.1中我们使用一个字符串常量作为参数调用标准格式化输出函数，将一段文字输出到窗口中。

2. ~~标准输出流~~

在计算机中程序运行的速度往往远高于输入或输出的速度。也就是说，输入时可能由于程序运行过快而丢失一部分数据，输出时可能由于程序运行过快而覆盖了之前输出的数据。

为了避免输入输出与程序运行不同步导致的错误，科学家们引入了一块缓冲区（Cache）用于暂时存储数据。由于对缓冲区的操作总是从一边进，从另一边出，所以将其形象地比喻为“流”。对于这种线性的先进先出数据结构，我们常称之为“队列”。队列与流比较相似，但不完全相同，我们以后再讲。



输入输出流的模型

流的特点之一就是下游的能若需要数据就会阻塞进程并等待，而上游的能不需要关心下游的状态都可进行操作，由此实现了同步。

3. 标准输出流：system.out

若不进行重定向，标准输出流的目的方向一般是命令提示行终端。

提示：在Windows等图形操作系统中每运行一个程序都相当于打开一个新的终端

输出流有很多方法（Method），我们今天只介绍两个格式化输出函数

① <输出流>.print(<变量>).

② <输出流>.println(<变量>).

①和②的区别只有①在输出后不换行而②在输出后会换行。

作业：尝试使用更多方法输出"Hello, World!"

4. 输出问候语

并例 1.4.1

```
var name = system.in.input().
system.out.print("Hello,").
system.out.println(name).
```

作业：运行示例程序，观察效果。。

如果我们在程序中键入"CarScript"并回车，我们将看到程序输出了文字"Hello, CarScript"

System.in.input被称为"标准格式化输入函数"，这个函数不需要任何参数，所以我们使用空括号调用它。。

"var name="的作用是将函数执行的结果保存到一个新变量中，关于变量的相关知识我们以后再讲。。

在例1.4.1中，我们调用标准格式化输入函数得到一个用户的输入并将其存放在一个变量中，然后再得到标准输出一个字符串常量和用户的输入组成一条问候语。。

4. 标准输入流：System.in.

若不进行重定向，标准输入流的源头一般是命令提示行终端。。

标输入流也有很多方法，我们今天只介绍两个函数：

① <输入流>.input()　　格式化输入函数。

② <输入流>.getline()　　获取输入流中的一行字符串。

提示：什么叫做格式化？与非格式化有何区别？

格式化指的是对源数据进行一定处理。如：

3.1415926 格式化 "3.1415926"

在输入时，格式化函数会把源数据中的字符串解析至合适的数据；在输出时，格式化函数会把源数据解析为字符串，交给操作系统处理。

而非格式化不会对源数据作任何处理，如②，将会直接获取输入流中的一行字符串，也就是说，"3.1415926"并不会变成可以进行运算的数字。

我们在第2节讲过，当流为空而下游又需要数据时，会阻塞进程并等待。也就是说，当调用①或②时，若输入流为空，程序会"暂停"以等待输入，直到用户输入一定内容并按回车提交。

二、变量、类型与作用域

1、变量的概念

变量在计算机科学中指的是就储存运算结果的一个抽象概念。

变量可以通过变量名来访问。

2、类型的概念

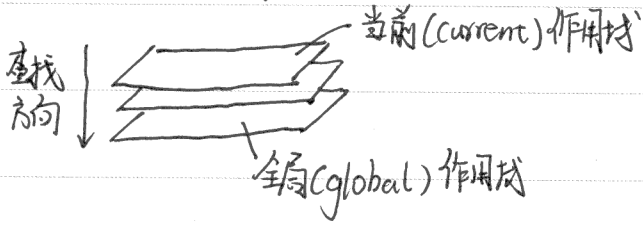
变量存储的运算结果根据其特征进行了分类，方便处理。

所以类型(Type)就是变量存储的数据类型。

3. 作用域以及作用域模型.

作用域(Domain)指的是一个变量的生命周期。

作用域的模型类似于栈(stack),由一层一层名称空间(namespace)堆叠而成,如下图:



每当离开一个作用域时,系统会抽掉最顶层名称空间并消毁其中的所有变量。

4. 变量及变量定义

要访问一个变量,只需写出变量名即可。

访问特定名称空间中的变量,使用 <名称空间名>.<变量名> 进行访问。

变量在使用前必须定义,定义变量的语法为:

var <变量名> = <表达式>

其中,表达式的值是变量的初始值。

若要新建一个特定类型的变量,使用 new 运算符:

new <类型>

CorScript 内建许多常用类型，这里只介绍几个简单的：

number：数值类型，表示数字中的全部有理数。

boolean：逻辑类型，表示布尔量的 true（真）和 false（假）。

string：文字（字符串）类型，表示程序中的文字

这里介绍的三种类型都有其字面量：

形似 3.14 等由数字和小数点组成的是数值字面量；

true 和 false 是逻辑字面量；

由双引号括起的任意内容组成的是字符串字面量。

关于类型的其他定义，参见《CorScript 参考文档》

书例 2.4.1

```
var a=3.1415926
var b=true
var c="Hello"
var d=new number
var e=new boolean
var f=new string
```

在这里，a 和 d 类型都是数值，b 和 e 类型都是逻辑，c 和 f 类型都是字符串。

5. 在程序中变量的作用

在例 1.4.1 中我们就用到了变量，这里变量的作用为暂存程序运行结果。

一般的程序都有输入和输出，所以我们可以将程序的作用总结为"处理用户输入并输出处理结果"。所以往往我们在编写程序时并不知道用户会输入什么，只能使用一个"未知量"来表示。但在计算机中我们将"未知量"称为"变量"。

抽象能力是评价一个程序员的重要指标，越是能够抽离无关因素的程序越强大。最基础的抽象就是使用变量代替具体的数据，或是使用变量抽象出计算过程。例如：

我们常见的二次函数，广义上是指自变量只含有二次和一次的函数。但前人总将其归约为 ax^2+bx+c 的形式，这便是一次"抽象"，将二次函数的其他细节全部抽离，只保留 a、b、c 三个参数用于处理。

接下来在求解时我们就需要考虑根的存在情况，分别是：

当 $b^2-4ac>0$ 时，存在两个解；

当 $b^2-4ac=0$ 时，存在两个相等的解；

当 $b^2-4ac<0$ 时，无解。

再在后来解时我们要用到求根公式 $\frac{\sqrt{b^2-4ac}-b}{2a}$

我们通过观察会发现，“b^2-4ac”这个表达式重复出现了很多次，而且其值在计算过程中不会改变，所以我们就可以将“b^2-4ac”的值保存到一个变量中，由此我们就能简化程序，减少运算量了。

#例 2.5.1

#首先，提醒用户分别输入方程的 a、b、c。

```
system.out.print("Please enter the a,b,c:").
var a=system.in.input().
var b=system.in.input().
var c=system.in.input().
```

#其次，提前计算出 deltu。

```
var deltu=b^2-4*a*c
```

#当 deltu>0 时，有两个解

```
if deltu>0
    var x1=(deltu^0.5-b)/(2*a).
    var x2=(-deltu^0.5-b)/(2*a).
    system.out.println(x1).
    system.out.println(x2).
end
```

```
#当delta=0时，有一个解。
if delta==0
    var x=(delta^0.5-b)/(2*a).
    system.out.println(x).
end.
#当delta<0时，无解。
if delta<0
    system.out.println("No Answer")
end.
```

例2.5.1是一个解一元二次方程的程序，让我们先强调几点注意事项：

① 在CorScript中，乘是"*"，除是"/"，幂是"^"

要注意运算符优先级，更多详情参考<<CorScript参考文档>>.

② 在CorScript中，"="的作用是赋值，所以比较时要使用"==".

③ if…end是流控制语句中的分支语句，作用为判断条件是否成立，如成立则执行end之前的语句。

思考：

- 若将 deltα 省去，b^2-4*a*c 会重复计算多少次？
- 若将 x1、x2、x 省去，程序有什么变化？是否应该省去？
- 此程序能否不用任何变量实现？

首先，很明显无论如何 a、b、c 这三个变量是必不可少的。就算不考虑求根公式中多次用到同一个值的情况，难不成你想要这样计算 delta？

(System.in.input())^2-4*(System.in.input())*(System.in.input())

真是心痛你的键盘两秒。

其次，要知道计算机运算也是需要时间的，省去 delta，无异于增加了计算机运算的负担。而且，说实话，delta 打起来不觉比 b^2-4*a*c 更省力吗？

最后，我相信你不会喜欢每行长度都一眼看不到边的程序，x1、x2 和 x 的作用纯粹是增加程序可读性，但如果你实在太懒：

System.out.println((b^2-4*a*c)^0.5-b)/(2*a))

我也没什么话说，不是吗？

作业：找出三个数中的最大值和最小值。