豫言编程语言教程

- 豫言编程语言简介

初级教程

- 第一章-教程序言
- 第二章-数据与函数
- 第三章-类型
- 第四章-更深一步了解函数
- 第五章-自建数据类型
- 第六章-入门教程结语

豫言编程语言简介

综述

设计理念

豫言中文编程语言以中文编程为核心，以现代化编译器框架 LLVM 为基础，吸取函数式编程领域数十年来的语言设计经验，自主研发，实现了从顶部语法，编译设计，代码生成的全中文编程环境。豫言编译器将全中文的源码，通过一系列编译步骤，生成了完全使用中文标识符 LLVM 后端码，最终由 LLVM 编译器框架生成后端执行程序。豫言编译器本身也使用了豫言编程语言实现，证明了豫言语言设计可以被用来构建大型程序，是众多编程语言以外企业和个人的又一项选择。

核心特征

与其他语言相比，豫言有着独特的风格。与其余大部分中文编程语言相比，豫言基于函数式编程，开创性地采用了依值类型系统，从根本上增强了语言的安全性与可靠性，也从某种程度上增加了软件开发效率。与同类型的英文编程语言相比，豫言的出现减少了语言学习的门槛，使得广大软件行业从业者乃至青少年不需要借助英文就可以学习和使用先进的编程语言范式，同时这些范式在豫言中拥有更直接的表达，这对于汉语在计算机行业及编程语言相关技术的发展有着直接的促进作用。

历史展望

豫言编程语言不是第一个，也不会是最后一个中文编程语言，我们已经看到有新的中文编程语言（例如入墨答语言）借用了豫言编程语言中的一些设计。我们希望全新设计的豫言编程语言成为集当今优秀的编程语言设计于一体，能够兼顾工业生产、人才教育、科学研究的一门编程语言。未来一定会有新的编程语言出现，我们希望通过豫言成为他们设计时的可靠参考。

为何设计中文编程语言?

中文有源远流长的历史，丰富的文化内涵。这样一门语言，在科技上仍然看不到广泛的使用，定有其特殊的原因。中文编程语言的设计虽然经历了照搬翻译英文关键字，自主设计语法，到使用本土表达的发展阶段，但目前仍有实用方面未能够解决的问题。我希望借本项目，探索中文在编程语言设计领域的各种问题，最终展现中文在编程语言行业的魅力。

第一章：教程序言

豫言是一门极简单又深入的语言。我们与大家分享我在豫言编程语言设计方面的思考，希望以此在中文编程语言的设计上继往开来，推陈出新。如有错误不周之处，恳请读者予以斧正。

语法

豫言有简略的文言语法，和易懂的现代汉语语法。文言语法主要用于实际开发，现代汉语语法主要用于教学。两种语法一一对应，详见附录语法表。本教程将使用现代汉语语法书写，豫言编译器本身使用文言语法书写。读者最终可以自由选择两种语法之一使用。

如何运行代码

代码有两种运行方式，一是使用在线 IDE（目前仍在开发中），二是在本地安装程序，并使用 VSCode 等文本编辑器编辑代码，使用命令行运行代码。

本教程大多数情况下需要依赖标准库，运行代码时请在文件顶部加上

```
导 入 并 打 开 标 准 库 。
```

第二章：数据与函数

编程的本质是对于数据的操作，操作数据的结构被称为函数。我们可以调用一个函数并计算对于给定输入的输出，函数的计算过程是可以与环境交互的，例如访问网络或打印调试信息。

基础数据：字符串

字符串是一串字符的在代码里的表示，使用一对 『』 符号表示。例如『 你好 』表示字符串“你好”。

输入提示　中文输入法下使用`Ctrl+{`或`Ctrl+}`可以输入 「『或』」 。

单参数函数调用

函数可以被**使用**。我们最基础的函数是“打印行”，在求值时会打印参数字符串到控制台上。例如，以下的代码将打印“你好”到控制台上。

```
打 印 行 使 用 于 『 你 好 』 。
```

基础数据：整数

一串仅包含数字零到九的标识符表示一个整数，例如整数十二的表示是「 一二 」，整数三百六十五的表示是「 三六五 」。符号 「」 用来表示单独的名字或标识符，在没有歧义时也可省略。

打印整数

之前的 打印行 仅可以打印字符串，所以如果要打印整数，需要先将整数转换为字符串。我们使用函数“整数表示”将整数转换为字符串。

```
打 印 行 使 用 于 （ 整 数 表 示 使 用 于 一 二 三 ） 。
```

我们用中文括号来有机地组合程序部件。在上例中，括号告诉我们必须先执行“整数表示”，取得字符串后才能打印。避免了与下文多参数函数调用产生歧义。

多参数函数调用

多参数函数是可以被多次使用函数，例如整数加法函数“加”的参数有两个，所以要两数相加，必须使用两次。例如我们使用如下表达式计算一加二的值：

```
「 加 」 使 用 于 「 一 」 使 用 于 「 二 」 。
```

在没有歧义的情况下，我们也可以省略括号。

```
加 使 用 于 一 使 用 于 二 。
```

要打印结果：

```
打 印 行 使 用 于 （ 整 数 表 示 使 用 于 （ 加 使 用 于 一 使 用 于 二 ） ） 。
```

使用临时命名

有时我们会发现句子很长，不利于阅读，我们可以将表达式存在一个名称或标识符中，并在接下来的计算中利用它，例如以下代码将一加二的值存在名为“结果”的标识符里，并在随后的打印函数调用中使用了“结果”。

```
让 结 果 为 加 使 用 于 一 使 用 于 二 随 后
    打 印 行 使 用 于 （ 整 数 表 示 使 用 于 （ 结 果 ） ） 。
```

使用定义

我们可以把以上的句子拆成两个句子。第一个句子定义了“结果”。

```
结 果 的 定 义 是 加 使 用 于 一 使 用 于 二 。
打 印 行 使 用 于 （ 整 数 表 示 使 用 于 （ 结 果 ） ） 。
```

模块

豫言中，一个文件就是一个模块，它由一系列声明构成。声明可以是定义声明，也可以是一个表达式。在接下来的教程中，我们可以看到更多的声明形式。

第三章：类型

类型描述了物体的性质。豫言中，每个物体都有其类型。

我们可以用类型声明来表示一个物体的类型。例如，我们可以标注结果的类型。

```
结 果 的 类 型 是 整 数 。
```

```
结 果 的 定 义 是 加 使 用 于 一 使 用 于 二 。
```

类型标注是一个良好的编程习惯。在编译时，编译器会检查所有的类型标注，并协助你改正那些不合理的程序。例如，编译器会对如下的程序提示类型错误：

```
结 果 的 类 型 是 整 数 。
结 果 的 定 义 是 『 你 好 』 。
```

我们接下来介绍豫言的一些类型，以及对应的数据对象。

基础类型

基础类型包括整数，小数和字符串。

它们之间可以通过 表示 或 理解 等操作相互转换。这些函数默认使用阿拉伯数字，在豫言程序源码中才使用汉字数字。

```
结 果 二 的 类 型 是 字 符 串 。
结 果 二 的 定 义 是 整 数 表 示 使 用 于 一 二 三 。

打 印 行 使 用 于 结 果 二 。「： 会 打 印 123：」

结 果 三 的 类 型 是 整 数 。
结 果 三 的 定 义 是 整 数 理 解 使 用 于 『123』。

打 印 行 使 用 于 （ 整 数 表 示 使 用 于 （「 减 」 使 用 于 结 果 三 使 用 于 一 二 三 ））。 「： 会 打 印
```

提示：用 「：：」 包裹的字符为注释，编译器会完全忽略这些注释。

对子类型

对子类型是将两个类型**结合**成一对。一个对子则是把两个元素**连结**起来。

```
对 子 甲 的 类 型 是 整 数 结 合 字 符 串 。
对 子 甲 的 定 义 是 一 二 三 连 结 『 你 好 』 。
```

可以从对子**中**可以选取需要的**第**几**个**元素，注意：豫言中的序数一律从零开始。

```
乙 的 类 型 是 整 数 。
乙 的 定 义 是 对 子 甲 中 的 第 零 个 。「： 乙 的 值 是 123：」

丙 的 类 型 是 字 符 串 。
丙 的 定 义 是 对 子 甲 中 的 第 一 个 。「： 丙 的 值 是 『 你 好 』：」
```

我们也有三联对，四联对。

```
碰 的 类 型 是 整 数 结 合 字 符 串 结 合 整 数 。
碰 的 定 义 是 一 二 三 连 结 『 你 好 』 连 结 四 五 六 。
```

```
杠 的 类 型 是 整 数 结 合 字 符 串 结 合 整 数 结 合 字 符 串 。
杠 的 定 义 是 一 二 三 连 结 『 你 好 』 连 结 四 五 六 连 结 『 七 八 九 』 。
```

我们可以自定义函数，例如我们可以用加法定义乘积

```
乘 的 类 型 是 从 整 数 到 从 整 数 到 整 数 。
乘 的 类 型 是 遇 到 了
```

函数类型

函数**从**一个类型**到**另一个类型的变换，我们之前已经使用过以下的函数

```
整 数 表 示 的 类 型 是 从 整 数 到 字 符 串 。
整 数 理 解 的 类 型 是 从 整 数 到 字 符 串 。
打 印 行 的 类 型 是 从 字 符 串 到 有 。
```

有是一个基础类型，表示一些基础操作的返回类型。

函数也可以被定义，我们只需要定义当我们**遇到了**某个对象时，**随后**需要怎么做。例如（翻一番）

```
翻 一 番 的 类 型 是 从 整 数 到 整 数 。
翻 一 番 的 定 义 是 遇 到 了 数 随 后 「 加 」 使 用 于 数 使 用 于 数 。
```

多参数函数是连续的从某些类型到最终类型的变换，比如

```
加 的 类 型 是 从 整 数 到 从 整 数 到 整 数 。
乘 的 类 型 是 从 整 数 到 从 整 数 到 整 数 。
```

在定义多参数函数的时候，我们要表明在我们连续**遇到了**某些事物的时候要怎么反应，比如

```
平 方 和 的 类 型 是 从 整 数 到 从 整 数 到 整 数 。
平 方 和 的 定 义 是
    遇 到 了 数 甲 随 后
    遇 到 了 数 乙 随 后
        让 甲 乙 和 为 「 加 」 使 用 于 数 甲 使 用 于 数 乙 随 后
        「 乘 」 使 用 于 甲 乙 和 使 用 于 甲 乙 和 。
```

我们可以巧妙地使用换行，空格和缩进让我们的代码变得容易阅读。空格本身并不影响程序的意思。

模块类型

一个模块就是是一系列声明，一个模块的类型是一系列类型声明，比如

```
加 法 操 作 模 块 的 类 型 是
    「
        翻 一 番 的 类 型 是 从 整 数 到 整 数 。
        平 方 和 的 类 型 是 从 整 数 到 从 整 数 到 整 数 。
    」 。
加 法 操 作 模 块 的 定 义 是
```

```
「
    翻 一 番 的 类 型 是 从 整 数 到 整 数 。
    平 方 和 的 定 义 是
        遇 到 了 数 甲 随 后
        遇 到 了 数 乙 随 后
            让 甲 乙 和 为 「 加 」 使 用 于 数 甲 使 用 于 数 乙 随 后
            「 乘 」 使 用 于 甲 乙 和 使 用 于 甲 乙 和 。
」 。
```

结语

豫言提供了完善的类型工具，我们强调的不是完全使用类型，也不是完全不用类型，而是根据可以根据情况来合理地使用类型系统。

更深一步了解函数

自然的函数调用语法

我们已经了解如何使用函数对数据进行操作，比如计算 $3+2,$。在豫言中，我们还可以使用用更加口语化的表达来使用函数。例如，我们或许直接说：

```
三 加 二 。
```

我们通过在函数名中插入〇来表示参数的出现位置，豫言编译器会自动理解把写在一起的句子并转换成函数调用。比如，要实现上例的效果，我们可以写

```
「〇 加 〇」 的 定 义 是 「 加 」 。
```

豫言标准库中对于数值计算提供了以下计算函数

```
「〇 加 〇」 的 类 型 是 从 整 数 到 从 整 数 到 整 数 。
「〇 减 〇」 的 类 型 是 从 整 数 到 从 整 数 到 整 数 。
「〇 乘 〇」 的 类 型 是 从 整 数 到 从 整 数 到 整 数 。
「〇 等 于 〇」 的 类 型 是 从 整 数 到 从 整 数 到 爻 。
```

爻是一个基础类型，它的值可以是阴（表否定），或者是阳（表肯定）。

巧妙地使用〇符号可以让我们的代码容易阅读，比如我们可以定义自己的特殊语法形式。

```
〇 的 〇 倍 的 类 型 是 从 整 数 到 从 整 数 到 整 数 。
「〇 的 〇 倍 」 的 定 义 是
    遇 到 了 底 数 随 后
    遇 到 了 倍 数 随 后
        底 数 乘 倍 数 。
```

我们可以用它来计算

```
打 印 行 使 用 于 （ 整 数 表 示 使 用 于 （ 三 的 五 倍 ） ） 。
```

条件判断

有的时候，我们想要的操作会根据实际情况的不同而做一些调整，比如计算翻番时时，我们可以会根据遇到的不同的数，提前给出一些变化，例如，如果计算一个数的一倍时，我们可以直接跳过乘法计算：

```
〇 的 〇 倍 的 类 型 是 从 整 数 到 从 整 数 到 整 数 。
「〇 的 〇 倍 」 的 定 义 是
    遇 到 了 底 数 随 后
    遇 到 了 倍 数 随 后
        如 果 倍 数 等 于 零
            那 么 底 数
            否 则 底 数 乘 倍 数 。
```

递归函数

在声明了函数的类型后，我们可以在任何地方使用这个函数，包括在定义函数本身的时候。

例如我们可以计算番数。

```
把 〇 翻 〇 番 的 类 型 是 从 整 数 到 从 整 数 到 整 数 。
「把 〇 翻 〇 番 」 的 定 义 是
    遇 到 了 底 数 随 后
    遇 到 了 番 数 随 后
        如 果 番 数 等 于 零
            那 么 底 数
            否 则 （ 把 底 数 翻 （ 番 数 减 一 ） 番 ） 乘 二 。

打 印 行 使 用 于 （ 整 数 表 示 使 用 于 （ 把 三 翻 一 番 ） ） 。「： 结 果 6： 」
打 印 行 使 用 于 （ 整 数 表 示 使 用 于 （ 把 三 翻 二 番 ） ） 。「： 结 果 12： 」
打 印 行 使 用 于 （ 整 数 表 示 使 用 于 （ 把 三 翻 三 番 ） ） 。「： 结 果 24： 」
打 印 行 使 用 于 （ 整 数 表 示 使 用 于 （ 把 三 翻 五 番 ） ） 。「： 结 果 96： 」
```

自建数据类型

我们看过了一些基础类型，比如整数和字符串，为了建模现实世界中更加复杂的数据类型，我们自己建立类型。

自建类型分两步：先说什么是一种类型，再说它有哪些东西。比如，我们可以定义五行。

先说五行**是一种**类型：

```
五 行 是 一 种 类 型 。
```

再说五行有哪些：

```
木 是 一 种 五 行 。
火 是 一 种 五 行 。
土 是 一 种 五 行 。
金 是 一 种 五 行 。
```

```
水 是 一 种 五 行 。
```

用一样的方法，我们可以定义颜色，

```
颜 色 是 一 种 类 型 。
    青 是 一 种 颜 色 。
    赤 是 一 种 颜 色 。
    黄 是 一 种 颜 色 。
    白 是 一 种 颜 色 。
    黑 是 一 种 颜 色 。
```

对于数据进行计算

我们可以对自己定义的数据进行分析，并且对于每一种情况做出相应的反应。例如，我们可以写一个函数来获取五行的颜色，通过**分析**遇到的五行，我们做出对应的判断：**如果是**我们知道的某一种，**那么**怎么办，**或者如果是**我们知道的另一种，**那么**要怎么办。

```
五 行 转 颜 色 的 类 型 是 从 五 行 到 颜 色 。
五 行 转 颜 色 的 定 义 是
    遇 到 了 一 行 随 后
        分 析 一 行 随 后
            如 果 是 木 那 么 青
            如 果 是 火 那 么 赤
            如 果 是 土 那 么 黄
            如 果 是 金 那 么 白
            如 果 是 水 那 么 黑 。
```

同样的方法，我们可以把颜色转换成字符串，以供打印。

```
颜 色 转 字 符 串 的 类 型 是 从 颜 色 到 字 符 串 。
颜 色 转 字 符 串 的 定 义 是
    遇 到 了 颜 色 随 后
        分 析 颜 色 随 后
            如 果 是 青 那 么 『 青 』
            如 果 是 赤 那 么 『 赤 』
            如 果 是 黄 那 么 『 黄 』
            如 果 是 白 那 么 『 白 』
            如 果 是 黑 那 么 『 黑 』 。

打 印 行 使 用 于 （ 颜 色 转 字 符 串 使 用 于 白 ） 。 「： 会 打 印 白 ： 」
打 印 行 使 用 于 （ 颜 色 转 字 符 串 使 用 于 （ 五 行 转 颜 色 使 用 于 土 ） ） 。 「： 会 打 印 黄 ： 」
```

带参数的数据

构建自定义数据类型的时候我们获取想带上参数，这时候，可以构建一个函数。比如，表示时间：

```
时 间 是 一 种 类 型 。
```

```
〇 点 整 是 一 种 从 整 数 到 时 间 。
〇 点 半 是 一 种 从 整 数 到 时 间 。
```

以及时间上的计算

```
从 〇 到 〇 的 分 钟 数 的 类 型 是 从 时 间 到 从 时 间 到 整 数 。
从 〇 到 〇 的 分 钟 数 的 定 义 是
      遇 到 了 开 始 时 间 随 后
      遇 到 了 结 束 时 间 随 后
            分 析 开 始 时 间 随 后
                  如 果 是 「 开 始 时 刻 」 点 整 那 么
                        （ 分 析 结 束 时 间 随 后
                              如 果 是 「 结 束 时 刻 」 点 整 那 么
                                    （ 结 束 时 刻 减 开 始 时 刻 ） 乘 六 零
                              或 者 如 果 是 「 结 束 时 刻 」 点 半 那 么
                                    （ （ 结 束 时 刻 减 开 始 时 刻 ） 乘 六 零 ） 加 三 零
                        ）
                  或 者 如 果 是 「 开 始 时 刻 」 点 半 那 么
                        （ 分 析 结 束 时 间 随 后
                              如 果 是 「 结 束 时 刻 」 点 整 那 么
                                    （ （ 结 束 时 刻 减 开 始 时 刻 ） 乘 六 零 ） 减 三 零
                              或 者 如 果 是 「 结 束 时 刻 」 点 半 那 么
                                    （ 结 束 时 刻 减 开 始 时 刻 ） 乘 六 零
                        ） 。

打 印 行 使 用 于 （ 整 数 表 示 使 用 于 （ 从 三 点 半 到 五 点 整 的 分 钟 数 ） ） 。 「： 会 打 印 90： 」
```

入门教程结语

恭喜你已经完成了入门教程的学习！