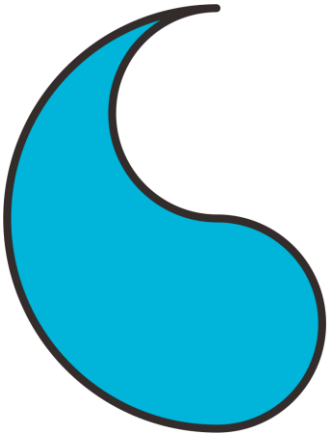

Covariant Script

Covariant Script 编程语言

参考文档(STD20170803)

语法参考

说明

本文档格式为：正文内容 是代码，*斜体内容* 是说明，**加粗内容** 是解释

引入 Package

import *Package 目录（文字常量）* **引入一个 Package**

声明 Package

package *Package 名* **声明一个 Package**

变量定义

var *变量名* **定义一个变量，初始值为 0**
var *变量名=表达式* **定义一个变量，初始值为表达式的值**
var *变量名* as *类型* **定义一个变量为指定类型**
const var *变量名=表达式* **定义一个常量，其值为表达式的值**

作用域与命名空间

block
    语句块
end
定义一个临时作用域
临时作用域中的变量会在离开作用域后销毁
block *命名空间名*
    语句块
end
定义一个命名空间

分支语句

if *逻辑表达式*
    语句块
end
逻辑表达式的值为真则执行语句块
if *逻辑表达式*
    语句块 1
else
    语句块 2
end
逻辑表达式的值为真则执行语句块 1
逻辑表达式的值为假则执行语句块 2
switch *表达式*
    case *常量标签*
        语句块
    end
    default
        语句块
    end
end
执行与表达式的值相等的常量标签对应的 case 中的语句块
当无匹配的常量标签时会执行 default 中的语句块，如 default 未找到则跳出

循环语句

while *逻辑表达式*
    语句块
end
当逻辑表达式的值为真时循环执行语句块
loop
    语句块
until *逻辑表达式*
end
直到逻辑表达式的值为真时跳出循环
loop
    语句块
end
循环执行语句块直到用户手动跳出
for *变量名=表达式* to *表达式*
    语句块
end
定义一个变量在闭区间中遍历，步长为 1
for *变量名=表达式* to *表达式* step *表达式*
    语句块
end
定义一个变量在闭区间中遍历，步长为用户指定的值
for *变量名* iterate *表达式*
    语句块
end
表达式的值必须是一个支持 for 遍历的容器
定义一个变量正序遍历容器

函数定义

function *函数名(参数列表（可选）)*
    语句块
end
定义一个函数
参数列表中的参数只能指定名称，参数名不可重复，各参数之间以逗号分隔，如：
function test(a0,a1,a2)

控制语句

break	**跳出循环**
continue	**进入下一轮循环**
return	**结束函数并返回 0**
return *表达式*	**结束函数并返回表达式的值**

结构定义

struct *结构名*
    结构体
end
结构定义后结构名就可以作为类型名使用
结构体中只允许变量定义和函数定义
结构的内存布局如图所示

作用域访问

变量名	**从最上层作用域开始向下查找变量**
current.*变量名*	**查找当前作用域中的变量**
global.*变量名*	**查找全局作用域中的变量**
this.*变量名*	**查找当前结构体中的变量**
名称空间名. 变量名	**查找名称空间中的变量**
变量名.变量名	**查找结构体或扩展中的变量**

作用与结构以及变量查找方式如图所示

注意：对于最后一种访问方法，仅变量类型为结构或支持扩展的类型时可用，如访问的是扩展中的函数，将会把点运算符左边的变量作为函数的第一个参数传入。
也就是说： types.char.isspace(ch) **等价于** ch.isspace()

运算符与表达式

表达式由操作数和运算符组成
操作数 运算符 操作数
一般有左右两个操作数的运算符是二元运算符
只有一个操作数的运算符是一元运算符
二元运算符有结合律，左结合是从右向左运算，右结合是从左向右运算
所有的运算符都有优先级，优先级越高越先计算

类型，函数及变量参考

全局

char 字符类型
number 数字类型
boolean 逻辑类型
string 文字类型
list 链表类型
array 数组类型
pair 映射类型
hash_map 哈希表类型
system 系统命名空间
runtime 运行时名称空间
math 数学名称空间
file 文件名称空间
darwin 图形名称空间
number to_integer(var) 将一个变量转换为整数并返回
string to_string(var) 将一个变量转换为文字并返回
var clone(var) 复制一个变量并返回
void swap(var,var) 交换两个变量的值

System 名称空间

max 数字类型最大值
inf 数字类型正无穷
var input(...) 从标准输入流中获取输入（堵塞，格式化输入）
void print(...) 向标准输出流中输出内容，仅可输出支持 to_string 的类型（不换行）
void println(...) 向标准输出流中输出内容，仅可输出支持 to_string 的类型（换行）
string getline() 从标准输入流中获取输入（堵塞，非格式化输入）
void setprecision(number) 设置输出精度（to_string 的精度）
number run(string) 在系统环境中运行一条指令，返回错误码
string getenv(string) 获取环境变量的值并返回
void exit(number) 清理资源并退出

Runtime 名称空间

void info() 输出版本信息
number time() 获取计时器的读数，单位毫秒
void delay(number) 使程序暂停一段时间，单位毫秒
number rand(number,number) 获取区间内的伪随机数
number randint(number,number) 获取区间内的伪随机整数
void error(string) 抛出一个运行时错误
[namespace] load_extension(string) 加载一个扩展并返回
[hash_value] hash(var) 计算一个变量的哈希值
[expression] build(string) 构建一个可用于计算的表达式
var solve([expression]) 计算一个表达式

字符类型扩展

boolean isalnum(char) 检查字符是否是字母或数字
boolean isalpha(char) 检查字符是否是字母
boolean islower(char) 检查字符是否是小写字母
boolean isupper(char) 检查字符是否是大写字母
boolean isdigit(char) 检查字符是否是数字
boolean iscntrl(char) 检查字符是否是控制字符
boolean isgraph(char) 检查字符是否是图形字符
boolean isspace(char) 检查字符是否是空白字符
boolean isblank(char) 检查字符是否是空格或 tab
boolean isprint(char) 检查字符是否是打印字符
boolean ispunct(char) 检查字符是否是标点符号
char tolower(char) 将字符转换为小写
char toupper(char) 将字符转换为大写

文字类型扩展

string append(string,var) 在尾部追加内容
string insert(string,number,var) 在指定位置处插入内容
string erase(string,number,number) 将范围内的字符删除
string replace(string,number,number,var) 将从指定位置开始的指定个数字符替换
string substr(string,number,number) 从指定位置截取指定长度的子文字
number find(string,string,number) 从指定位置开始从左向右查找一段文字
number rfind(string,string,number) 从指定位置开始从右向左查找一段文字
string cut(string,number) 从尾部删除指定长度的文字
void clear(string) 清空
number size(string) 获取字符个数

链表类型扩展

var front(list) 访问第一个元素
var back(list) 访问最后一个元素
[iterator] begin(list) 返回指向容器第一个元素的迭代器
[iterator] term(list) 返回指向容器尾端的迭代器
[iterator] forward([iterator]) 向前移动迭代器
[iterator] backward([iterator]) 向后移动迭代器
var data([iterator]) 访问迭代器指向的元素
boolean empty(list) 检查容器是否为空
number size(list) 返回容纳的元素数
void clear(list) 删除全部内容
[iterator] insert(list,[iterator],var) 插入元素，插入到迭代器指向的元素之前，返回指向插入的元素的迭代器
[iterator] erase(list,[iterator]) 删除元素，返回指向要删除的元素的下一个元素的迭代器
void push_front(list,var) 在容器的开始处插入新元素
void pop_front(list) 删除第一个元素
void push_back(list,var) 将元素添加到容器末尾
void pop_back(list) 删除最后一个元素
void remove(list,var) 删除所有与指定变量相等的元素
void reverse(list) 将该链表的所有元素的顺序反转
void unique(list) 删除连续的重复元素

数组类型扩展

var at(array,number) 访问指定的元素，同时进行越界检查
var front(array) 访问第一个元素
var back(array) 访问最后一个元素
[iterator] begin(array) 返回指向容器第一个元素的迭代器
[iterator] term(array) 返回指向容器尾端的迭代器
[iterator] forward([iterator]) 向前移动迭代器
[iterator] backward([iterator]) 向后移动迭代器
var data([iterator]) 访问迭代器指向的元素
boolean empty(array) 检查容器是否为空
number size(array) 返回容纳的元素数
void clear(array) 删除全部内容
[iterator] insert(array,[iterator],var) 插入元素，插入到迭代器指向的元素之前，返回指向插入的元素的迭代器
[iterator] erase(array,[iterator]) 删除元素，返回指向要删除的元素的下一个元素的迭代器
void push_front(array,var) 在容器的开始处插入新元素
void pop_front(array) 删除第一个元素
void push_back(array,var) 将元素添加到容器末尾
void pop_back(array) 删除最后一个元素
list to_list(array) 将数组转换为链表

映射类型扩展

var first(pair) 获取第一个元素
var second(pair) 获取第二个元素

哈希表类型扩展

boolean empty(hash_map) 检查容器是否为空
number size(hash_map) 返回容纳的元素数
void clear(hash_map) 删除全部内容
void insert(hash_map,var,var) 插入一对映射
void erase(hash_map,var) 删除键对应的映射
var at(hash_map,var) 访问指定的元素，同时进行越界检查
boolean exist(hash_map,var) 查找是否存在映射

数学名称空间

pi 圆周率
e 自然底数
number abs(number) 绝对值
number ln(number) 以 e 为底的对数
number log10(number) 以 10 为底的对数
number log(number a,number b) 以 a 为底 b 的对数
number sin(number) 正弦
number cos(number) 余弦
number tan(number) 正切
number asin(number) 反正弦
number acos(number) 反余弦
number atan(number) 反正切
number sqrt(number) 开方
number root(number a,number b) a 的 b 次方根
number pow(number a,number b) a 的 b 次方
number min(number a,number b) a 和 b 的最小值
number max(number a,number b) a 和 b 的最大值

文件名称空间

read_method 读文件
write_method 写文件
[file] open(string path,[method]) 打开一个文件
boolean is_open([file]) 判断文件是否打开
boolean eof([file]) 判断是否到达文件结尾
string getline([file]) 从文件中获取输入（阻塞，非格式化输入）
var read([file],...) 从文件中获取输入（阻塞，格式化输入）
var write([file],...) 向文件中输出内容，仅可输出支持 to_string 的类型（不换行）

图形名称空间

black 黑色
white 白色
red 红色
green 绿色
blue 蓝色
pink 粉色
yellow 黄色
cyan 青色
[pixel] pixel(char,[color] front,[color] back) 创建一个像素
[drawable] picture(number width,number height) 创建一幅图片
void load(string path) 加载 Darwin 功能
void exit(number code) 退出程序并清理资源
boolean is_kb_hit() 判断是否有按键按下
char get_kb_hit() 获取按下的按键
void fit_drawable() 使画布适合当前屏幕大小
[drawable] get_drawable() 获取画布
void update_drawable() 将画布中的内容更新至屏幕上
void set_frame_limit(number fps) 设置帧率
void clear_drawable([drawable]) 清空画布
void fill_drawable([drawable],[pixel]) 填充画布
void resize_drawable([drawable],number width,number height) 重新设置画布大小
number get_width([drawable]) 获取画布宽度
number get_height([drawable]) 获取画布高度
void draw_pixel([drawable],nuber x,number y,[pixel]) 在画布上画点
void draw_picture([drawable],number x,number y,[drawable]) 将一幅图片绘制到画布上
void draw_line([drawable],number x1,number y1,number x2,number y2,[pixel])
在画布上画线
void draw_rect([drawable],number x,number y,number width,number height,[pixel])
在画布上绘制线框
void fill_rect([drawable],number x,number y,number width,number height,[pixel])
在画布上填充矩形
void draw_triangle([drawable],number x1,number y1,number x2,number y2,number x3,number y3,[pixel])
在画布上绘制三角形
void fill_triangle([drawable],number x1,number y1,number x2,number y2,number x3,number y3,[pixel])
在画布上填充三角形
void draw_string([drawable],number x,number y,string,[pixel]) 在画布上绘制文字
void message_box(string title,string message,string button) 弹出一个消息对话框
var input_box(string title,string message,string default,boolean format) 弹出一个输入对话框