atitit 编程语言之道attilax著.docx

[1. 编程语言常见概念与理论 5](#_Toc26092)

[1.1. solid原则 5](#_Toc21199)

[1.2. 变量6属性 5](#_Toc29396)

[1.3. 三大流程 5](#_Toc8490)

[1.4. 异常控制流程 5](#_Toc25510)

[1.5. 流程图（Flow Chart 5](#_Toc2474)

[1.6. 多分支结构 5](#_Toc3425)

[1.7. 表达式（中缀前缀表达式 5](#_Toc9829)

[1.8. 运算符 5](#_Toc7824)

[1.9. 数据类型 5](#_Toc7512)

[1.10. Lambda 表达式 5](#_Toc6228)

[1.11. 方法引用 6](#_Toc31189)

[1.12. 生命周期 6](#_Toc29857)

[1.13. Cohesion 6](#_Toc8770)

[1.14. 耦合Coupling 6](#_Toc2945)

[1.15. 过程 6](#_Toc21716)

[1.16. 函数 6](#_Toc5979)

[1.17. 返回值 6](#_Toc11059)

[1.18. 循环(loop), 6](#_Toc30868)

[1.19. 递归(recursion), 6](#_Toc25639)

[1.20. 遍历(traversal), 6](#_Toc6909)

[1.21. 迭代(iterate). 7](#_Toc27580)

[1.22. 泛型 7](#_Toc13292)

[1.23. 环境变量 7](#_Toc7555)

[1.24. s-exp 7](#_Toc1529)

[1.25. 类型系统 7](#_Toc22999)

[2. 编程语言种类 and趋势 逻辑式语言..函数式语言...命令式语言 7](#_Toc32685)

[2.1. SQL既是自含式语言，又是嵌人式语言 7](#_Toc3663)

[3. 语言实现 10](#_Toc29864)

[3.1. 词法分析 10](#_Toc4810)

[3.2. fsm状态机 10](#_Toc27920)

[3.3. 正则表达式 10](#_Toc1964)

[3.4. 词性标注 10](#_Toc9533)

[3.5. 语法分析 10](#_Toc1870)

[3.6. ast 10](#_Toc26622)

[3.7. 语义分析 10](#_Toc19379)

[3.8. 内部DSL实现模式 10](#_Toc32760)

[3.9. 外部DSL 10](#_Toc13041)

[3.10. 生成AST 10](#_Toc8414)

[3.11. BNF 11](#_Toc8189)

[3.12. 正则表达式表的词法分析器 11](#_Toc28572)

[3.13. 递归下降法语法解析器 11](#_Toc25273)

[3.14. 解释器 11](#_Toc18603)

[3.15. 表达式生成器 11](#_Toc26985)

[3.16. 嵌套函数 11](#_Toc7714)

[3.17. 方法级联 11](#_Toc20234)

[3.18. 状态机fsm 11](#_Toc21321)

[4. 函数式理论 章节 12](#_Toc13593)

[4.1. lambda 12](#_Toc27055)

[4.2. stream 12](#_Toc3273)

[4.3. fluce接口 12](#_Toc13337)

[4.4. 连锁接口 12](#_Toc21182)

[4.5. pipe 12](#_Toc9668)

[4.6. map & reduce 12](#_Toc6594)

[4.7. pipeline 12](#_Toc18635)

[4.8. 高阶函数 12](#_Toc25622)

[4.9. 闭包 12](#_Toc31924)

[4.10. 表达式特化 12](#_Toc2119)

[4.11. 表达式合成 13](#_Toc26227)

[4.12. 外循环 13](#_Toc1459)

[4.13. 内循环 13](#_Toc1372)

[4.14. 惰性求值 13](#_Toc3610)

[5. 类与对象实现 章节 14](#_Toc15583)

[6. 章节 15](#_Toc29949)

[7. dsl 章节 15](#_Toc24846)

[8. 常见类库api=---------------- 15](#_Toc31374)

[8.1. 源码组织管理 章节 15](#_Toc1064)

[8.2. 标准库 章节 15](#_Toc22394)

[8.3. io输入输出 章节 15](#_Toc30158)

[8.4. 重构 章节 章节 16](#_Toc22504)

[8.5. 开发规范 章节 16](#_Toc10376)

[8.6. 序列化 章节 16](#_Toc1869)

[8.7. ioc 章节 16](#_Toc6337)

[8.8. 性能提升 章节 16](#_Toc15371)

[9. other 章节 16](#_Toc26731)

[9.1. ide 章节 16](#_Toc2301)

[9.2. exop互操作接口 章节 17](#_Toc15586)

[9.3. 常用协议 章节 17](#_Toc3157)

[9.4. 常用web服务器 章节 17](#_Toc6345)

[9.5. 代码管理与同步 章节 17](#_Toc3704)

[9.6. 架构知识 章节 17](#_Toc31223)

[9.7. 图像处理 章节 17](#_Toc14532)

[9.8. uml 章节 17](#_Toc946)

[9.9. web体系 章节 17](#_Toc11096)

[9.10. 简单的图像处理 章节 17](#_Toc32708)

[9.11. vm 章节 17](#_Toc12893)

[10. bug管理 章节 18](#_Toc5419)

[11. api标准化 章节 19](#_Toc7780)

[11.1. 核心core标准化 19](#_Toc18814)

[11.2. 文本处理 19](#_Toc8398)

[11.3. 网络处理 19](#_Toc19000)

[11.4. 图像处理 19](#_Toc21594)

[11.5. 数据库处理 19](#_Toc9602)

[11.6. 进程处理 19](#_Toc6994)

[11.7. 鼠标键盘管理 19](#_Toc5690)

[11.8. 时间处理 19](#_Toc2028)

[11.9. 数字处理 19](#_Toc25927)

[11.10. 容器 20](#_Toc26837)

[12. 常用算法 章节 20](#_Toc12923)

[13. 软件工程 章节 20](#_Toc8968)

[14. 设计模式 章节 20](#_Toc18833)

[15. 标准化 章节 20](#_Toc2905)

[15.1. ecma标准 21](#_Toc3080)

[15.2. iso标准 21](#_Toc17756)

[15.3. apache fund 21](#_Toc32488)

[15.4. jcp jsr 21](#_Toc6808)

[15.5. gbk 21](#_Toc17343)

[15.6. rfc 21](#_Toc5146)

[15.7. w3c 21](#_Toc15945)

[15.8. ansi 21](#_Toc12778)

[15.9. omg 21](#_Toc14072)

[15.10. BCP，即Best Current Practice 21](#_Toc26139)

[16. 语言高级机制------------------ 22](#_Toc30058)

[16.1. 并发与锁机制 章节 22](#_Toc13449)

[16.2. api设计 章节 22](#_Toc9852)

[16.3. gc资源释放 章节 23](#_Toc8036)

[16.4. 调试技术 章节 23](#_Toc29268)

[17. 反射 章节 23](#_Toc692)

[18. 异常处理 章节 23](#_Toc12150)

[19. 事件机制 章节 24](#_Toc25405)

[19.1. event对象 25](#_Toc24242)

[19.2. 事件目标（event target 25](#_Toc6436)

[19.3. 事件传播（event propagation 25](#_Toc20430)

[19.4. event handler 25](#_Toc29828)

[19.5. 事件分发器 25](#_Toc31800)

[19.6. 事件委托 25](#_Toc21448)

[19.7. 事件代理 25](#_Toc20607)

[19.8. 事件循环 25](#_Toc8656)

[19.9. 事件驱动 vs 轮询机制 25](#_Toc8812)

[19.10. 事件队列 26](#_Toc8735)

[19.11. vs消息机制 26](#_Toc16612)

[19.12. 事件注册 26](#_Toc25514)

[19.13. 事件调用 26](#_Toc13322)

[19.14. 事件对象模型 26](#_Toc14085)

[19.15. 事件冒泡 26](#_Toc14131)

[19.16. 事件捕获 26](#_Toc11002)

[19.17. 事件底层 中断原理 26](#_Toc934)

[19.18. xml序列化 27](#_Toc27341)

[19.19. php hash序列化 27](#_Toc31303)

[19.20. json序列化 27](#_Toc29631)

[19.21. 二进制序列化 27](#_Toc27539)

[19.22. 循环引用解决 27](#_Toc911)

[19.23. base64 27](#_Toc30882)

[19.24. bin2txt 27](#_Toc28675)

# 编程语言常见概念与理论

|  |
| --- |
| solid原则 |
| 变量6属性 |
| 三大流程 |
| 异常控制流程 |
| 流程图（Flow Chart |
| 多分支结构 |
| 表达式（中缀前缀表达式 |
| 运算符 |
| 数据类型 |
| Lambda 表达式 |
| 方法引用 |
| 生命周期 |
| Cohesion |
| 耦合Coupling |
| 过程 |
| 函数 |
| 返回值 |
| 循环(loop), |
| 递归(recursion), |
| 遍历(traversal), |
| 迭代(iterate). |
| 泛型 |
| 环境变量 |
| s-exp |
| 类型系统 |

# 编程语言种类 and趋势 逻辑式语言..函数式语言...命令式语言

## SQL既是自含式语言，又是嵌人式语言

。作为自含式语言，它能够独立地用于联机交互的使用方式，用户可以在终端键盘上直接输入SQL命令对数据库进行操作。作为嵌入式语言，SQL语句能够嵌入到高级语言(如C、 C#、JAVA)程序中，供程序员设计程序时使用。而在两种不同的使用方式下，SQL的语法结构基本上是一致的。这种以统一的语法结构提供两种不同的操作方式，为用户提供了极大的灵活性与方便性

Atitit.编程语言的主要的种类and趋势 逻辑式语言..函数式语言...命令式语言p825

1. 编程语言的主要的种类 逻辑式语言..函数式语言...命令式语言 1

2. 命令式语言主要组成（运算符与控制结构，if，foreach） 2

3. 函数式语言（全部是函数） 2

4. 逻辑式语言,,不必考虑实现过程而只需考虑定义和结果 2

1. 控制结构将消失,select ,foreach 等...表格将取代select结构 3

5. 第五代语言 3

3.编程语言的主要的种类 逻辑式语言..函数式语言...命令式语言

在FAQ3里说过，世界上有C和LISP两种编程语言，你现在学了C，以后学了C++、Java、.NET

等等，也仍然只认识了半个世界。LISP和 Haskell构成了另外半个世

4.命令式语言主要组成（运算符与控制结构，if，foreach）

5.函数式语言（全部是函数）

6.逻辑式语言,,不必考虑实现过程而只需考虑定义和结果

例如著名的 prolog。逻辑式语言一般依据成熟的逻辑体系来设计，比如 first-order logic（用于 prolog），因为这样有几点好处：

1）可以进行严密的逻辑推理；

2）可以方便有效地定义 knowledge base，这在专家系统中很重要；

3）开发者不必考虑实现过程而只需考虑定义和结果。这符合专家系统的设计理念。

作者:: 老哇的爪子 Attilax 艾龙，  EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

6.1.控制结构将消失,select ,foreach 等...表格将取代select结构

只需考虑定义和结果,表格将取代select结构...

7.第五代语言

第五代语言就是自然语言又被称为知识库语言或人工智能语言，人工智能语言主要有LISP、Prolog

人们可能会问，用人工智能语言解决问题与传统的方法有什么区别呢？

传统方法通常把问题的全部知识以各种的模型表达在固定程序中，问题的求解完全在程序制导下按着预先安排好的步骤一步一步（逐条）执行。解决问题的思路与冯.诺依曼式计算机结构相吻合。当前大型数据库法、数学模型法、统计方法等都是严格结构化的方法。

对于人工智能技术要解决的问题，往往无法把全部知识都体现在固定的程序中。通常需要建立一个知识库（包含事实和推理规则），程序根据 环境和所给的输入信息以及所要解决的问题来决定自己的行动，所以它是在环境模式的制导下的推理过程。这种方法有极大的灵活性、对话能力、有自我解释能力和 学习能力。这种方法对解决一些条件和目标不大明确或不完备，（即不能很好地形式化，不好描述）的非结构化问题比传统方法好，它通常采用启发式、试探法策略 来解决问题。

其实，每种编程语言的设计都有

独到之处，体现了每种语言的精髓，在融汇百家之后积累下来的正是方法论。

# 语言实现

|  |
| --- |
| 词法分析 |
| fsm状态机 |
| 正则表达式 |
| 词性标注 |
| 语法分析 |
| ast |
| 语义分析 |
| 内部DSL实现模式 |
| 外部DSL |
| 生成AST |
| BNF |
| 正则表达式表的词法分析器 |
| 递归下降法语法解析器 |
| 解释器 |
| 表达式生成器 |
| 嵌套函数 |
| 方法级联 |
| 状态机fsm |

# 函数式理论 章节

|  |
| --- |
| lambda |
| stream |
| fluce接口 |
| 连锁接口 |
| pipe |
| map & reduce |
| pipeline |
| 高阶函数 |
| 闭包 |
| 表达式特化 |
| 表达式合成 |
| 外循环 |
| 内循环 |
| 惰性求值 |
| 尾递归 |
| yield |
| geneor |
| 一等函数 |
| [闭包或者 仿函数（functor）对](http://baike.baidu.com/view/2070037.htm" \o ") |
| 惰性计算（lazy evaluation |
| 递归 |
| 只用"表达式 |
| 尾递归，伪递归 |
| first-class types |
| generic types |
| pattern match |
| Guard |
| curry |
| immutable 、mutable |
| 引用透明（Referential transparency）的和没有副作用（No Side Effect）。 |
| continuation |

章节

# 类与对象实现 章节

|  |
| --- |
| class模板 |
| dock type |
| is-a has-a |
| mix in |
| oo三大特性 |
| solid原则 |
| 抽象方法 |
| 抽象类和 |
| 单根继承 |
| 单例 |
| 动态绑定（dynamic binding |
| 动态对象 |
| 对象持久性（object persistence |
| 对象实现 |
| 多继承 |
| 多态 |
| 反射api加载 |
| 方法重载 |
| 访问方法（accessor method）和 |
| 封装 |
| 覆盖（Overriding |
| 构造方法 |
| 构造函数 |
| 环境变量 |
| 基于原型 |
| 继承 |
| 接口用interface |
| 类加载 |
| 菱形继承 |
| 魔术方法 |
| 魔术属性 |
| 设置方法(mutator method) |
| 实例 |
| 属性读写 |
| 外部文件加载 |
| 析勾方法 |
| 消息 |
| 消息传递 |
| 延迟绑定方法 |
| 自动语句加载 |
| 字节数组加载 |

# 章节

# dsl 章节

# 常见类库api=----------------

## 源码组织管理 章节

章节

## 标准库 章节

## ioc 章节

## 性能提升 章节

章节

# other 章节

## ide 章节

章节

## exop互操作接口 章节

## 常用协议 章节

章节

## 常用web服务器 章节

## 代码管理与同步 章节

## 架构知识 章节

章节

## 图像处理 章节

章节

## uml 章节

## web体系 章节

章节

## 简单的图像处理 章节

## vm 章节

章节

# bug管理 章节

章节

# api标准化 章节

|  |
| --- |
| 核心core标准化 |
| 文本处理 |
| 网络处理 |
| 图像处理 |
| 数据库处理 |
| 进程处理 |
| 鼠标键盘管理 |
| 时间处理 |
| 数字处理 |
| 容器 |

# 常用算法 章节

# 软件工程 章节

章节

# 设计模式 章节

章节

# 标准化 章节

|  |
| --- |
| ecma标准 |
| iso标准 |
| apache fund |
| jcp jsr |
| gbk |
| rfc |
| w3c |
| ansi |
| omg |
| BCP，即Best Current Practice |

# 语言高级机制------------------

## 并发与锁机制 章节

|  |
| --- |
| Volatile 内存屏障 |
| cas |
| atomic |
| 同步关键字 |
| ReentrantLock |
| 自旋锁 |
| 信号量 |
| 互斥锁Mutex |
| 线程死锁 |
| 线程死锁kill |
| 异步 |
| 线程池 |
| task模型 |
| futuretask模式 |
| 毒丸 |
| threadlocalhost |

## api设计 章节

|  |
| --- |
| 方法连 |
| jquery |
| guava |
| option |
| 面向接口编程 |
| 工厂方法优于构造函数 |
| polyfill |
| dsl命名法 |
| 参考知名api |
| 归一化 |
| 命名参数 |
| 属性风格API |
| 类型自动转换 |
| 通用属性方法attr( like jq) |
| 事件（events） |
| 回调（callbacks） |
| api扩展性 |
| dsl化 |
| what how分离 |
| 函数式接口 |

## gc资源释放 章节

|  |
| --- |
| 分代 |
| 基于时间 |
| 引用计数 |
| amr |
| watchdog |

## 调试技术 章节

|  |
| --- |
| log调试 |
| ide调试 |
| 监测工具 |
| sdk调试支持api |

# 反射 章节

章节

参考资料

atititprgrm tech tree编程语言知识点体系 v7 qb20.xlsx