Atitit.编程语言新特性 通过类库框架模式增强 提升草案 v3 q27

[1. 修改历史 2](#_Toc8906)

[2. 适用语言：：几乎所有编程语言。语言提升的三个渠道：：语法，类库，框架，ide 2](#_Toc1445)

[2.1. 单根继承 vs 多跟继承 3](#_Toc21381)

[2.2. 默认参数 3](#_Toc29443)

[2.3. 等号判断相等，儿不是equ 3](#_Toc29658)

[2.4. 隐式类型，类型推导 3](#_Toc9037)

[2.5. 匿名类型 3](#_Toc20290)

[2.6. 初始化器 对象初始化器 与 集合初始化器 { } 4](#_Toc27627)

[2.7. 委托 4](#_Toc1542)

[2.8. 内置委托 Func / Action 4](#_Toc12897)

[2.9. 标准查询运算符 Standard Query Operator 4](#_Toc7287)

[2.10. 范型委托 4](#_Toc13106)

[2.11. 匿名方法 4](#_Toc18614)

[2.12. Lambda 4](#_Toc12541)

[2.13. 扩展方法 4](#_Toc7138)

[2.14. 面向过程与面向对象结合 4](#_Toc21875)

[2.15. Linq 5](#_Toc18866)

[2.16. 操作符重载 5](#_Toc6934)

[2.17. Preprocess源码预处理，宏 5](#_Toc10677)

[2.18. Pointer指针 5](#_Toc3558)

[2.19. 智能指针 5](#_Toc10201)

[2.20. 强类型 与弱类型同时支持 5](#_Toc25181)

[2.21. 手动转转struct指针支持 5](#_Toc17896)

[2.22. RAII(资源获取即是初始化 5](#_Toc7534)

[2.23. 泛型 5](#_Toc7701)

[2.24. 语法向前兼容 6](#_Toc7526)

[2.25. 支持typedef. 6](#_Toc5843)

[2.26. 支持单行和多行的注释之外，JAVA增加了第三种注释方法：文档注释。文档注释以结尾。 6](#_Toc4826)

[2.27. Reflection支持反射 6](#_Toc14990)

[2.28. 可选的自动边界检查 6](#_Toc12810)

[2.29. 支持指针,引用,值传递（无论对象还是基本数值） 7](#_Toc31228)

[2.30. 函数/方法的参数设置缺省值, . 但是方法重载(en:method overloading)可以达到同样的效果. 7](#_Toc26724)

[2.31. 最小的编译单位是一个函数 7](#_Toc5662)

[2.32. 允许基本类型之间的一些隐式的转换, 也允许程序员对于用户自定义类型相关的隐式转换规则 7](#_Toc32522)

[2.33. 支持编译器检查与运行时检查。。可以显示警告代替except 7](#_Toc5934)

[2.34. 异步api 代替大部分多线程api 7](#_Toc6782)

[2.35. 方法缺省情况下是虚的 7](#_Toc8198)

[2.36. 内存Gc 但是可以手动触发。。 支持析构函数. 7](#_Toc18676)

[2.37. 网络资源，io资源，线程，窗口等其他的资源的gc 8](#_Toc10699)

[2.38. 显式的覆盖 8](#_Toc7119)

[2.39. 阻塞和非阻塞的I/O 8](#_Toc20959)

[2.40. 大的标准库，但是模块化 8](#_Toc28235)

[2.41. 多重继承 8](#_Toc29794)

[2.42. Lambda函数指针支持 8](#_Toc19064)

[2.43. 代码内嵌文档机制 8](#_Toc24531)

[2.44. Anno，attr标记机制 8](#_Toc9954)

[2.45. 支持 goto 语句 8](#_Toc4740)

[2.46. 源码可以平台相关，也可以不相关，vm与native捷克 8](#_Toc19099)

[2.47. C++ 可以申请任意的内存块. 8](#_Toc14604)

[2.48. 可移植性 9](#_Toc21129)

[2.49. 显式锁操作(java.util.concurent.locks)与影视的synchronized  9](#_Toc7380)

[2.50. 常用的C++预处理编辑 9](#_Toc27166)

[2.51. 参考 10](#_Toc27757)

# 修改历史

V1 增加了c++中的语法特性

V2 增加了c#中的语法特性 v3

V3 增加js 中的语法特性

V4 增加 oc swift的语法特性

V5 增加lisp 语法特性

V6 增加php python语法特性

V7 增加ruby语法特性

# 适用语言：：几乎所有编程语言。语言提升的三个渠道：：语法，类库，框架，ide

不过主要特性来自于常用流行语言c++ java c# php js python ruby 。。另外参考参考 go rust list语言的好特性移植，过来

语法特性通过调整语法是最终的的，在语法未来得及提升的情况下，通过类库框架提升也是一个很好的方案

作者:: 绰号:老哇的爪子 （ 全名：：Attilax Akbar Al Rapanui 阿提拉克斯 阿克巴 阿尔 拉帕努伊 ） 汉字名：艾龙，  EMAIL:1466519819@qq.com

转载请注明来源： http://blog.csdn.net/attilax

## · **区分栈变量和堆变量**

。Java就是这么做的，现代编译器技术，可以做到不损耗性能，又降低程序员负担。

## · **没有分号**。

分号散布在代码里，充满了小麻点，太干扰视线了。现代编译技术完全可以做到不不需要分号，例如Scala, Go就是这么做的。

## 通过括号而不是缩减格式化

## · 支持**单机并行**的标准库或语言特性。 现在谁还用线程这种抽象工具呢

现在大部分机器都是多核了。不要用老旧的线程和进程模型，这种模型编程复杂，很容易死锁。Java 7里的Fork/Join 也不太好用。C++11加进了一个thread库，太落后了，现在谁还用线程这种抽象工具呢？像Scala里就有Actor，Erlang有轻量级process，Go语言有gorutine和channel，这几个接近我心目中的样子

## · 支持**分布式**的标准库。

现在的服务端编程，多机集群是家常便饭，标准库应该有一个方便易用的工具，方便程序员编写分布式程序。例如Erlang OTP, Scala有AKKA

## **类型放在变量名后**。我个人也认为这样可读性要好一些

。Go语言是这么做的，[为什么 Go 语言把类型放在后面？](http://www.zhihu.com/question/21656696)，Scala也是，ActionScript也是。

## 单根继承 vs 多跟继承

貌似目前单身继承较多。Cplus是卫衣的多继承实现。。应该允许多继承比较方便

## 默认参数

## 等号判断相等，儿不是equ

Java是少数不能用双等号判断str的lang

## ****隐式类型，类型推导****

**Var O=1**

## ****匿名类型****

　　　(1)源起

　　　　　　创建一个对象，一定要先定义这个对象的类型吗？

　　　　　　不一定的！

　　　　　　来看看这段代码

　　　　（2）使用

var obj = new {Guid.Empty, myTitle = "匿名类型", myOtherParam = new int[] { 1, 2, 3, 4 } };

如果你监视变量obj，你会发现，obj的类型是Anonymous Type类型的

这个特性在网站开发中，序列化和反序列化JSON对象时很有用

## ****初始化器**** 对象初始化器 与 集合初始化器 { }

　　var myObj1 = new MyObj ("allen") { id = Guid.NewGuid(), Title = "allen" };

## 委托

## 内置委托 Func / Action

## 标准查询运算符 Standard Query Operator

## ****范型委托****

## ****匿名方法****

## ****Lambda****

## ****扩展方法****

如果想给一个类型增加行为，一定要通过继承的方式实现吗？

4>第一个参数必须附加this关键字作为前缀

扩展方法的本质是在运行时调用扩展类的静态方法

## 面向过程与面向对象结合

世界并不是全部是对象模式的

## Linq

## 操作符重载

## Preprocess源码预处理，宏

* C++ 编译多了一个文本[预处理](http://blog.csdn.net/wiki/%E9%A2%84%E5%A4%84%E7%90%86%E5%99%A8" \o "预处理器)过程, Java 是没有的. 因此一些使用者在他们的编译过程之前增加了一个预处理的过程,这样能更好的支持需要条件编译的情况.

预处理器为开发人员提供了方便，但增加丁编译的复杂性。JAVA虚拟机没有预处理器，但它提供的引入语句(import)与c十十预处理器的功能类似。

## Pointer指针

完全取消指针是错误的，可以通过类库增加指针功能。。

## 智能指针

## 强类型 与弱类型同时支持

强类型检查和四种转型

## 手动转转struct指针支持

## [RAII](http://blog.csdn.net/w/index.php?title=RAII&action=edit&redlink=1" \o "RAII)(资源获取即是初始化

Java 主要依赖垃圾搜集,which can only reclaim memory and may be a last shot at other resources(不知道咋翻译...), 而 C++ 主要依赖 [RAII (资源的获取就是初始化](http://blog.csdn.net/w/index.php?title=RAII&action=edit&redlink=1" \o "RAII)

## 泛型

其主要目的是提供类型安全的容器

## 语法向前兼容

语言向前兼容，尽可能的兼容流行语言的语法

## 支持typedef.

## 支持单行和多行的注释之外，JAVA增加了第三种注释方法：文档注释。文档注释以结尾。

## Reflection支持反射

## 可选的自动[边界检查](http://blog.csdn.net/w/index.php?title=%E8%BE%B9%E7%95%8C%E6%A3%80%E6%9F%A5&action=edit&redlink=1" \o "边界检查)

简单的说，就算检查你的索引是否在0~(len-1)之内

貌似是由字节码指令附带的功能，由虚拟机执行时进行检查。约见except

* 因为表达方式不受限制,低级的 C++ 语言特性(例如:不被检查的数组访问,原始指针,[类型双关语](http://blog.csdn.net/w/index.php?title=%E7%B1%BB%E5%9E%8B%E5%8F%8C%E5%85%B3%E8%AF%AD&action=edit&redlink=1" \o "类型双关语)([en:type punning](http://en.wikipedia.org/wiki/type_punning" \o "en:type punning)))不能在编译期间或者运行期间可靠地被检查. 相关的编程错误会导致低级的[缓存溢出](http://blog.csdn.net/wiki/%E7%BC%93%E5%AD%98%E6%BA%A2%E5%87%BA" \o "缓存溢出)和[段错误](http://blog.csdn.net/w/index.php?title=%E6%AE%B5%E9%94%99%E8%AF%AF&action=edit&redlink=1" \o "段错误)([en:segmentation fault](http://en.wikipedia.org/wiki/segmentation_fault" \o "en:segmentation fault)). [标准模板库](http://blog.csdn.net/wiki/%E6%A0%87%E5%87%86%E6%A8%A1%E6%9D%BF%E5%BA%93" \o "标准模板库) 提供了高级的抽象(例如 vetor,list 和 map)来帮助避免这样的错误. 在 Java 里, 低级错误不会发生或者会被[JVM](http://blog.csdn.net/wiki/JVM" \o "JVM)检测到并以[异常](http://blog.csdn.net/w/index.php?title=%E5%BC%82%E5%B8%B8&action=edit&redlink=1" \o "异常)的形式报告给应用.

## 支持指针,引用,值传递（无论对象还是基本数值）

## 函数/方法的参数设置缺省值, . 但是[方法重载](http://blog.csdn.net/w/index.php?title=%E6%96%B9%E6%B3%95%E9%87%8D%E8%BD%BD&action=edit&redlink=1" \o "方法重载)([en:method overloading](http://en.wikipedia.org/wiki/method_overloading" \o "en:method overloading))可以达到同样的效果.

## 最小的编译单位是一个函数

## 允许基本类型之间的一些隐式的转换, 也允许程序员对于用户自定义类型相关的隐式转换规则

## 支持编译器检查与运行时检查。。可以显示警告代替except

## 异步api 代替大部分多线程api

## 方法缺省情况下是虚的

这样方便覆盖

C++ 方法缺省情况下不是虚的. 在 Java 里, 方法缺省情况下是虚的, 但可以使用final关键字使之声明为非虚的.

## 内存Gc 但是可以手动触发。。 支持析构函数.

强制使用自动垃圾搜集导致了在 Java 里编写[实时软件](http://blog.csdn.net/w/index.php?title=%E5%AE%9E%E6%97%B6%E8%AE%A1%E7%AE%97&action=edit&redlink=1" \o "实时计算)([en:Real-time computing](http://en.wikipedia.org/wiki/Real-time_computing" \o "en:Real-time computing))是困难的.[[3]](http://blog.csdn.net/lile269/article/details/6424414" \l "cite_note-Martin-2)

## 网络资源，io资源，线程，窗口等其他的资源的gc

## 显式的覆盖

## 阻塞和非阻塞的I/O

## 大的标准库，但是模块化

## 多重继承

## Lambda函数指针支持

## 代码内嵌文档机制

## Anno，attr标记机制

## 支持 goto 语句

## 源码可以平台相关，也可以不相关，vm与native捷克

## C++ 可以申请任意的内存块.

* Java 只能通过对象实例化来申请内存. (注意:在 Java 里, 程序员可以通过创建一个字节数组模拟申请任意的内存块. 不过 Java [数组](http://blog.csdn.net/wiki/%E6%95%B0%E7%BB%84" \o "数组)仍然是对象.)

## 可移植性

## 显式锁操作(java.util.concurent.locks)与影视的**synchronized**

显式锁，锁可以沿着代码块的方向释放，其中包括异常被抛出，从而消除错误的常见来源。

## 常用的C++预处理[编辑](http://baike.baidu.com/javascript:;)

1)常用的预处理：

#include 包含头文件

#if 条件

#else 否则

[#elif](http://baike.baidu.com/subview/8443374/8377174.htm" \t "http://baike.baidu.com/_blank) 否则如果

[#endif](http://baike.baidu.com/subview/1617237/1617237.htm" \t "http://baike.baidu.com/_blank) 结束条件

[#ifdef](http://baike.baidu.com/subview/1617233/1617233.htm" \t "http://baike.baidu.com/_blank) 或 #if defined 如果定义了一个符号, 就执行操作

[#ifndef](http://baike.baidu.com/subview/1617265/1617265.htm" \t "http://baike.baidu.com/_blank) 或 #if !defined 如果没有定义一个符号，就指执行操作

#define 定义一个符号

[#undef](http://baike.baidu.com/subview/1998342/1998342.htm" \t "http://baike.baidu.com/_blank) 删除一个符号

#line 重新定义当前行号和文件名

[#error](http://baike.baidu.com/subview/2231819/2231819.htm" \t "http://baike.baidu.com/_blank) 输出编译错误 消息,　停止编译

[#pragma](http://baike.baidu.com/subview/1451188/1451188.htm" \t "http://baike.baidu.com/_blank) 提供 机器专用的特性,同时保证与C++的完全兼容

## 参考

C++和java技术特性对比 - lile269的专栏 - 博客频道 - CSDN.NET.html

线程基础之JAVA和C++0x的特性 \_ 并发编程网 - ifeve.com.html

(2 封私信 / 5 条消息) C++语言与Java语言的区别有哪些？ - 知乎