NatsuLang规范

1. 范围

这个文档说明了对NatsuLang的实现的需求，并定义了NatsuLang的概念及规范。

1. 术语与定义

仅在较小区域使用的术语将在第一次使用时就地定义。

* 1. 访问

（执行期行为）读取或修改一个对象的值。

* 1. 参数

（函数调用表达式）括号中由逗号分隔的列表中的表达式。

* 1. 参数

（抛出表达式）throw的操作数。

* 1. 阻塞

线程执行时等待某些状态被满足后再继续执行阻塞操作之后的操作。

* 1. 选择性支持

实现不需要支持的功能。

* 1. 诊断信息

一些实现输出的信息，内容由实现定义。

* 1. 病式的程序

并非合式的程序（2.13）。

* 1. 实现定义行为

指合式的程序的决定于实现的行为，这个行为应当在实现的文档中说明。

* 1. 实现限制

由实现施加给程序的限制。

* 1. 签名

（函数）名称、参数类型列表、以及包含它的模块。

[注释：签名作为连接等的基础——注释结束]

* 1. 未定义行为

标准未施加任何要求的行为。

* 1. 未说明行为

指合式的程序的决定于实现的行为，实现不必说明这个行为。

* 1. 合式的程序

遵守语法规则、可诊断的语义规则的NatsuLang程序。

1. 一般原则
   1. 实现承诺
      1. 可诊断规则包括本规范所有的句法与语义规则，除了包括显式标记为“不需要诊断”或者被描述为造成“未定义行为”的规则。
      2. 由标准库定义的名称位于std模块中，翻译单元通过引用这个模块以访问这些名称。
      3. 一个遵守规则的实现可能具有一些提供了不修改任何合式的程序的行为的扩展。实现应当诊断使用了这样的扩展的程序是否根据本规范是非合式的。之后可以编译和运行这样的程序。
      4. 所有的实现应当提供包含了所有不支持的可选支持功能的文档。
   2. 语法标记
      1. 在本文档所使用的标记中，语法类别将使用*斜体*指示，字面的词语及字符将使用定宽字体。可选项列在独立的行中，或者在由“以下其一”表示的下一行之中列出，如果可选项的文字过长并超出一行，则文字将会在下一行继续，并且开头多出一个缩进。可选的终结或非终结符号将会由“可选”下标指示。
      2. 语法分类的名称一般将会由以下的规则命名：
         1. ——*X名称* 是一个标识符，在当前上下文中决定它的含义（例：*类型名称*）。
         2. ——*X标识符* 是一个具有不依赖上下文的含义的标识符（例：*限定标识符*）。
         3. ——*X序列* 表示一个或多个*X*，且无中间分隔符（例：*声明序列* 表示一个声明的序列）。
         4. ——*X列表* 表示一个或多个*X*，且由中间分隔符分开（例：*标识符列表* 表示一个由逗号分隔开的标识符的序列）。
2. 词法约定
   1. 分离编译
   2. 编译阶段
   3. 字符集
   4. 记号

记号：

标识符

关键字

字面量

操作符

符号

* + 1. 记号分为5种：标识符、关键字、字面量、操作符以及符号。
  1. 注释
  2. 标识符
  3. 关键字
  4. 操作符和符号
  5. 字面量
     1. 字面量的类型
     2. 整数字面量
     3. 字符字面量
     4. 浮点字面量
     5. 字符串字面量
     6. 逻辑字面量

1. 基本概念
   1. 条款说明
   2. 声明和定义
   3. 范围
      1. 定义域和范围
      2. 声明的时机
      3. 块范围
      4. 函数原型范围
      5. 函数范围
      6. 模块范围
      7. 类范围
      8. 枚举范围
      9. 名称隐藏
   4. 名称查找
      1. 非限定名称查找
      2. 限定名称查找
         1. 类成员
         2. 模块成员
         3. 类成员访问
   5. 程序和链接
   6. 启动和结束
      1. 入口函数
      2. 结束
      3. 存储期
      4. 子对象的存储期
      5. 对象生存期
   7. 类型
      1. 初等类型
      2. 组合类型
   8. 对齐
2. 标准转换
3. 表达式
   1. 条款说明
      1. [注释：条款6定义了表达式的语法、求值顺序以及含义。表达式是由操作符和操作数组成的，表示了一个计算的序列。表达式可以得到一个值作为结果，还可以造成副作用。——注释结束]
      2. 条款6定义了未被当前应用的类型重载的操作符的效果。重载的操作符不能修改*内建操作符*的规则，即应用在由标准定义的类型的操作符。这些内建操作符参与重载决议。
      3. 若在表达式的求值中，其结果没有在数学上定义或者不在这个类型的可表示范围内，则行为是未定义的。
   2. 初等表达式

初等表达式：

字面量

this

(表达式)

标识符表达式

* + 1. 字面量
       1. 字面量是一个初等表达式，它的类型取决于它的形式。
    2. This
       1. 关键字this在非静态成员函数被执行或非静态数据成员的初始化器被求值时表示一个对对象的引用。
       2. 表达式this不能在其他上下文出现。
    3. 括号
       1. 被括号包围的表达式(E)是一个初等表达式，它的类型与值都和E相同。被括号包围的表达式可以在和E可以被使用的上下文被使用，并且具有相同的意义，除非另有说明。
    4. 名称

标识符表达式：

未限定标识符

限定标识符

* + - 1. 概述
         1. 标识符表达式是初等表达式的限制表示。[注释：标识符表达式可以在.操作符后出现——注释结束]
         2. 标识符表达式用于表示一个类的非静态数据成员或者非静态成员函数时只能用在：

——作为类成员访问的一部分，当对象表达式引用成员的类[[1]](#footnote-1)或者由此派生的类，或者

——构成一个指向成员的引用，或者

——如果标识符表达式表示一个非静态数据成员并且它出现在不求值操作数中。

[示例:——示例结束]

* + - 1. 未限定名称

未限定标识符:

标识符

* + - * 1. *标识符*是一个被合适地声明的标识符表达式，表达式的类型是标识符的类型。结果是一个由标识符表示的实体。
      1. 限定名称

限定标识符：

嵌套名称说明符 未限定标识符

嵌套名称说明符：

类型名称.

模块名称.

typeof说明符.

嵌套名称说明符 标识符.

* + - * 1. 在嵌套名称说明符中由typeof说明符表示的类型应当是类或者枚举类型。
        2. 若嵌套名称说明符表示了一个类，其后是该类或该类的基类的成员的名称，则如此表示的是一个限定标识符。结果是表示的成员，结果的类型是该成员的类型。
        3. 若嵌套名称说明符表示了一个枚举，其后是这个枚举的枚举值的名称，则如此表示的是一个引用了该枚举值的限定标识符。结果是该枚举值，结果的类型是这个枚举的类型。
    1. Lambda表达式
  1. 后缀表达式
     1. 概述
        1. 后缀表达式由左向右结合。

后缀表达式：

初等表达式

后缀表达式 [ 表达式或大括号初始化列表 ]

后缀表达式 ( 表达式列表可选 )

简单类型说明符 ( 表达式列表可选 )

类型名说明符 ( 表达式列表可选 )

后缀表达式 . 标识符表达式

后缀表达式 ++

后缀表达式 --

表达式 as 类型标识符

表达式列表：

初始化器列表

* + 1. 下标
    2. 函数调用
    3. 类成员访问
    4. 自增与自减
    5. 类型转换
  1. 一元表达式
     1. 概述
        1. 具有一元操作符的表达式由右向左结合。

一元表达式：

后缀表达式

++ 类型转换表达式

-- 类型转换表达式

一元操作符 转换表达式

new表达式

delete 表达式

一元操作符：以下其一：

+ - ! ~

* + 1. 一元操作符
       1. 一元+操作符的操作数应当具有数学类型，结果是操作数的值，结果的类型是操作数的类型。
       2. 一元-操作符的操作数应当具有数学类型，结果是操作数取负的值，对于位数为n的无符号数的负值，结果是2n减去该操作数的值，结果的类型为操作数的类型。
       3. 逻辑取反操作符!的操作数应当具有bool类型，若操作数的值为false，则结果为true，否则结果为false，结果的类型为bool。
       4. 按位取反操作符~的操作数应当具有整数类型，结果是操作数的按位取反值，结果的类型为操作数的类型。
    2. 自增与自减

1. 语句
2. 声明
3. 声明符
4. 类
5. 重载
6. 异常处理

1. 包括隐式的this。 [↑](#footnote-ref-1)