# SEAL++语法手册



## 简介

Seal++（Simple Education Accompanying Language Plus Plus）是一款用于编译原理教学的编程语言。是一种静态类型语言。

## 基本语法

### 2.1 基本数据类型

Seal的基本数据类型包括了**Int**、**Float**、**String、Bool**和**Void**。其中，**Int**默认为64位整数（同C++里的long long），**Float**为64位浮点数（同double），**String**为字符串，**Bool**为布尔变量；**Void**为空类型，用于函数标识无返回值，以及无类型表达式的值，例如空表达式。其中，**Int**类型可以写作十进制形式，15，。**Bool**有两个值，即**0**和**1**。浮点数**Float**即按照正常的十进制表示法，即如1.214，不允许0.和.5的形式，所有的浮点数必须以x.x的形式，例如8的浮点形式为8.0，如果写作8则视为整型。

除了五个基本数据类型的名称是以大写字母开头，其余所有变量、函数命名一定要以小写英文字母做开头，名字的其他部分可以用到大小写字母、数字以及下划线。

### 2.2 变量声明

Seal可以在函数体外、函数体内声明，基本格式如

Float x;

Float y;

赋值

x := 1.0;

其中，**Float**为类型，放在开头。要求，每一行只能声明一个变量。但是请注意，在函数声明的范围之外仅仅可以声明变量，不可以执行赋值等任何语句。

### 2.3 函数调用和定义

Seal++中的函数声明如下格式

Float func max(Float x, Float y)

{

if x > y

{

return x;

}

return y;

}

可以看到函数的形参的声明部分，第一个**Float**表示返回类型，小括号内类型**Float**在前， 变量名**x**在后。关键词**func**用于指示函数声明，返回值最多只能有一个。返回关键词为**return**，后面跟要返回的值，当返回值为空时，该处填为**void**。在调用时，可以直接如max(1.1,1.2)来调用。必须有一个Void func main()函数，即该函数参数为空，且返回值为空，作为整个程序的入口。

此外，为了在目标代码生成阶段的方便，限制每个函数最多有6个参数。

### 2.4 条件语句

基本条件语法类似C++，如

if x>10

{

x := x + 1;

}

或

if x==0

{

x := 3;

}

else

{

x := 2;

}

注意，在Seal++中，**if**的条件判断部分不用带小括号，所以这里直接就是x>10。第一个条件用**if**开头，使用**else**作为其他条件，如果只有一个判断条件，只用**if**即可。

### 2.5 循环语句

Seal++的循环语法有**while**和**aafor**两种，**while**形如

while <condition> Body

condition为while的条件部分，不可缺失，必须由bool类型的表达式构成。Body是一个表达式块，即用大括号括起来的一系列语句。具体执行时，按照<condition> Body <condition> Body的顺序。形式上的例子如

while x < 10

{

x := x+1;

y := 3 + 2;

}

**aafor**形如:

aafor <initexpr>;<condition>;<loopact>

{Stmts;}

这里的initexpr为循环执行前执行的表达式，condition为执行循环体之前的条件判断语句，loopact为每轮循环体结束的执行动作。例如

aafor x := 0; x < 10; x := x + 1

{

s := s + x;

p := x + 3;

}

最开始先会执行x=0，而后判断x<10是否成立，成立则进入循环体，循环体执行完毕之后，执行loopact语句x:=x+1，而后再判断x<10是否成立，以此类推。

同样的，**while**和**aafor**的条件部分都是不用加括号的。**aafor**语句中，由分号隔开的initexpr、condition、loopact三个部分，可以缺失其中的任何或者全部，例如

aafor ;x<10;

{

x := x+1;

}

对于两个循环语句类型，存在关键字**continue**和**break**，**break**能够直接中断循环体，执行后续的部分；**continue**则跳出当前循环体，**while**语句直接到<condition>部分，**for**语句直接到loopact部分。

有一个特殊的函数，fprintf，与C++中的类似，其参数第一个是输出字符串的设备，其参数第二个为String类型，后面可以跟若干个其他类型的参数，用于输出一个字符串。而对应参数在第一个参数字符串中必须有占位符，对应的占位符如下表

|  |  |
| --- | --- |
| Int | %lld |
| Float | %f |
| Bool | %lld |
| String | %s |

例如可以调用写作fprintf(stdout, "%f, %lld", 1.1, b);

或者可以写作fprintf(stderr, "%f, %lld", 1.1, b);

这个函数不可以重新声明，在语义分析部分，要检查是否重定义、第一个参数是否为stdout或stderr、第二个参数类型是否为String；目标代码生成部分，相关的部分已经给出。

### 2.6 表达式

Seal++的运算符，将其大致上分为算术运算符和逻辑运算符。在实际操作中，我们将每一个表达式会标识为某个类型，例如Int类型的a和b，式子a+b的结果类型为Int，而(a+1)>b的类型为Bool。对于循环和条件语句的条件部分，必须由Bool类型的表达式构成。

|  |  |  |
| --- | --- | --- |
|  | 算数运算符 | 逻辑运算符 |
| Int | + - \* / %（余）-(单目，取负) &（按位与） |（按位或） ^（按位异或） ~（单目、按位取反）:= | > >= == != <= < |
| Bool | := | && || ! != == ^（异或） |
| Float | + - \* / -(单目，取负) := | > >= == != <= < |

原则上，只有相同类型的数据类型能够进行算术运算，得到的结果是相同类型的表达式。但是，对于数值**比较**，可以在Int与Float之间进行，+ - \* /也可以在Int与Float之间执行，而得到的表达式类型为Float。

特别要注意的是，String类型并没有支持任何的运算符，意即我们不会对String类型的值做任何修改操作。

我们考虑运算符的优先级，规定：

|  |  |
| --- | --- |
| 结合律 | 符号 |
| 左 | ~ & | ^ |
| 无 | -（单目取负）! |
| 左 | \* / % |
| 左 | + -（双目减法） |
| 无 | < <= >= > |
| 无 | == != |
| 右 | && |
| 右 | || |
| 无 | := |

优先级顺序是最上面优先级最高，往下优先级依次降低。当然，小括号内的是最优先的。

### 2.7 语句块

特别需要强调一下语句块，语句块即后文语法形式定义中的，也即利用大括号括起来的语句们。典型的如**while**语句的Body部分即是一个语句块。语句块内如果声明局部变量，则其所有的局部变量声明语句必须位于语句块中的开头部分，然后是正常的语句。具体的语句块如何构成，可以参考语法形式部分。

### 2.8 注释

Seal++语言的注释为单行注释。单行注释以#起始，一直持续到行末换行符结束。

## 语法形式定义

Seal++所包含的语法形式定义如下（以下出现的“：=”表示“定义为”，与赋值运算符无关）

其中，竖线|表示或者，[]内的内容为可选择，双方括号同样为可选择，后面跟\*号表示可以重复0个或者多个，需要特别强调说明的是，基本的数据类型都是大写开头的。