**Init函数**

init函数代表operator初始化运算过程，**为块语句，可以包含任意程序员自定义的变量声明和语句内容**，并仅在operator第一次执行时执行。

**work函数**

work函数代表operator的最细粒度的运算部分，**为块语句，可以包含任意程序员自定义的变量声明和语句内容**，是operator的核心结构和数据流程序稳态执行过程中每次计算的执行单元。work函数内含有对operator输入流、输出流数据的操作。

**Window**

window结构规定了work函数每次执行时所需的输入流和输出流窗口类型以及大小。窗口类型有两种：sliding和tumbling。sliding代表滑动窗口类型，这种窗口类型带有2个参数，第一个指定滑动窗口长度，第二个指定每次滑动的距离，单位都是数据流中数据项的个数。例子， S sliding(10,1);表示Stream流S上定义了一个sliding窗口，长度为10个数据项长度，work函数每次执行后，向后滑动1个数据项长度。

tumbling则代表翻转窗口类型，这种窗口只带有1个参数，指定窗口长度。如T tumbling(10);表示stream流T上定义了一个tumbling窗口，窗口长度为10个数据项长度，work函数每次执行后，窗口中的10个数据被清空。对于输入数据流，表示10个数据被消耗，输入流中后续数据将进入窗口；对于输出数据流，表示10产生的数据已经填满窗口，将被全部输出到输出数据流中，后续产生的数据将继续填充窗口。

一般而言，sliding常用于输入数据流，而tumbling常用于输出数据流。

**work函数采用窗口触发机制执行：**即只有当数据填满window时，work函数才会执行，否者operator将会一直等待，直到有足够的数据。

**stream数据的访问**

**operator输入流和输出流数据只能在work函数中访问。**访问的方式采用C语言的数组下标访问方式，访问的下标按照输入流与输出流在window队列中的顺序采用FIFO编号。图1显示了window中流数据下标访问的顺序。



图1： window中流数据下标访问的顺序

对于流数据项的每一个分量，采用类似于C语言中结构体分量的访问方式，采用取分量的“.”操作来实现。如，stream<int x, int y> S; S[0].x表示取S的window中下标为0的数据项中的整型分量x。

以图2的source节点为例，其state函数内声明了一个整型变量x，并在init函数中初始化为0。work函数中每次执行都会对x执行加1操作，并将结果赋给输出流（S：S[0].i=x）。window定义了source的输出流S的窗口类型为tumbling，并决定了大小为1（S tumbling (1)）。所以，source作为由“0”开始的自然数序列数据源，每次产生一个递增为1的输出数据。



图2： operator的构成