Blockly编程案例

案例一 原码反码和补码

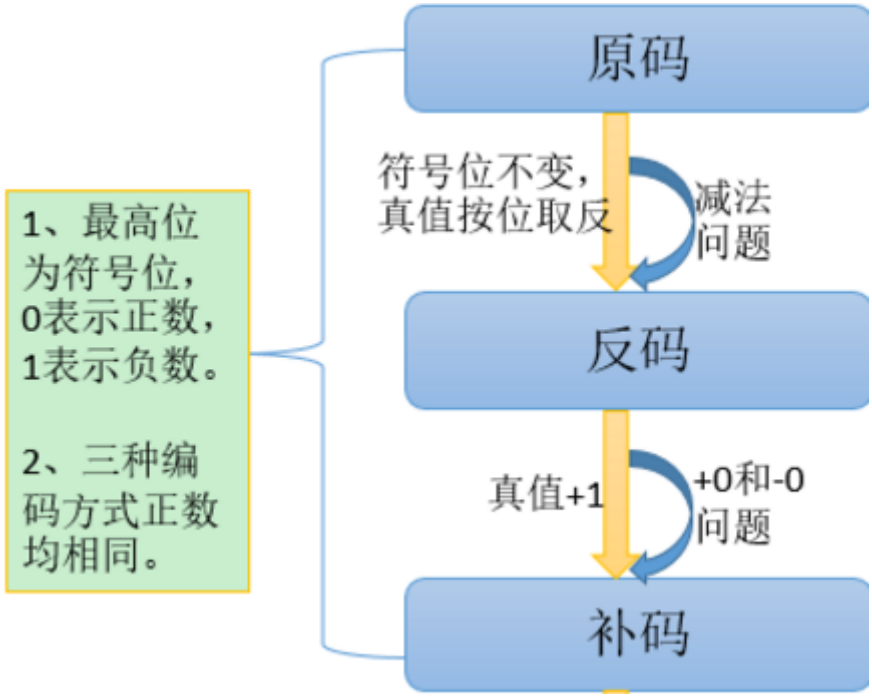
我们把一个数在计算机内被表示的二进制形式称为机器数，该数称为这个机器数的真值。机器数有固定的位数，具体是多少位与机器有关，通常是8位或16位。

原码:是指符号位用0或1表示，0表示正，1表示负，数值部分就是该整数的绝对值的二进制表示。例如：假设机器数的位数是8，那么：[+17]原=00010001 [-39]原=10100111

反码:在反码的表示中，正数的表示方法与原码相同；负数的反码是把其原码除符号位以外的各位取反（即0变1，1变0）。通常，用[X]反表示X的反码。例如： [+45]反 = [+45]原 = 00101101，[-32]原 = 10100000，[-32]反 = 11011111

补码:在补码的表示中，正数的表示方法与原码相同；负数的补码在其反码的最低有效位上加1。通常用[X]补表示X的补码。例如： [+14]补 = 10100100,[-36]反 = 11011011,[-36]补 = 11011100

实事求是地说，引入补码意义非同寻常，可以说是先辈们智慧的结晶。因为，通过补码运算，可以把减法运算变成加法运算；而乘法可以用加法来做，除法可以转变成减法。这样一来，加、减、乘、除四种运算“九九归一”了。这对简化CPU的设计非常有意义，CPU里面只要有一个加法器就可以做算术运算了。



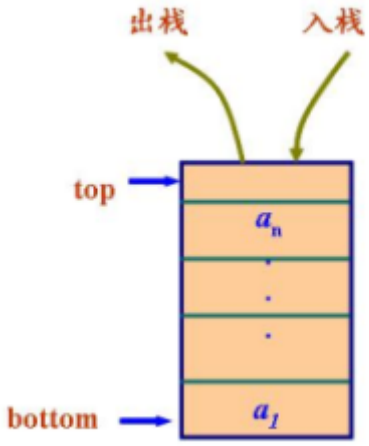
因此只要能将数字转化为二进制机器数，那么就可以模拟计算机完成各种数值运算，实现加减乘除的功能。下面我们来一起将正负数的十进制数转化为二进制：

需求分析：输入：一个十进制数，输出：这个数的原码，反码和补码。

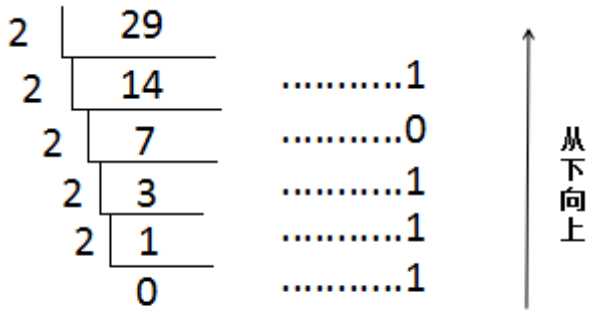
实现：

1、正数的原码反码和补码都是相同的，因此只要求出一个即可，我们通常用栈来实现向二进制的转换，blockly提供list作为指针数组可以实现栈的操作，如下图所示：

栈操作实际上就是一种先进后出的数据结构：

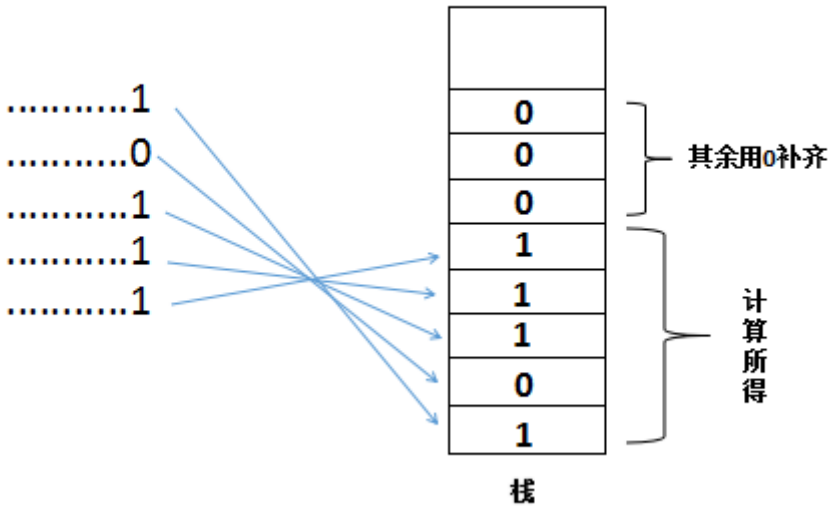


我们举一个例子，比如求29的原码，那么过程如下：

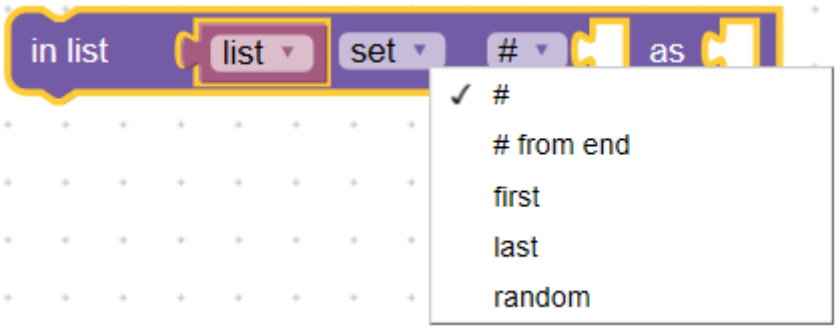


29的原码为00011101

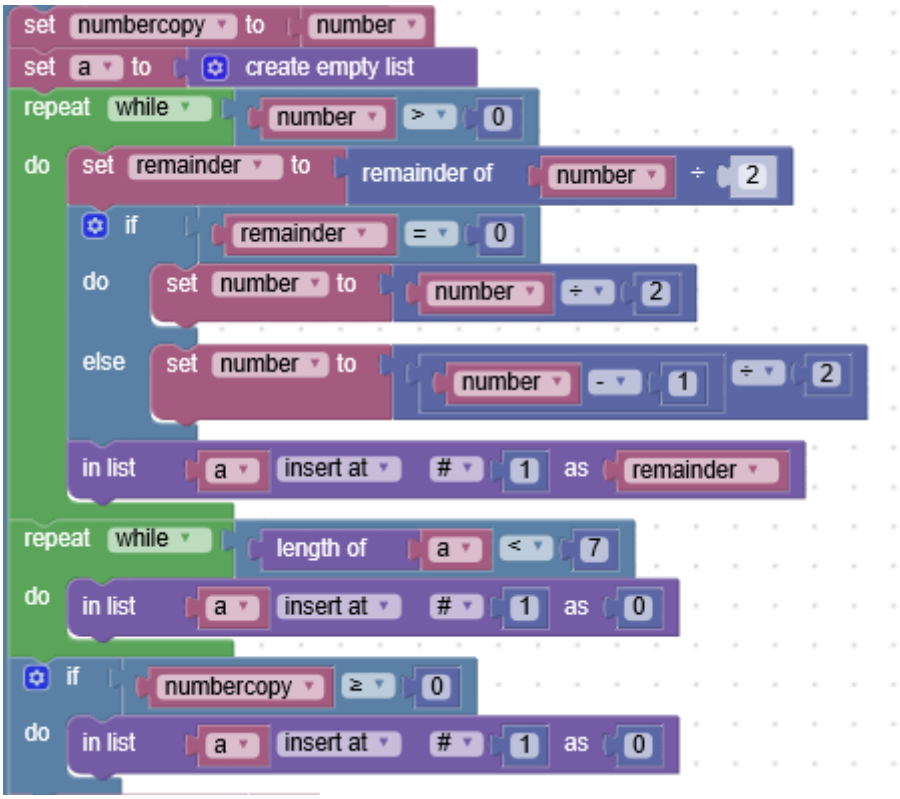
29的原码为00011101



list数组提供从头尾插入和删除元素，并能在指定位置提取元素：

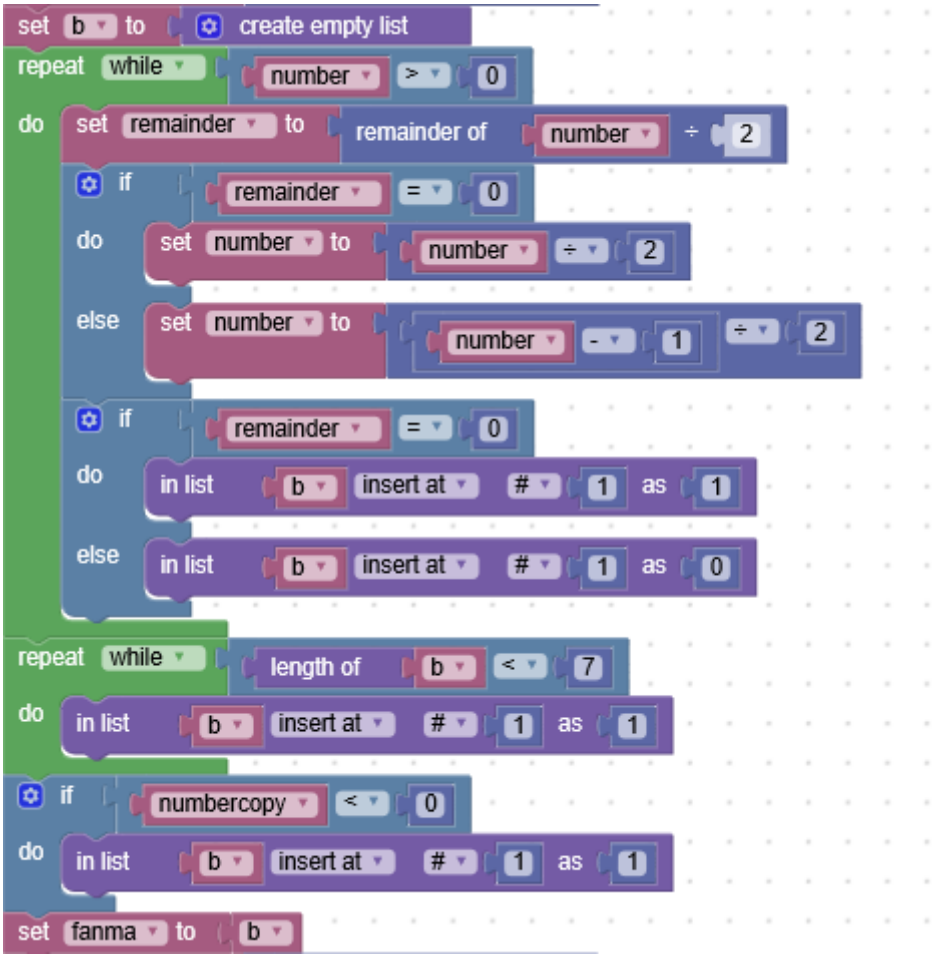


因此我们只要依次将求出的余数从数组头处插入即可，读取数组的时候从头到尾开始阅读。



2.负数的原码其实内容上和正数是一样的，只是符号位与正数是不同的，我们之所以把数设置成8位，就是为了留出1位作为符号位，因此8位数可以表示-128~127，第一位表示符号位，0为正数，1为负数，所以-29二进制表示为：10011101。

负数的反码就是符号位不变，其余位取反，反码的意义就在于进行下一步取补码。负数的补码就是反码加1，求出补码的算法就是从最后一位开始判断，如果是0，就将其变为1，如果是1，就判断倒数第二位，依次类推，你可以举几个例子自行判断一下。


set b to create empty list
repeat while number > 0
do set remainder to remainder of number ÷ 2
if remainder = 0
do set number to number ÷ 2
else set number to number - 1 ÷ 2
if remainder = 0
do in list b insert at # 1 as 1
else in list b insert at # 1 as 0
repeat while length of b < 7
do in list b insert at # 1 as 1
if numbercopy < 0
do in list b insert at # 1 as 1
set fanma to b

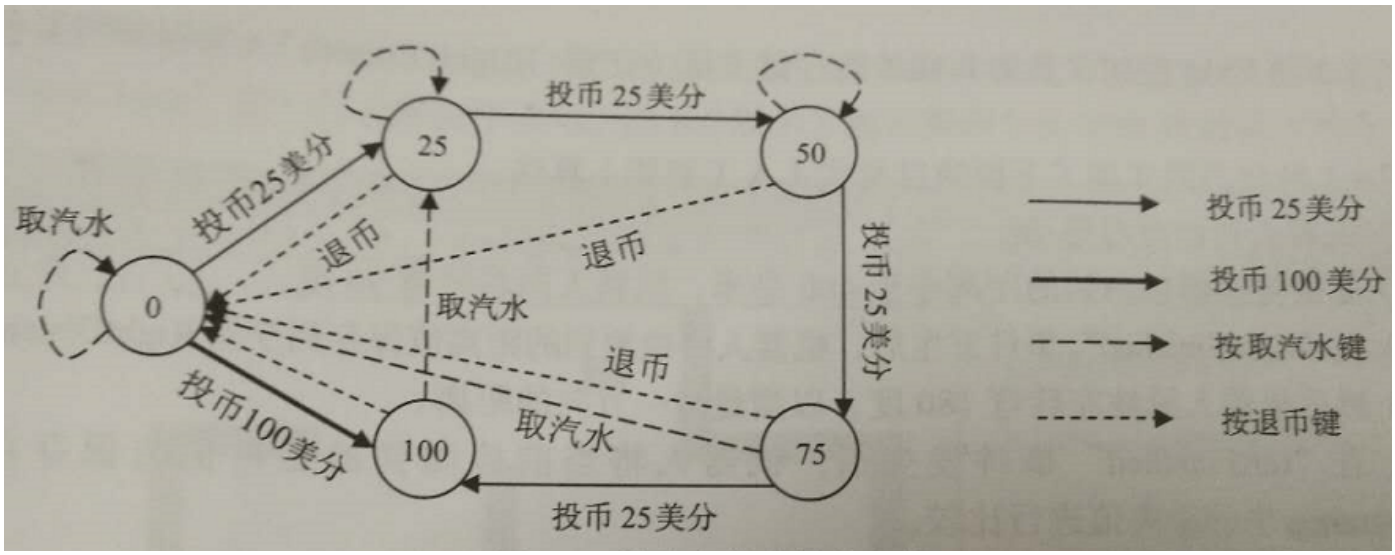
案例二 有限状态机

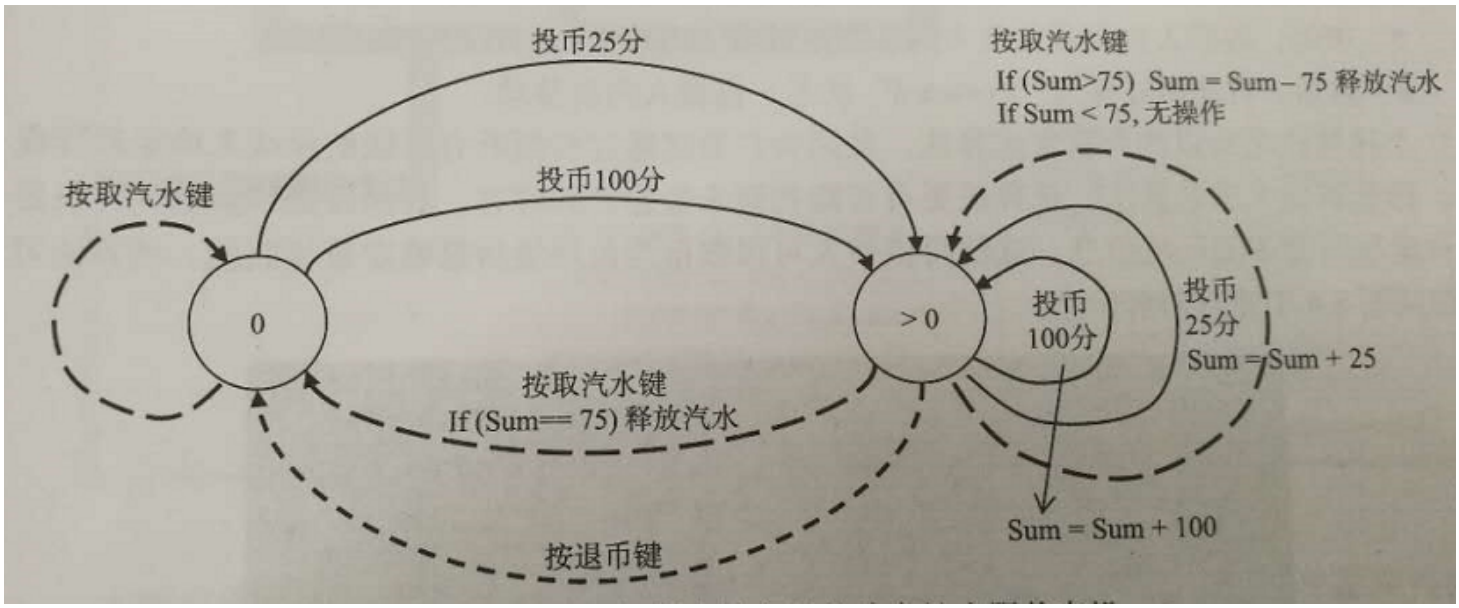
有限状态机，（英语：Finite-state machine，FSM），又称有限状态自动机，简称状态机，是表示有限个状态以及在这些状态之间的转移和动作等行为的数学模型。它反映从系统开始到现在时刻的输入变化，转移指示状态变更，并且用必须满足来确使转移发生的条件来描述它；动作是在给定时刻要

进行的活动的描述。　我们举一个例子，我们平时在自动售货机上买饮料，自动售货机的内部实际上就是一个有限状态机：

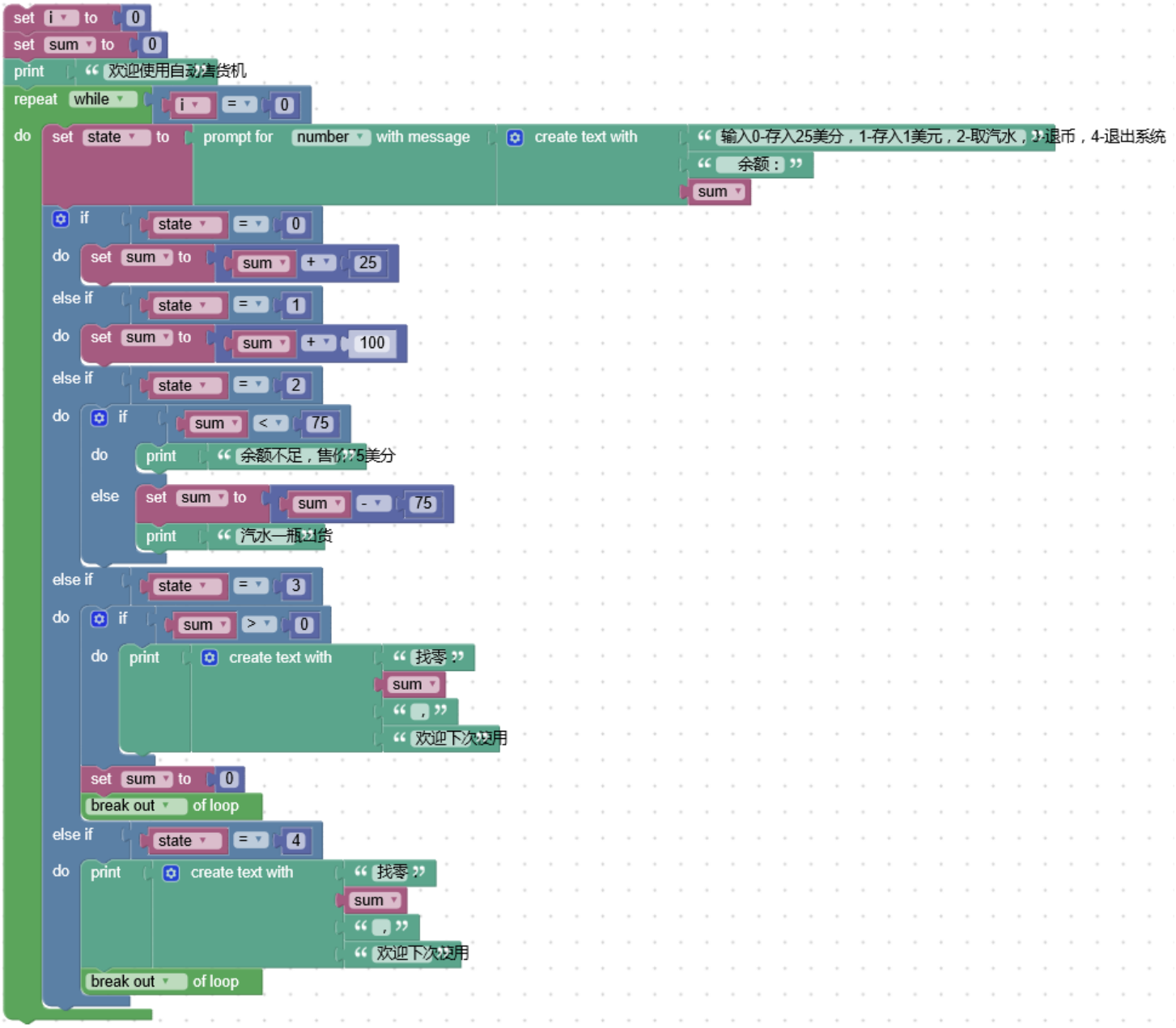
假如有这样一个自动售货机，售卖苏打汽水，它有如下五种输入：

- 输入0 -- 存入25美分
- 输入1 -- 存入1美元
- 输入2 -- 取汽水
- 输入3 -- 退币
- 输入4 -- 退出系统

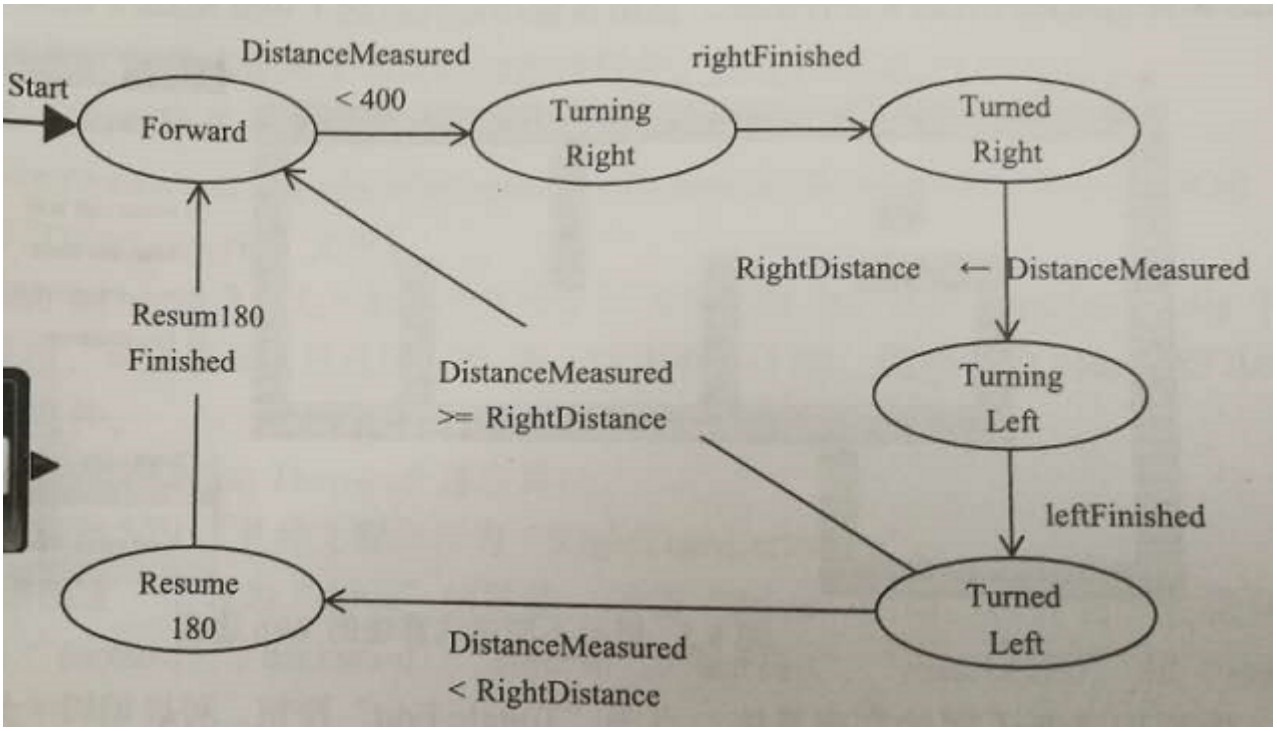




实现如下：



其实我们的小车就是通过有限状态机来实现的，我们可以把小车的行为依依列举出来，比如左右转，前进，发出声波，光线，声音等，我们只要通过控制这些状态就能控制小车完成各样的工作：



你能自己设计出一个更加复杂的售货机么?