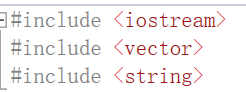
实验一

姓名：黄丹禹

学号：2012030

日期：2022/10/26

1. 实验平台
   1. 基于Visual Studio 2022，使用+语言编写
   2. 调用函数库：
2. 实验目的

单序列基本操作（并判断是否受序列长度影响，对于无限长序列应完车即时允许情况下应完成即时操作)

* 1. 满足前、后补零操作
  2. 满足序列延迟、提前操作
  3. 满足序列反转操作
  4. 满足序列拉伸、压缩操作（上采样、下采样)
  5. 满足序列差分、累加操作

1. 实验核心公式及问题
   1. 前后补零实现：

实验中，后补零直接向vector中利用push\_back()函数插入k个0；前补零通过vector的insert函数，向位置0插入k个0。

* 1. 若用数组完成补零操作:

方法一：为新建固定长度的数组，长度固定比如为20，后补零就是将已初始化的部分后面继续初始化k个0即可；前补零需要将数组中已经初始化的每个元素从后向前，每个元素向后移动k个位置，之后把移动出来的k个位置赋成0。当数组长度不够是，将数组长度扩大为原来的二倍后再插入新的。

方法二：通过begin和end值得到已有（end-begin+1）个数据，每次补零都新建一个数组将新插入的0和原数据复制到新数组中，之后delete原数组，原理和方法一相同，不同处在于方法一有空间冗余但更新数组次数（即double数组长度的次数）较少，时间复杂度较低，方法二每次都需要new一个新数组并进行大量数据移动、delete旧数组，时间复杂度很大。

* 1. 上采样后的序列长度：

设拉伸num倍，上采样后的序列的长度不单单是原序列长度乘num这么简单，因为0时刻不移动，相当于以0时刻为基准，向正负无穷方向拉伸，所以相当于把序列的初始、结束时间扩大num倍。原序列的长度为(end-begin+1)，结果序列的长度为(end\*num-begin\*num+1)。

* 1. 下采样中0时刻的保留：

传入的num为采样率，通过循环中每次将循环的i=i+num来控制采样频率。

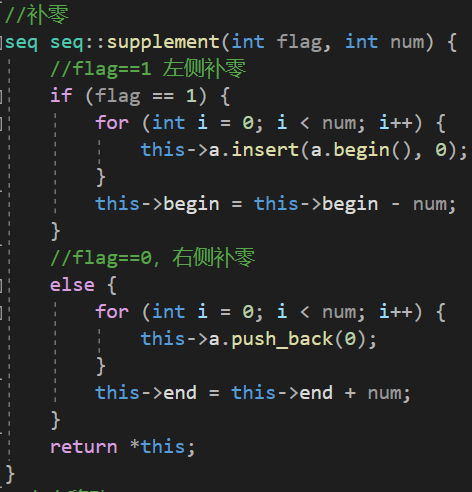
通过将第一个采样点设置为abs(begin)%num来确保下采样中一定能采到0时刻的数据，并保证0时刻的数据不变（相当于以0时刻数据为中心向两侧以每num个数据采一次样）。

1. 实验设计
   1. **前后补零：**

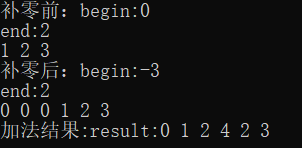
前补零（flag为1时在左侧补零）通过insert函数在0位置处插入num个0；

后补零（flag为0时在右侧补零）通过vector封装的push\_back函数向后添加num个0。

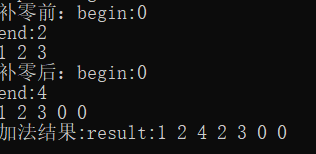
数组的实现方式、思路见第三部分核心问题1。



测试数据：A={1,2,3} ，n=0:2，在左侧补3个0：

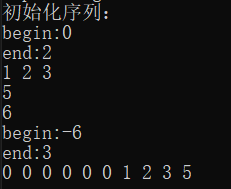


测试数据：A={1,2,3} ，n=0:2，在右侧补2个0：



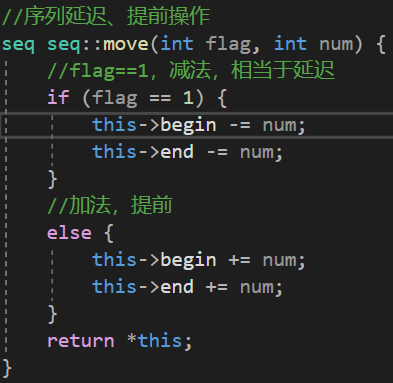
无限长序列可以进行前向补零：

初始化为A={1,2,3} n=0:2。在无限长序列中输入5，并在前向补6个零：

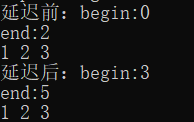


* 1. 序列延迟、提前操作：

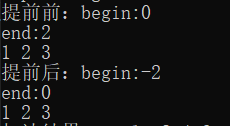
修改序列的begin和end后将修改后的序列返回即可：



测试数据：A={1,2,3} ，n=0:2，延迟3：



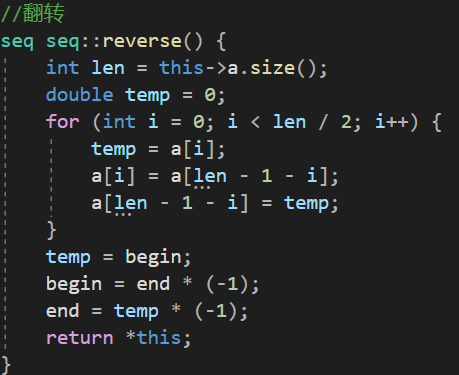
测试数据：A={1,2,3} ，n=0:2，提前2：



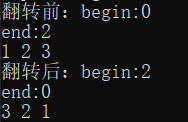
无法操作无限长序列。

* 1. 序列反转操作：

以中间数为轴，两侧的数据进行交换，以得到反转后的vector，然后修改序列的begin和end值即可：



测试数据：A={1,2,3} ，n=0:2 ：

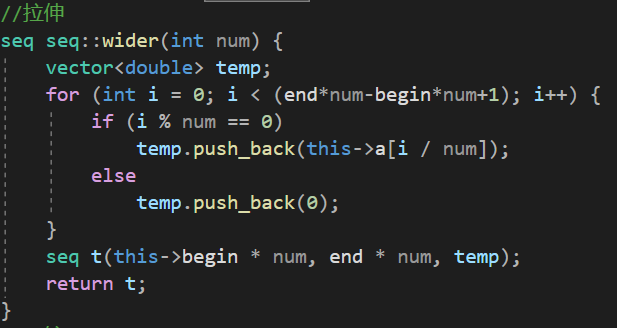


无法操作无限长序列。

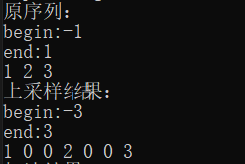
* 1. 序列拉伸操作（上采样)

原序列的长度为(end-begin+1)，结果序列的长度为(end\*num-begin\*num+1)，详细解释见第三部分核心问题3。

注意：由于上采样的结果序列中包含所有原序列中的值，所以0时刻不需要特殊考虑（与下采样不同），所以通过i循环拉伸后的长度次，通过i与采样率n除法的余数控制插入的值为原数据还是0，最后得到上采样后的序列。



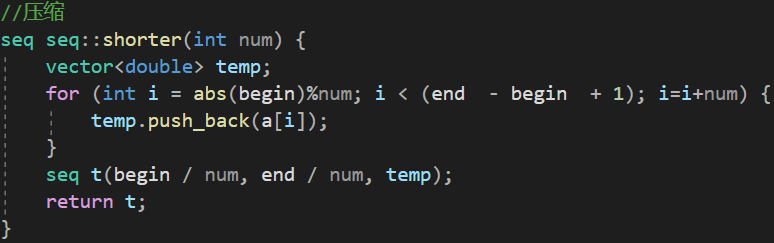
测试数据：A={1,2,3} ，n=-1:1，上采样率为3：



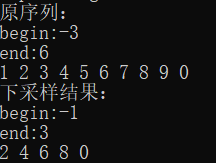
无法操作无限长序列。

* 1. 序列压缩操作（下采样)

注意如何从头遍历vector过程中保证0时刻可以被采样，达到以0时刻为基准，从左右两侧向0时刻压缩num倍的效果，详细解释见第三部分核心问题4。



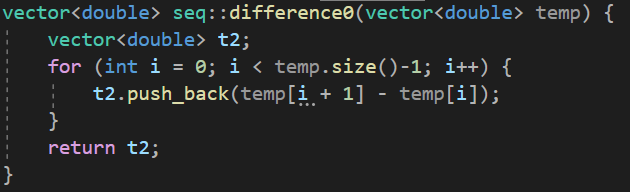
测试数据：A={1,2,3,4,5,6,7,8,9,0} ，n=--3:6，上采样率为2：



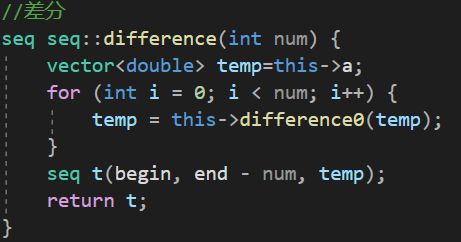
无法操作无限长序列。

* 1. 序列差分操作

N重差分即为在N-1差分结果的基础上再进行一次差分操作，所以先构造单次差分：

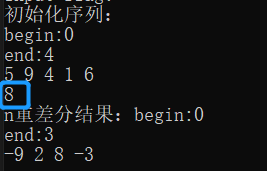


在多重差分中循环调用：



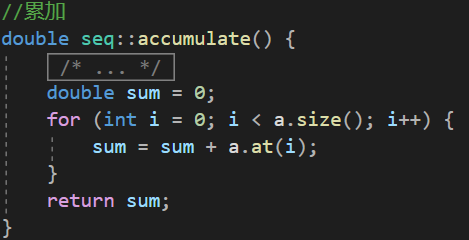
处理无限长序列：

测试数据：A={5,9,4,1,6} n=0:4 输入8后，序列变为5,9,4,1,6,8，二重积分结果为-9,2,8,-3，符合预期：



* 1. 序列累加操作

将序列中所有值进行累加，将结果返回：



测试数据：

A={1,2,3} n=0:2,输入8、2得到14、16，符合预期：

