```
#include <iostream>
#include "ThreadGroup.h"
using namespace std;

const int N_THREADS = 2;

int encode(int v) {
      // do something time-consuming (and arbitrary)
      for (int i = 0; i < 500; i++)
            v = ((v * v) + v) % 10;
      return v;
}

int decode(int v) {
      // do something time-consuming (and arbitrary)
      return encode(v);
}

/**
 * @class - InputArray
 * Contains properties data and length
 * data is the pointer to the sharedData array to be passed to the threads
 * length is the integer that represents the length of the shared data
 */
class InputArray {
    public:
    int *data;
    int length;
};

/**
 * @class - EncodeThread
 * @method - operation()
 * @param id int that represents the id of the thread
 * @param sharedData pointer to sharedData that the thread works on
 * Encodes every element present in the sharedData
 */
class EncodeThread {
    public:
      void operator()(int id, void *sharedData) {
        auto *ourData = (InputArray*)sharedData;
        for (int i = id; i < ourData->length; i += N_THREADS) {
          ourData->data[i] = encode(ourData->data[i]);
        }
      }
};

/**
 * @class - DecodeThread
 * @method - operation()
 * @param id int that represents the id of the thread
 * @param sharedData pointer to sharedData that the thread works on
 * Decodes every element present in the sharedData
 */
class DecodeThread {
    public:
    void operator()(int id, void *sharedData) {
        auto *ourData = (InputArray*)sharedData;
        for (int i = id; i < ourData->length; i += N_THREADS) {
            ourData->data[i] = decode(ourData->data[i]);
        }
    }
};

void prefixSums(int *data, int length) {
      // Create Object of InputArray to be used of the Threads
    InputArray inputData;
    inputData.data = data;
    inputData.length = length;

      // Create Encoding Threads and start encoding
    ThreadGroup<EncodeThread> encoders;
    for (int i = 0; i < N_THREADS; i++)
            encoders.createThread(i, &inputData);
      encoders.waitForAll();

      // Find prefix sum on the encoded values, in main thread
    for (int i = 1; i < length; i++) {
            data[i] += data[i - 1];
      }

      // Create Decode Threads
    ThreadGroup<DecodeThread> decoders;
    for (int i = 0; i < N_THREADS; i++)
            decoders.createThread(i, &inputData);
      decoders.waitForAll();
}

int main() {
      int length = 1000 * 1000;

      // make array
      int *data = new int[length];
      for (int i = 1; i < length; i++)
            data[i] = 1;
      data[0] = 6;

      // transform array into converted/deconverted prefix sum of original
      prefixSums(data, length);

      // printed out result is 6, 6, and 2 when data[0] is 6 to start and the rest
1
      cout << "[0]: " << data[0] << endl
                  << "[" << length/2 << "]: " << data[length/2] << endl
                  << "[end]: " << data[length-1] << endl;

    delete[] data;
      return 0;
}
```