Peterson Algorithm

Problem: Given 2 process $i$ and $j$, you need to write a program that can guarantee mutual exclusion between the two without any additional hardware support.

Please first read the following code (Peterson Algorithm in c++) and then write it (skip comments):

```cpp
#include <iostream>
#include <thread>
using namespace std;

int flag[2];
int turn;

const int loop = 1000;
int ans = 0;

// Initialize lock by resetting the desire of both the threads to acquire the locks. And, giving turn to one of them.
void lock_init(){
    flag[0] = flag[1] = 0;
    turn = 0;
}

// Executed before entering critical section
void lock(int self){
    flag[self] = 1;
    turn = 1-self;
    while (flag[1-self]==1 && turn==1-self) ;
}

// Executed after leaving critical section
void unlock(int self){
    flag[self] = 0;
}

// A Sample function run by two threads created in main()
void func(int self){
    int i = 0;
    cout<<"Thread Entered: "<<self<<" \n";
}
for (int i=0; i<loop; i++){
    lock(self);
    ans++;
    unlock(self);
}

// Driver code
int main()
{
    // Initialized the lock then fork 2 threads
    thread p1(func, 0);
    thread p2(func, 1);
    lock_init();

    // Wait for the threads to end.
    p1.join();
    p2.join();

    cout<<"Actual Count: "<<ans<<" | Expected Count: "<<(2*loop)<<endl;
    return 0;
}

- What is your result?
- Is the Peterson algorithm a reliable solution?
- Change loop to 100000000. What is the result?
- What do we learn from this example?