UMMC

Jiamu Li & Frank Tang & Edward Wang

July 17, 2022

Hello, world!

```cpp
#include <chrono>
#include <future>
#include <iostream>
#include <random>
#include <string>
#include <thread>
#include <vector>

using namespace std::chrono;

std::random_device rd;
std::mt19937 rng(rd());

const char particles[] = "XY";

unsigned int num_threads = std::thread::hardware_concurrency();

std::string gen_tube(int length) {
  std::uniform_int_distribution<int> pick(0, 1);
  std::string tube;
  for (int i = 0; i < length; i++) {
    tube += particles[pick(rng)];
  }
  return tube;
}

std::string annihilate(std::string tube) {
  std::string res = "";
  for (char &c : tube) {
    if (res.size() && res.back() == c && c == 'X')
      res.pop_back();
    else
      res.push_back(c);
  }
  return res;
}

bool check_consec_x(std::string tube) {
  for (int i = 0; i < tube.length() - 1; i++) {
    if (tube[i] == tube[i + 1])
      return 0;
  }
  return 1;
}

double prob_consec(int length, int runs) {
  long count = 0;
  for (int i = 0; i < runs; i++) {
    if (check_consec_x(gen_tube(length)))
      count++;
  }
  return count/double(runs);
}

int main() {
  auto start = high_resolution_clock::now();

```

```
58    int runs = 10000000;
59
60    // std::vector<std::future<long>> threads;
61    //
62    // for (int i = 8; i < num_threads+8; i++) {
63    //   threads.push_back(std::async(std::launch::async, count_check, i, runs));
64    // }
65    // for (auto &t : threads) {
66    //   std::cout << t.get() << '\n';
67    // }
68
69    for (int i = 8; i < 16; i++) {
70      std::cout << prob_consec(i, runs) << '\n';
71    }
72
73    auto stop = high_resolution_clock::now();
74    auto elapsed = duration_cast<milliseconds>(stop - start);
75    std::cout << elapsed.count() << "ms\n";
76  }
```