Петров Андрей, 13 группа 3 курс. N17 std::search

Задание. В строке найти позицию искомого слова.

Замеры времени

|  |  |  |
| --- | --- | --- |
| Размер файла/ Количество строк, байты/к-во | Время выполнения | |
| Последовательная программа | Параллельная программа |
| 10000, строк 1 | 1.783600 | 1.644600 |
| 100000, строк 1 | 14.137700 | 9.136700 |

Код программы:

#include <string\_view>  
#include <functional>  
#include <algorithm>  
#include <iostream>  
#include <vector>  
#include <execution>  
#include <fstream>  
#include <windows.h>  
#include <cmath>  
#include <string>  
  
using namespace std;  
  
const string needle{"odio"};  
vector<pair<int, long>> result = {};  
  
long search(string\_view text) {  
 auto it = search(text.begin(), text.end(), needle.begin(), needle.end());  
 if (it != text.end()) {  
 return it - text.begin();  
 }  
  
 return -1;  
}  
  
long searchIsParallel(string\_view text) {  
 auto it = search(execution::par, text.begin(), text.end(), needle.begin(), needle.end());  
 if (it != text.end()) {  
 return it - text.begin();  
 }  
  
 return -1;  
}  
  
  
double program(){  
 LARGE\_INTEGER liFrequency, liStartTime, liFinishTime;  
 double dElapsedTime;  
 QueryPerformanceFrequency(&liFrequency);  
  
 ifstream file("../file.txt");  
 string buffer;  
 int line = 0;  
  
 QueryPerformanceCounter(&liStartTime);  
 while (getline(file, buffer)) {  
 line++;  
 file >> buffer;  
 long position = search(buffer);  
 if(position != -1) {  
 result.push\_back(pair<int, long>(line, position));  
 }  
 }  
 file.close();  
  
  
 QueryPerformanceCounter(&liFinishTime);  
 dElapsedTime = 1000.0 \* (liFinishTime.QuadPart - liStartTime.QuadPart) / liFrequency.QuadPart;  
 return dElapsedTime;  
}  
  
double programParallel(){  
 LARGE\_INTEGER liFrequency, liStartTime, liFinishTime;  
 double dElapsedTime;  
 QueryPerformanceFrequency(&liFrequency);  
  
 ifstream file("../file.txt");  
 string buffer;  
 int line = 0;  
  
 QueryPerformanceCounter(&liStartTime);  
 while (getline(file, buffer)) {  
 line++;  
 file >> buffer;  
 long position = searchIsParallel(buffer);  
 if(position != -1) {  
 result.push\_back(pair<int, long>(line, position));  
 }  
 }  
 file.close();  
  
  
 QueryPerformanceCounter(&liFinishTime);  
 dElapsedTime = 1000.0 \* (liFinishTime.QuadPart - liStartTime.QuadPart) / liFrequency.QuadPart;  
 return dElapsedTime;  
}  
  
int main() {  
 double time = 0;  
  
 time = program();  
 printf("finished! Time: %f\n", time);  
 cout << "result = { \n";  
 for (pair<int, long> res : result) {  
 cout << "\tline: " << res.first << " position: " << res.second << ",\n";  
 }  
 cout << "}; \n";  
 result.clear();  
  
 //==================================================  
 time = programParallel();  
 printf("Parallel finished! Time: %f\n", time);  
 cout << "result = { \n";  
 for (pair<int, long> res : result) {  
 cout << "\tline: " << res.first << " position: " << res.second << ",\n";  
 }  
 cout << "}; \n";  
}