**Отчет**

**Код программы:**

#include <iostream>

#include <algorithm>

#include <future>

#include <vector>

#include <chrono>

using namespace std;

int max\_(int\* start, int\* end)

{

return \*max\_element(start, end);

}

int parallel\_max(std::vector<int>& v)

{

using Task\_type = int(int\*, int\*);

packaged\_task<Task\_type> pt0{ max\_ };

packaged\_task<Task\_type> pt1{ max\_ };

packaged\_task<Task\_type> pt2{ max\_ };

packaged\_task<Task\_type> pt3{ max\_ };

packaged\_task<Task\_type> pt4{ max\_ };

packaged\_task<Task\_type> pt5{ max\_ };

future<int> f0{ pt0.get\_future() };

future<int> f1{ pt1.get\_future() };

future<int> f2{ pt2.get\_future() };

future<int> f3{ pt3.get\_future() };

future<int> f4{ pt4.get\_future() };

future<int> f5{ pt5.get\_future() };

int\* first = &v[0];

int delta = v.size() / 6;

thread t1{ move(pt0), first, first + delta };

thread t2{ move(pt1), first + delta, first + 2 \* delta };

thread t3{ move(pt2), first + 2 \* delta, first + 3 \* delta };

thread t4{ move(pt3), first + 3 \* delta, first + 4 \* delta };

thread t5{ move(pt4), first + 4 \* delta, first + 5 \* delta };

thread t6{ move(pt5), first + 5 \* delta, first + 6 \* delta };

t1.join();

t2.join();

t3.join();

t4.join();

t5.join();

t6.join();

std::vector<int> results{ f0.get(), f1.get(), f2.get(), f3.get(), f4.get(), f5.get() };

return \*max\_element(results.begin(), results.end());

}

int main()

{

srand(0);

unsigned int length = 1000000;

std::vector<int> mas;

mas.reserve(length);

for (int i = 0; i < length; ++i)

{

mas.push\_back(rand());

}

auto start = std::chrono::high\_resolution\_clock::now();

int result = \*max\_element(mas.begin(), mas.end());

auto end = std::chrono::high\_resolution\_clock::now();

std::cout << "Time of sequential algorithm execution: " << std::chrono::duration\_cast<std::chrono::microseconds>(end - start).count() << " ms" << std::endl;

std::cout << "Result = " << result << std::endl;

std::cout << "--------------------------------------" << std::endl;

start = std::chrono::high\_resolution\_clock::now();

result = parallel\_max(mas);

end = std::chrono::high\_resolution\_clock::now();

std::cout << "Time of simple parallel algorithm execution: " << std::chrono::duration\_cast<std::chrono::microseconds>(end - start).count() << " ms" << std::endl;

std::cout << "Result = " << result << std::endl;

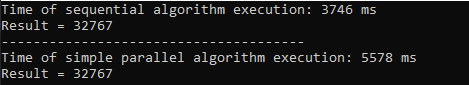
std::cout << "--------------------------------------" << std::endl;

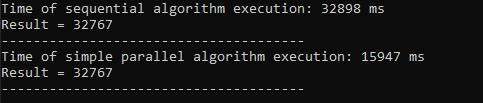
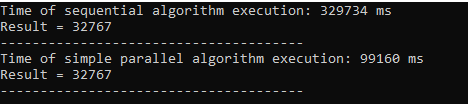
return 0;

}

**Результат:**

* **Размер 100000:**

****

* **Размер 1000000: **
* **Размер 10000000:**