**Отчет**

**Код программы (последовательная):**

#include <iostream>  
#include <vector>  
#include <time.h>  
#include <stdlib.h>  
#include <omp.h>  
#include <chrono>  
using namespace std;  
  
  
int minNumber(vector<int> mas)  
{  
 int minNumber = INT\_MAX;

for(int i = 0; i < mas.size(); i++)  
 {  
 minNumber = std::min(minNumber, mas[i]);  
 }  
 return minNumber;  
}  
  
int main()  
{  
 int size = 10000;  
 vector <vector<int>> myVector(size);  
  
  
 srand(time(0));  
 for (int i = 0; i < size; i++)  
 {  
 for (int j = 0; j < size; j++)  
 {  
 myVector[i].push\_back( rand());  
 }  
 }  
  
 vector<int> masMin(size);  
 auto t1 = std::chrono::system\_clock::now();  
  
 for (int i = 0; i < size; i++) {  
 masMin[i] = minNumber(myVector[i]);  
 }  
 int maxNumber = 0;

for (int i = 0; i < size; i++)  
 {  
 maxNumber = std::max(masMin[i], maxNumber);  
 }  
 cout << maxNumber<< endl;  
 auto t2 = std::chrono::system\_clock::now();  
 auto time = 1.0 \* (t2 - t1).count() \* std::chrono::system\_clock::period::num / std::chrono::system\_clock::period::den;  
  
 std::cout << "Time: " << time << std::endl;  
}

**Код программы (задание 1):**

#include <iostream>  
#include <vector>  
#include <time.h>  
#include <stdlib.h>  
#include <omp.h>  
#include <chrono>  
using namespace std;  
  
  
int minNumber(vector<int> mas)  
{  
 int minNumber = INT\_MAX;

for(int i = 0; i < mas.size(); i++)  
 {  
 minNumber = std::min(minNumber, mas[i]);  
 }  
 return minNumber;  
}  
  
int main()  
{  
 int size = 10000;  
 vector <vector<int>> myVector(size);  
  
  
 srand(time(0));  
 for (int i = 0; i < size; i++)  
 {  
 for (int j = 0; j < size; j++)  
 {  
 myVector[i].push\_back( rand());  
 }  
 }  
  
 vector<int> masMin(size);  
 auto t1 = std::chrono::system\_clock::now();  
  
#pragma omp parallel for num\_threads(6)  
 for (int i = 0; i < size; i++) {  
 masMin[i] = minNumber(myVector[i]);  
 }  
 int maxNumber = 0;  
  
 for (int i = 0; i < size; i++)  
 {  
 maxNumber = std::max(masMin[i], maxNumber);  
 }  
 cout << maxNumber<< endl;  
 auto t2 = std::chrono::system\_clock::now();  
 auto time = 1.0 \* (t2 - t1).count() \* std::chrono::system\_clock::period::num / std::chrono::system\_clock::period::den;  
  
 std::cout << "Time: " << time << std::endl;  
}

**Код программы (задание 2):**

#include <iostream>  
#include <vector>  
#include <time.h>  
#include <stdlib.h>  
#include <omp.h>  
#include <chrono>  
using namespace std;  
  
  
int minNumber(vector<int> mas)  
{  
 int minNumber = INT\_MAX;  
#pragma omp parallel for reduction(min:minNumber) num\_threads(6)  
 for(int i = 0; i < mas.size(); i++)  
 {  
 minNumber = std::min(minNumber, mas[i]);  
 }  
 return minNumber;  
}  
  
int main()  
{  
 int size = 10000;  
 vector <vector<int>> myVector(size);  
  
  
 srand(time(0));  
 for (int i = 0; i < size; i++)  
 {  
 for (int j = 0; j < size; j++)  
 {  
 myVector[i].push\_back( rand());  
 }  
 }  
  
 vector<int> masMin(size);  
 auto t1 = std::chrono::system\_clock::now();  
  
#pragma omp parallel for num\_threads(6)  
 for (int i = 0; i < size; i++) {  
 masMin[i] = minNumber(myVector[i]);  
 }  
 int maxNumber = 0;  
#pragma omp parallel for reduction(max:maxNumber) num\_threads(6)  
 for (int i = 0; i < size; i++)  
 {  
 maxNumber = std::max(masMin[i], maxNumber);  
 }  
 cout << maxNumber<< endl;  
 auto t2 = std::chrono::system\_clock::now();  
 auto time = 1.0 \* (t2 - t1).count() \* std::chrono::system\_clock::period::num / std::chrono::system\_clock::period::den;  
  
 std::cout << "Time: " << time << std::endl;  
}

**Результат:**

* **Размер 1000:**

**Последовательное выполнение 0.0063052**

**1 задание 0.0047268**

**2 задание 0.0051302**

**Ускорение 1 1.3339257**

**Ускорение 2 1.2290359**

* **Размер 10000:**

**Последовательное выполнение 0.519463**

**1 задание 0.143433**

**2 задание 0.112475**

**Ускорение 1 3.6216421**

**Ускорение 2 4.6184752**

* **Размер 30000:**

**Последовательное выполнение 4.90851**

**1 задание 1.21643**

**2 задание 1.01253**

**Ускорение 1 4.0351767**

**Ускорение 2 4.8477674**