Текст программы:

```
#include <iostream>
#include <omp.h>
#include <queue>
#include <fstream>


#define vec_num 10000000
#define length 25
#define threads_number 2
#define file_vec "in.txt"


void generate_vectors_to_file() {
    std::ofstream write("in.txt");
    for (int i = 0; i < vec_num; ++i) {
        for (int j = 0; j < length; ++j) {
            int n = abs(rand() % 23);
            write << n << ' ';
        }
        write << '\n';
    }
}


typedef long long (*function)();
long long parallel() {
    bool finished = false;
    std::queue<int*> queue;
    long long mult_res = 0;
    omp_lock_t ompLock;
    omp_init_lock(&ompLock);


#pragma omp parallel
    {
#pragma omp sections
        {
#pragma omp section
            {
                std::ifstream fin(file_vec);
                int vec_element = 0;
                for (int i = 0; i < vec_num; i++) {
                    int* pInt = new int[length];
                    for (int j = 0; j < length; j++) {
                        fin >> vec_element;
                        pInt[j] = vec_element;
                    }
                    omp_set_lock(&ompLock);
                    queue.push(pInt);
                    omp_unset_lock(&ompLock);
                }
                omp_set_lock(&ompLock);
                finished = true;
                omp_unset_lock(&ompLock);
                fin.close();
            }
#pragma omp section
            {
                int* first;
                int* second;
                while (true) {
                    omp_set_lock(&ompLock);
                    if (queue.empty() && finished) {
                        omp_unset_lock(&ompLock);
                        break;
                    }
                    omp_unset_lock(&ompLock);
                    while (queue.size() < 2) {}
```

```
                omp_set_lock(&ompLock);
                first = queue.front();
                queue.pop();
                second = queue.front();
                queue.pop();
                omp_unset_lock(&ompLock);
                long long tmp = 0;
                for (int j = 0; j < length; j++) {
                    tmp += first[j] * second[j];
                }
                mult_res += tmp;
                delete[] first;
                delete[] second;
            }
        }
    }
    omp_destroy_lock(&ompLock);
    return mult_res;
}



long long scalar_mult() {
    std::ifstream fin(file_vec);
    std::queue<int*> queue;
    int num = 0;
    for (int i = 0; i < vec_num; i++) {
        int *pInt = new int[length];
        for (int j = 0; j < length; j++) {
            fin >> num;
            pInt[j] = num;
        }
        queue.push(pInt);
    }
    fin.close();
    int* first;
    int* second;
    long long mult_res = 0;
    while (!queue.empty()) {
        first = queue.front();
        queue.pop();
        second = queue.front();
        queue.pop();
        long long tmp = 0;
        for (int j = 0; j < length; j++) {
            tmp += first[j] * second[j];
        }
        mult_res += tmp;
        delete[] first;
        delete[] second;
    }
    return mult_res;
}


void do_task(const std::string &name, function func) {
    std::cout << name << std::endl;
    double start, end;
    start = omp_get_wtime();
    long long result = func();
    end = omp_get_wtime();
    std::cout << "Time is " << (end - start) * 1000 << " ms" << std::endl;
    std::cout << "Result is " << result << std::endl;
}


int main() {
    generate_vectors_to_file();
    do_task("Single thread:", scalar_mult);
```

```
    std::cout << std::endl;
    do_task("Parallel count:", parallel);
    return 0;
}
```

Результаты экспериментов:

Размерность задачи	Последовательная программа, мс	Параллельная программа на 2 потока			Параллельная программа на 4 потока		
		Время выполнения	Ускорение	Эффективность	Время выполнения	Ускорение	Эффективность
10^4	14.0378	17.952	0.7819630125	0.3909815062	17.256	0.8135025498	0.2033756375
10^6	1514.11	1362.3	1.111436541	0.5557182706	1245.5	1.215664392	0.303916098
10^7	16710.3	14353.8	1.164172554	0.5820862768	15021.2	1.112447741	0.2781119351