Текст программы:

```
matrix_t genMatrix(std::size_t n_rows, std::size_t n_columns) {
    std::mt19937 gen(std::random_device{}());
    std::uniform_real_distribution dist(0.0, 100.0);

    auto rand_val = [&](){return dist(gen);};
    auto generate_range = [&](auto&& row){std::ranges::generate(row, rand_val);};

    matrix_t matrix(n_rows, std::vector<double>(n_columns));
    std::ranges::for_each(matrix, generate_range);

    return matrix;
}

double maxInMin(const matrix_t& vector) {
    double max_value = vector[0][0];
    auto find_max_in_mins = [&](auto&& row){ max_value = std::max(max_value, std::ranges::min(row)); };
    std::ranges::for_each(vector, find_max_in_mins);

    return max_value;
}

double innerMaxInMin(const matrix_t& vector) {
    double max_value = vector[0][0];
    for (auto&& row : vector) {
        std::vector<double> mins(omp_get_num_procs());

#pragma omp parallel for
        for (auto&& elem: row) {
            double& thread_min = mins[omp_get_thread_num()];
            thread_min = std::min(thread_min, elem);
```

```
#pragma omp parallel for
        for (auto&& elem: row) {
            double& thread_min = mins[omp_get_thread_num()];
            thread_min = std::min(thread_min, elem);
        }
        max_value = std::max(max_value, std::ranges::max(mins));
    }

    return max_value;
}

double outerMaxInMin(const matrix_t& vector) {
    double max_value = vector[0][0];

    #pragma omp parallel for
    for (auto&& row : vector) {
        auto min_in_row = std::ranges::min(row);

        #pragma omp critical
        max_value = std::max(max_value, min_in_row);
    }

    return max_value;
}

void time() {
    static constexpr std::string_view format = "Result: {:.4}; time: {:.8}; overhead: {:.8}\n";
    static const std::size_t n_threads = omp_get_num_procs();

    for (std::size_t dim = 5'000; dim ⩽ 15'000; dim += 5'000) {
        auto matrix = genMatrix(dim, dim);
        auto [seq_time, seq_result] = flow::timer<>::duration_r(maxInMin, matrix);
        auto [par_inner_time, par_inner_result] = flow::timer<>::duration_r(innerMaxInMin, matrix);
        auto [par_outer_time, par_outer_result] = flow::timer<>::duration_r(outerMaxInMin, matrix);
```

```cpp
void time() {
    static constexpr std::string_view format = "Result: {:.4}; time: {:.8}; overhead: {:.8}\n";
    static const std::size_t n_threads = omp_get_num_procs();

    for (std::size_t dim = 5'000; dim ⩽ 15'000; dim += 5'000) {
        auto matrix = genMatrix(dim, dim);
        auto [seq_time, seq_result] = flow::timer<>::duration_r(maxInMin, matrix);
        auto [par_inner_time, par_inner_result] = flow::timer<>::duration_r(innerMaxInMin, matrix);
        auto [par_outer_time, par_outer_result] = flow::timer<>::duration_r(outerMaxInMin, matrix);

        fmt::print("Dim: {}x{}\n", dim, dim);
        fmt::print(format, seq_result, seq_time, 0.0);
        fmt::print(format, par_inner_result, par_inner_time, par_inner_time * n_threads - seq_time);
        fmt::print(format, par_outer_result, par_outer_time, par_outer_time * n_threads - seq_time);
        fmt::print("-----------------------------------------------------------------\n\n");
    }
}

int main() {
    omp_set_num_threads(omp_get_num_procs());
    time();

    return 0;
}
```

Результаты экспериментов:

Размерность задачи	Последовательная программа, мкс	Параллельная программа			Разделяй и властвуй		
		Время выполнения	Ускорение	Эффективность	Время выполнения	Ускорение	Эффективность
10^4	305,6	1235,4	0,25632	0,06352	1256,3	0,21256	0,05325
10^5	3256,2	2012,5	1,4520	0,39525	1834,2	1,56325	0,39533

10^6	30523,7	14256,3	2,4023	0,60232	14921,2	2,00215	0,49652