Петров Андрей, 13 группа 3 курс

Задача 1, стр. 131

4. Разработайте программу решения задачи поиска максимального значения среди минимальных элементов строк матрицы (такая задача имеет место для решения матричных игр)

$$y = \max_{1 \le i \le N} \min_{1 \le j \le N} a_{ij},$$

1. Создадим последовательную программу

```
int find_row_max(vector<int> &matrix) {
    int max_value = 0;
    for (int i = 0; i < matrix.size(); i++) {
        if (matrix[i] > max_value) max_value = matrix[i];
    }
    return max_value;
}

float find_min_value(vector<int> &matrix) {
    int min_value = INT_MAX;
    for (int i = 0; i < matrix.size(); i++) {
        if (matrix[i] < min_value) min_value = matrix[i];
    }
    return min_value;
}

int find_total_max(vector<vector<int>> &matrix, int row) {
    vector<int> min_elements(row);

    for (int i = 0; i < row; i++)
        min_elements[i] = find_min_value(matrix[i]);
    }
    return find_row_max(min_elements);
}
```

2. Создадим параллельную программу

```
int find_row_max(vector<int> &matrix) {
    int max_value = 0;

#pragma omp parallel for
    for (int i = 0; i < matrix.size(); i++) {
        if (matrix[i] > max_value) {
#pragma omp critical
            max_value = matrix[i];
        }
    }
    return max_value;
}

float find_min_value(vector<int> &matrix) {
    int min_value = INT_MAX;
```

```
#pragma omp parallel for
    for (int i = 0; i < matrix.size(); i++) {
        if (matrix[i] < min_value) {
#pragma omp critical
            min_value = matrix[i];
        }
    }
    return min_value;
}

int find_total_max(vector<vector<int>> &matrix, int row) {
    vector<int> min_elements(row);

    for (int i = 0; i < row; i++) {
        min_elements[i] = find_min_value(matrix[i]);
    }
    return find_row_max(min_elements);
}
```

3. Замерим время

```
int main(int argc, char **argv) {
    LARGE_INTEGER liFrequency, liStartTime, liFinishTime;
    double dElapsedTime;
    QueryPerformanceFrequency(&liFrequency);

    vector<vector<int>> matrix(ROW_NUMBER, vector<int>(COLUMN_NUMBER));
    fill_matrix(matrix, ROW_NUMBER, COLUMN_NUMBER);

    QueryPerformanceCounter(&liStartTime);
    int total_max = find_total_max(matrix, ROW_NUMBER);
    QueryPerformanceCounter(&liFinishTime);

    dElapsedTime = 1000.0 * (liFinishTime.QuadPart - liStartTime.QuadPart) /
liFrequency.QuadPart;
    printf("Time:  %f ms\n", dElapsedTime);
    printf("Total max: %d\n", total_max);
}
```

Задача 2, стр. 132

10. Разработке программу для задачи 4 с использованием распараллеливания циклов разного уровня вложенности. Выполните вычислительные эксперименты и сравните полученные результаты. Оцените величину накладных расходов на создание и завершение потоков.

1. Создадим версию программы с вложенным распараллеливанием

```
int find_total_max_nesting(vector<vector<int>> &matrix, int row, int cols) {
    int max_value = INT_MIN;
    omp_set_nested(true);
```

```
#pragma omp parallel for reduction(max:max_value)
    for (int i = 0; i < row; i++) {
        int min_value = INT_MAX;

#pragma omp parallel for reduction(min:min_value)
        for (int j = 0; j < cols; j++) {
            min_value = min(min_value, matrix[i][j]);
        }
        max_value = max(max_value, min_value);
    }
    return max_value;
}
```

Произведем вычислительные эксперименты и составим таблицу

Размерность матрицы	Последовательная программа	Параллельная программа	Параллельная программа с вложенностью	Ускорение	
				1	2
100 x 100	0.230100	0.105100	0.176000	0.4568	0.76488
1 000 x 1 000	10.390000	8.761400	14.170500	0.84325	1.36385
10 000 x 10 000	796.968400	775.129100	1086.539200	0.97259	1.36334