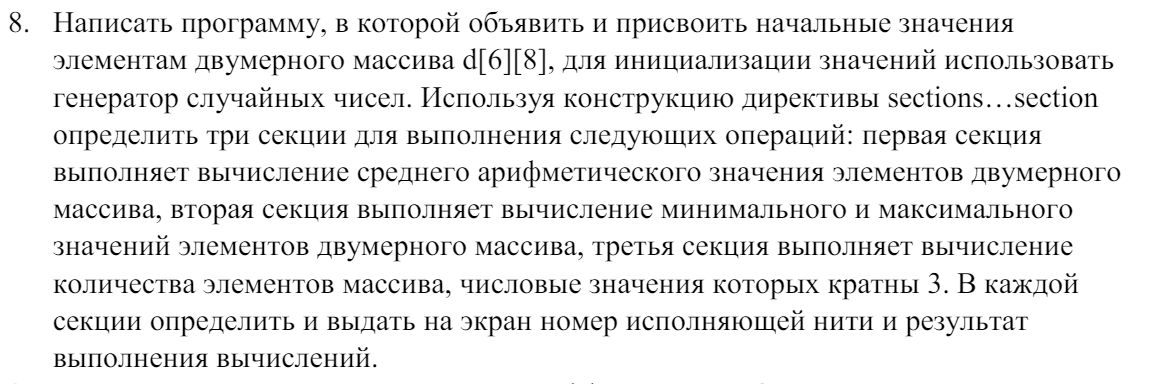
Отчет №7

Гренц Елизавета Алексеевна группа 932220



**Описание работы:**

Начинаем с инициализации матрицы 6x8 случайными числами от 0 до 99, после инициализации матрицы программа выводит содержимое матрицы на консоль.

Вводим блок параллельных секций с помощью #pragma omp parallel sections директивы. Этот блок содержит три раздела, каждый из которых будет выполняться одновременно разными потоками.

1. Вычисление среднего арифметического : в первом разделе каждый поток вычисляет сумму всех элементов в матрице ( local\_sum).
2. Расчет минимального и максимального значений : во втором разделе каждый поток вычисляет минимальные и максимальные значения элементов матрицы.
3. Подсчет кратных 3 : В третьем разделе каждый поток подсчитывает количество элементов в матрице, которые делятся на 3 ( local\_count\_mult\_of\_3).

**Код для вставки :**

#include <stdio.h>

#include <omp.h>

#include <stdlib.h>

#include <limits.h>

#define ROWS 6

#define COLS 8

int main() {

int d[ROWS][COLS];

int i, j;

double sum = 0.0;

int min = 101;

int max = 0;

int count\_mult\_of\_3 = 0;

#pragma omp parallel for private(i, j) shared(d)

for (i = 0; i < ROWS; i++) {

for (j = 0; j < COLS; j++) {

d[i][j] = rand() % 100;

}

}

//Р’С‹РІРѕРґ РјР°С‚СЂРёС†С‹

printf("Matrix d:\n");

for (i = 0; i < ROWS; i++) {

for (j = 0; j < COLS; j++) {

printf("%3d ", d[i][j]);

}

printf("\n");

}

printf("\n");

#pragma omp parallel sections shared(sum, min, max, count\_mult\_of\_3)

{

//РЅР°С…РѕРґРёРј СЃСЂРµРґРЅРµРµ Р°СЂРёС„Рј

#pragma omp section

{

int thread\_num = omp\_get\_thread\_num();

double local\_sum = 0.0;

for (i = 0; i < ROWS; i++) {

for (j = 0; j < COLS; j++) {

local\_sum += d[i][j];

}

}

#pragma omp atomic

sum += local\_sum;

printf("Thread %d: Arithmetic mean = %.2f\n", thread\_num, local\_sum / (ROWS \* COLS));

}

//РќР°С…РѕРґРёРј РјРёРЅРёРјР°Р»СЊРЅС‹Р№ Рё РјР°РєСЃРёСЊР°Р»СЊРЅС‹Р№ СЌР»

#pragma omp section

{

int thread\_num = omp\_get\_thread\_num();

int local\_min = min;

int local\_max = max;

for (i = 0; i < ROWS; i++) {

for (j = 0; j < COLS; j++) {

if (d[i][j] < local\_min) local\_min = d[i][j];

if (d[i][j] > local\_max) local\_max = d[i][j];

}

}

printf("Thread %d: Min value = %d, Max value = %d\n", thread\_num, local\_min, local\_max);

}

// РќР°С…РѕРґРёРј РІСЃРµ СЌР» РєРѕС‚РѕСЂС‹Рµ РґРµР»СЏС‚СЃСЏ РЅР° 3

#pragma omp section

{

int thread\_num = omp\_get\_thread\_num();

int local\_count\_mult\_of\_3 = 0;

for (i = 0; i < ROWS; i++) {

for (j = 0; j < COLS; j++) {

if (d[i][j] % 3 == 0) local\_count\_mult\_of\_3++;

}

}

printf("Thread %d: Number of elements divisible by 3 = %d\n", thread\_num, local\_count\_mult\_of\_3);

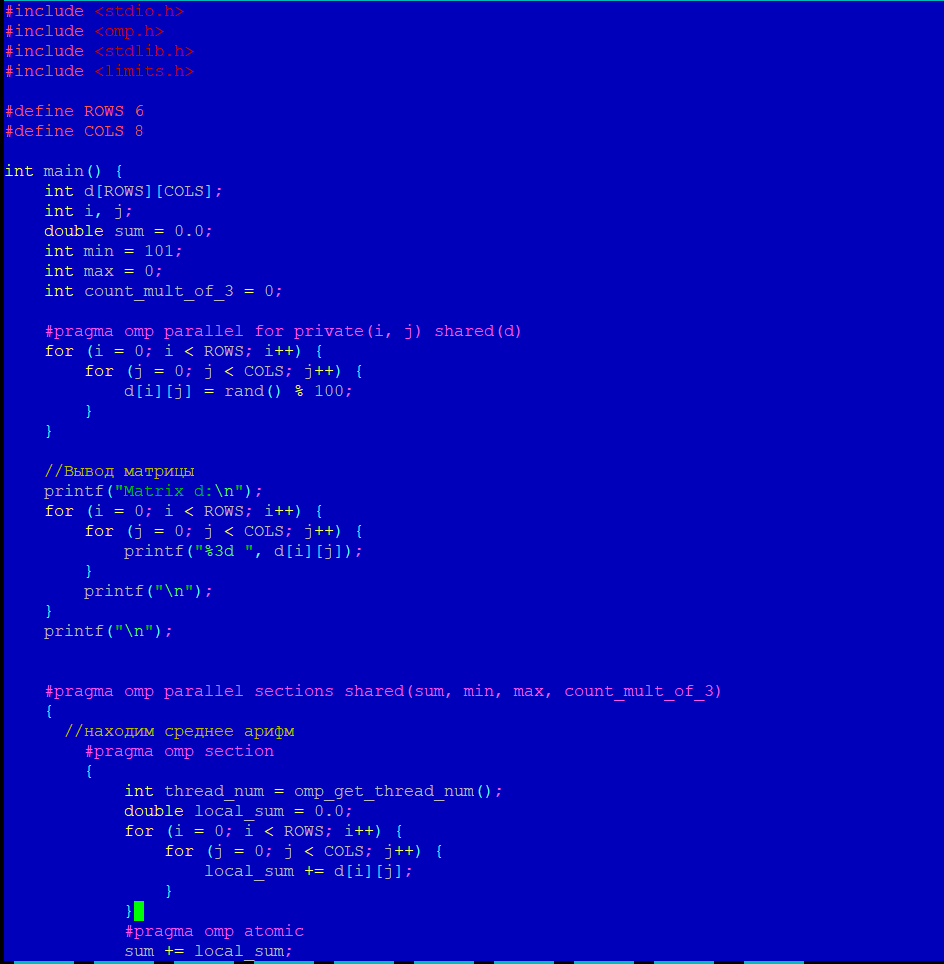
}

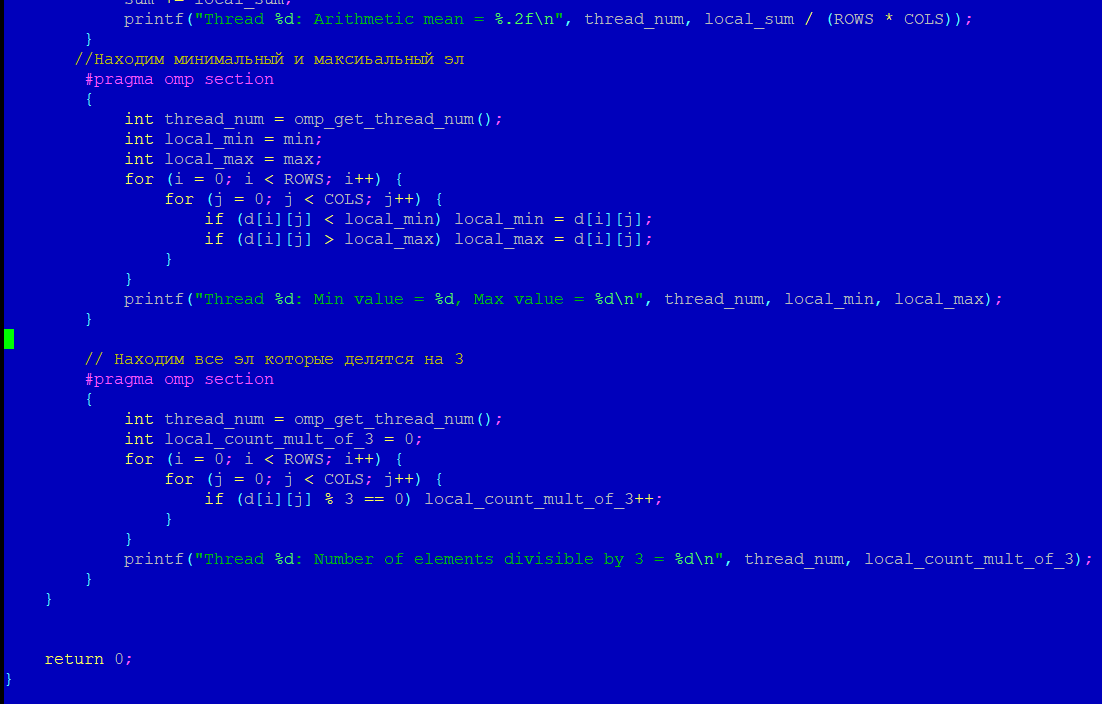
}

return 0;

}

**Скрин\_для \_просмотра**





**Результаты**

**Результат на 2**

Matrix d:

83 86 77 15 93 35 86 92

49 21 62 27 90 59 63 26

40 26 72 36 11 68 67 29

82 30 62 23 67 35 29 2

22 58 69 67 93 56 11 42

29 73 21 19 84 37 98 24

Thread 0: Arithmetic mean = 50.96

Thread 0: Min value = 2, Max value = 98

Thread 1: Number of elements divisible by 3 = 15

**Результат на 3**

Matrix d:

83 86 77 15 93 35 86 92

49 21 62 27 90 59 63 26

40 26 72 36 11 68 67 29

82 30 62 23 67 35 29 2

22 58 69 67 93 56 11 42

29 73 21 19 84 37 98 24

Thread 1: Min value = 2, Max value = 98

Thread 0: Arithmetic mean = 50.96

Thread 2: Number of elements divisible by 3 = 15

**Вывод :**

Я познакомилась с директивой section которая задаёт участок кода внутри секции sections для выполнения одной нитью. В целом, директива section может повысить скорость выполнения кода, позволяя одновременно выполнять разные задачи, оптимизируя нагрузку между нитями