Programowanie współbieżne i rozproszone	PWIR06	
17.05.2022	P2	Szymon Zwoliński

5.1. Dopisz dyrektywę num_threads do programu z PWIR_02_01.cpp. Przetestuj czas wykonywania programu dla dwóch i więcej wątków.

```
#pragma omp parallel for num_threads(20)  shared(matrix) private(i, k)
    for (i = 0; i < MATRIX_H; i++) {
        for (k = 0; k < MATRIX_W; k++) {
            matrix[i][k] = (uint16_t)(rand() % 100);
        }
    }


#pragma omp parallel for num_threads(20) shared(vector) private(i)
    for (i = 0; i < VECTOR_S; i++) {
        vector[i] = (uint16_t)(rand() % 100);
    }
    end = std::chrono::high_resolution_clock::now();

    printf("Fill parallel way in %lu miliseconds\n",
        std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count());

    start = std::chrono::high_resolution_clock::now();
#pragma omp parallel for num_threads(20) shared(matrix, vector, result) private(i, k)
    for (i = 0; i < MATRIX_H; i++) {
        for (k = 0; k < MATRIX_W; k++) {
            result[i] += matrix[i][k] * vector[k];
        }
    }
```

Liczba wątków 2:

```
Fill in 6696 miliseconds
Calculated normal way in 228 miliseconds
Fill parallel way in 6628 miliseconds
Calculated parallel way in 227 miliseconds
```

Liczba wątków 3:

```
Fill in 6647 miliseconds
Calculated normal way in 226 miliseconds
Fill parallel way in 6590 miliseconds
Calculated parallel way in 226 miliseconds

F:\PWIR\PWIR05\02,1\Debug\02,1.exe (proces 15544) zakończono z kodem 0.
```

Liczba wątków 20:

```
Fill in 6659 miliseconds
Calculated normal way in 227 miliseconds
Fill parallel way in 6594 miliseconds
Calculated parallel way in 227 miliseconds

F:\PWIR\PWIR05\02,1\Debug\02,1.exe (proces 6764) zakończono z kodem 0.
```

Wyniki w zależności od ilości wątków różnią się. Wraz z ilością wątków zmniejsza się czas wykonywania polecenia, jednak tylko do określonej liczby wątków

6.1 Przetestuj działanie _01 z klauzulą nowait oraz bez. Sprawdź również działanie na większej ilości wątków.

nowait 2 wątki:

```
Sections - Thread 0 working...
Iteration 0 execute thread 0.
Iteration 1 execute thread 0.
Iteration 2 execute thread 0.
Iteration 3 execute thread 0.
Iteration 4 execute thread 0.
Sections - Thread 1 working...
Iteration 5 execute thread 1.
Iteration 6 execute thread 1.
Iteration 7 execute thread 1.
Iteration 8 execute thread 1.
Iteration 9 execute thread 1.
Parallel normal way 6037 ms
```

Bez nowait 2 wątki:

```
Sections - Thread 0 working...
Sections - Thread 1 working...
Iteration 5 execute thread 1.
Iteration 0 execute thread 0.
Iteration 6 execute thread 1.
Iteration 1 execute thread 0.
Iteration 2 execute thread 0.
Iteration 7 execute thread 1.
Iteration 3 execute thread 0.
Iteration 8 execute thread 1.
Iteration 9 execute thread 1.
Iteration 4 execute thread 0.
Parallel normal way 6031 ms
```

nowait 4 wątki:

```
Iteration 6 execute thread 2.
Iteration 8 execute thread 3.
Iteration 7 execute thread 2.
Iteration 9 execute thread 3.
Sections - Thread 0 working...
Iteration 0 execute thread 0.
Iteration 1 execute thread 0.
Iteration 2 execute thread 0.
Sections - Thread 1 working...
Iteration 3 execute thread 1.
Iteration 4 execute thread 1.
Iteration 5 execute thread 1.
Parallel normal way 5224 ms
```

Bez nowait 4 wątki:

```
Sections - Thread 0 working...
Sections - Thread 1 working...
Iteration 3 execute thread 1.
Iteration 0 execute thread 0.
Iteration 6 execute thread 2.
Iteration 8 execute thread 3.
Iteration 9 execute thread 3.
Iteration 4 execute thread 1.
Iteration 1 execute thread 0.
Iteration 7 execute thread 2.
Iteration 2 execute thread 0.
Iteration 5 execute thread 1.
Parallel normal way 5225 ms
```

nowait 8 wątków:

```
Iteration 4 execute thread 2.
Iteration 5 execute thread 3.
Iteration 6 execute thread 4.
Iteration 7 execute thread 5.
Iteration 8 execute thread 6.
Iteration 9 execute thread 7.
Sections - Thread 0 working...
Iteration 0 execute thread 0.
Iteration 1 execute thread 0.
Sections - Thread 1 working...
Iteration 2 execute thread 1.
Iteration 3 execute thread 1.
Parallel normal way 4821 ms
```

Bez nowait 8 wątków:

```
Sections - Thread 0 working...
Sections - Thread 1 working...
Iteration 8 execute thread 6.
Iteration 5 execute thread 3.
Iteration 6 execute thread 4.
Iteration 9 execute thread 7.
Iteration 4 execute thread 2.
Iteration 7 execute thread 5.
Iteration 2 execute thread 1.
Iteration 0 execute thread 0.
Iteration 3 execute thread 1.
Iteration 1 execute thread 0.
Parallel normal way 4828 ms
```

Bez użycia nowait wątki wykonują się w określonej uporządkowanej kolejności, zazwyczaj jeden wątek skończy się cały, zanim kolejny wątek zacznie się wykonywać.
W przypadku zastosowania klauzuli nowait, wątki wykonują się w sposób losowy.

6.2. Napisz program liczący długość wektora na czterech wątkach, używając sekcji.

```
int main()
{
    auto start = std::chrono::high_resolution_clock::now();
#pragma omp parallel num_threads(4) default(shared)
    {
#pragma omp sections
        {
#pragma omp section
            {
                wspolrzedne_A(-3, 8);
                wait(1000);
            }
#pragma omp section
            {
                wspolrzedne_B(5, 2);
                wait(1000);
            }
#pragma omp section
            {
                wspolrzedne_wektora(&wspolrzedne_A, &wspolrzedne_B, -3, 8, 5, 2);
                wait(2000);
            }
#pragma omp section
            {
                dlugosc_wektora(&wspolrzedne_A, &wspolrzedne_B, -3, 8, 5, 2);
                wait(2000);
            }
        }
    }
    auto end = std::chrono::high_resolution_clock::now();
    printf("Parallel normal way %llu ms\r\n",
        std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count());

    return 0;
```

```
double dlugosc_wektora(vector<double>(*A)(double, double), vector<double>(*B)(double, double), double x1, double y1, double x2, double y2)
{
    try {
        if (A(x1, y1).size() != 2 || B(x2, y2).size() != 2) throw 2;
        vector<double> ab;
        ab.push_back(A(x1, y1)[0] - B(x2, y2)[0]);
        ab.push_back(A(x1, y1)[1] - B(x2, y2)[1]);
        if (ab.size() != 2) throw 3;

        double dlugosc = sqrt(ab[0] * ab[0] + ab[1] * ab[1]);
        cout << endl << "dlugosc wektora =" << dlugosc << endl;
        return dlugosc;
    }
    catch (int x) {
        cout << "wyjatek " << x << " dlugosc wektora inna od 2";
    }
}
```

```
dlugosc wektora =10
Parallel normal way 2003 ms
```