

Akademia Nauk Stosowanych w Nowym Sączu Programowanie Współbieżne i Rozproszone			
Temat: PWIR_10			spr nr 10
Nazwisko i imię: Ciapała Tadeusz		Ocena sprawozdania	Zaliczenie:
Data wykonania ćwiczenia:	Grupa: P3		

1) KODY:

KOD DO ZADANIA PIERWSZEGO:

```
#include <stdio>
#include <stdint>
#include <stdlib>
#include <chrono>
#include <assert.h>
#include <windows.h>
#include <omp.h>

void DoSomethingFast() {
    Sleep(1000);
}

void DoSomethingLong() {
    Sleep(6000);
}

int main() {
    uint8_t id;

    auto start = std::chrono::high_resolution_clock::now();
    DoSomethingFast();
    auto end = std::chrono::high_resolution_clock::now();

    printf("Fast in time: %llu ms\r\n",
        std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count());

    start = std::chrono::high_resolution_clock::now();
    DoSomethingLong();
    end = std::chrono::high_resolution_clock::now();

    printf("Long in time: %llu ms\r\n",
        std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count());

    start = std::chrono::high_resolution_clock::now();
    #pragma omp parallel num_threads(2) private(id)
    {
        id = omp_get_thread_num();

        if (id % 2) {
            DoSomethingLong();
        }
        else {
            DoSomethingFast();
        }

        printf("Thread %d done work and wait on barrier\n", id);
    }
    #pragma omp barrier

    printf("Thread %d done work and already finished\n", id);
}
```

```

        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;
    }

```

KOD DO ZADANIA DRUGIEGO:

```

#include <stdio>
#include <stdint>
#include <stdlib>
#include <chrono>
#include <assert.h>
#include <windows.h>
#include <omp.h>

void wait(int x) {
    Sleep(x);
}

int main() {
    int32_t i;
    uint32_t n = 10;
    auto start = std::chrono::high_resolution_clock::now();
    #pragma omp parallel num_threads(2) default(shared) private(i)
    {
        #pragma omp sections nowait
        {
            #pragma omp section
            {
                wait(2000);
                printf("Sections - Thread %d working...\n",
                    omp_get_thread_num());
            }

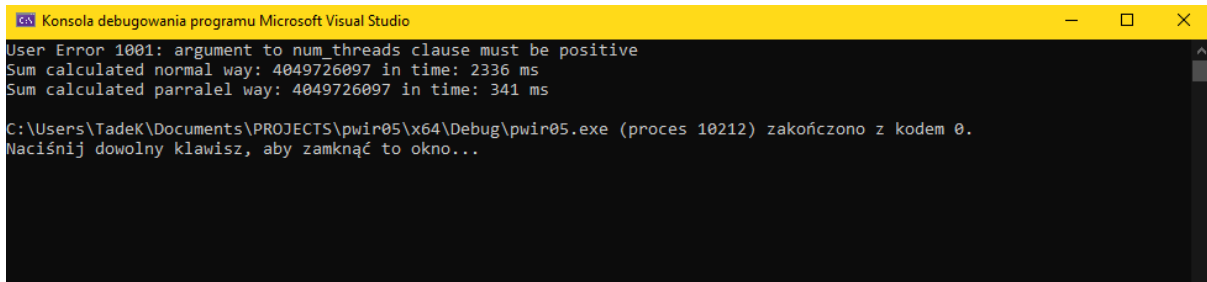
            #pragma omp section
            {
                wait(4000);
                printf("Sections - Thread %d working...\n",
                    omp_get_thread_num());
            }
        }
        #pragma omp for
        for (i = 0; i < n; i++) {
            printf("Iteration %d execute thread %d.\n", i, omp_get_thread_num());
            wait(400);
        }

        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;
    }

```

2) Zadania i rozwiązania:

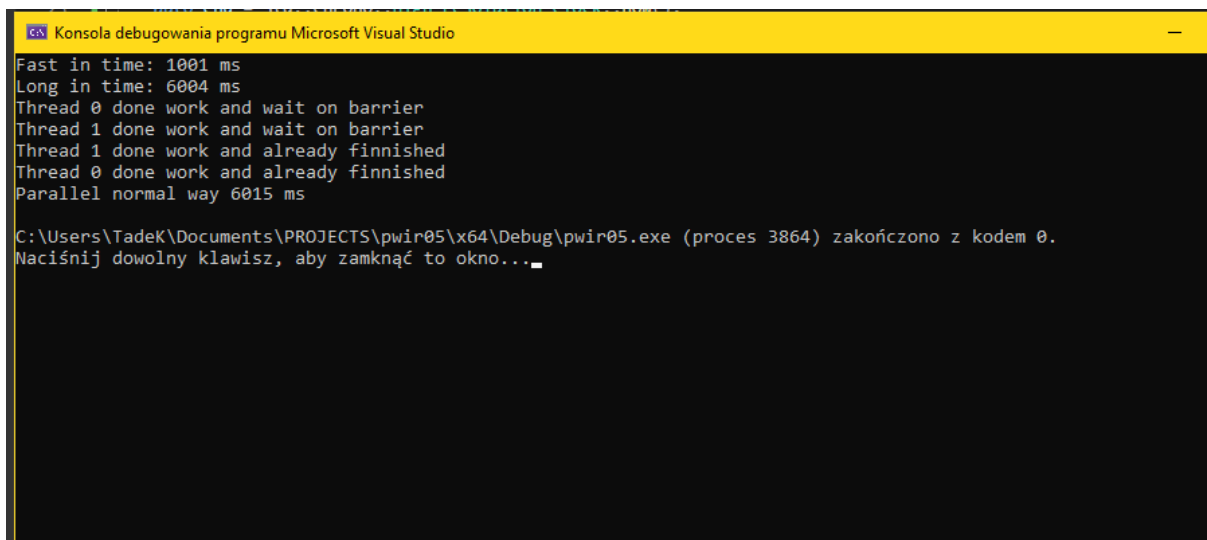


```
Konsola debugowania programu Microsoft Visual Studio
User Error 1001: argument to num_threads clause must be positive
Sum calculated normal way: 4049726097 in time: 2336 ms
Sum calculated parralel way: 4049726097 in time: 341 ms

C:\Users\TadeK\Documents\PROJECTS\pwir05\x64\Debug\pwir05.exe (proces 10212) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Testy - Kod 05_00

Pierwsze zadanie polegało na dopisaniu dyrektywy `num_threads` do programu oraz na przetestowaniu czasu wykonania.



```
Konsola debugowania programu Microsoft Visual Studio
Fast in time: 1001 ms
Long in time: 6004 ms
Thread 0 done work and wait on barrier
Thread 1 done work and wait on barrier
Thread 1 done work and already finnnished
Thread 0 done work and already finnnished
Parallel normal way 6015 ms

C:\Users\TadeK\Documents\PROJECTS\pwir05\x64\Debug\pwir05.exe (proces 3864) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

```
Konsola debugowania programu Microsoft Visual Studio
Fast in time: 1009 ms
Long in time: 6011 ms
Thread 4 done work and wait on barrier
Thread 10 done work and wait on barrier
Thread 8 done work and wait on barrier
Thread 0 done work and wait on barrier
Thread 2 done work and wait on barrier
Thread 6 done work and wait on barrier
Thread 3 done work and wait on barrier
Thread 1 done work and wait on barrier
Thread 5 done work and wait on barrier
Thread 9 done work and wait on barrier
Thread 11 done work and wait on barrier
Thread 7 done work and wait on barrier
Thread 3 done work and already finished
Thread 5 done work and already finished
Thread 1 done work and already finished
Thread 9 done work and already finished
Thread 11 done work and already finished
Thread 8 done work and already finished
Thread 10 done work and already finished
Thread 4 done work and already finished
Thread 6 done work and already finished
Thread 0 done work and already finished
Thread 7 done work and already finished
Thread 2 done work and already finished
Parallel normal way 6014 ms

C:\Users\TadeK\Documents\PROJECTS\pwir05\x64\Debug\pwir05.exe (proces 15632) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Wykorzystanie wielu wątków może znacznie przyspieszyć czas wykonywania programów, szczególnie w przypadku programów, które wykonują wiele skomplikowanych operacji.

Zadanie drugie polegało na przetestowaniu działania kodu 06_01 z klauzulą `nowait` oraz `bez`.

Z KLAUZULĄ `nowait`:

```
Konsola debugowania programu Microsoft Visual Studio
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 6015 ms

C:\Users\TadeK\Documents\PROJECTS\pwir05\x64\Debug\pwir05.exe (proces 2676) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

BEZ KLAUZULI:

```
WybierzKonsola debugowania programu Microsoft Visual Studio
Sections - Thread 0 working...
Sections - Thread 1 working...
Iteration 5 execute thread 1.
Iteration 0 execute thread 0.
Iteration 1 execute thread 0.
Iteration 6 execute thread 1.
Iteration 7 execute thread 1.
Iteration 2 execute thread 0.
Iteration 3 execute thread 0.
Iteration 8 execute thread 1.
Iteration 4 execute thread 0.
Iteration 9 execute thread 1.
Parallel normal way 6044 ms

C:\Users\Tadek\Documents\PROJECTS\pwir05\x64\Debug\pwir05.exe (proces 17888) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Wyniki jednoznacznie dowodzą, że zastosowanie klauzuli "nowait" skutecznie redukuje czas wykonania programu, szczególnie w przypadku, gdy wątki nie muszą oczekiwać na siebie nawzajem. Ponadto, wyniki potwierdzają, że większa liczba wątków może znacznie skrócić czas wykonania programu.

Zadanie trzecie polegało na napisaniu programu liczącego długość wektora na czterech wątkach, używając sekcji.

```
#include <cstdio>
#include <cstdlib>
#include <stdlib.h>
#include <chrono>
#include <assert.h>
#include <windows.h>
#include <omp.h>
#define VECT 10000

uint16_t* v;
uint32_t sumVector() {
    uint32_t sum = 0;
    for (uint32_t i = 0; i < VECT; i++) {
        sum += v[i];
    }
    return sum;
}

uint32_t sumVectorParallel() {
    uint32_t sum = 0;
    int32_t i;
    #pragma omp parallel for shared(v) private(i) reduction(+: sum)
    for (i = 0; i < VECT; i++) {
        sum = sum + v[i];
    }
    return sum;
}

void createVector() {
    v = (uint16_t*)new uint16_t[VECT];
    for (uint32_t i = 0; i < VECT; i++) {
        v[i] = (uint16_t)(rand() % 10);
    }
}
```

```

}

void wait(int x) {
    Sleep(x);
}

int main() {
    int32_t i;
    uint32_t n = 10;
    createVector();
    uint32_t sum = 0;
    auto start = std::chrono::high_resolution_clock::now();
#pragma omp parallel num_threads(4) default(shared) private(i)
    {
#pragma omp sections nowait
    {
#pragma omp section
    {
        wait(2000);
        printf("Sections - Thread %d working...\n", omp_get_thread_num());
    }
}
#pragma omp for
    for (i = 0; i < VECT; i++) {
        sum = sum + v[i];
    }
}

    auto end = std::chrono::high_resolution_clock::now();
    printf("Vector sum=%llu , in %llu ms\r\n", sum,
        std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count());

    return 0;
}

```

```

WybierzKonsola debugowania programu Microsoft Visual Studio
Sections - Thread 0 working...
Vector sum=44949 , in 2014 ms

C:\Users\TadeK\Documents\PROJECTS\pwir05\x64\Debug\pwir05.exe (proces 17220) zakończono z kodem 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...

```

Sections dzieli obliczenia na sekcje, po jednej dla wątku. W tym przypadku na cztery. Każda sekcja oblicza sumę odpowiednich elementów wektora “v”, a następnie dodaje tę sumę do zmiennej sum, która jest współdzielona między wątkami.