

Państwowa Wyższa Szkoła Zawodowa w Nowym Sączu			
Temat: PWIR02			
Nazwisko i imię: Wąchała Paweł		Ocena sprawozdania	Zaliczenie:
Data wykonania ćwiczenia: 12.04.2022		Grupa: P3	

Zadania do KOD1:

1. Uruchom program wiele razy i porównaj wyjście

```

Konsola debugowania programu Microsoft Visual Studio
Uruchamiam watek 2
Uruchamiam watek 1
Uruchamiam watek 3
Koncze watek 2
Koncze watek 3
Koncze watek 1
Koniec programu

C:\Users\student\Desktop\PW\05_04_zad1\Debug\05_04_zad1.exe (proces 14328) zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie
nie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...

Konsola debugowania programu Microsoft Visual Studio
Uruchamiam watek 1
Uruchamiam watek 2
Uruchamiam watek 3
Koncze watek 3
Koncze watek 1
Koncze watek 2
Koniec programu

Konsola debugowania programu Microsoft Visual Studio
Uruchamiam watek 1
Uruchamiam watek 2
Uruchamiam watek 3
Koncze watek 2
Koncze watek 1
Koncze watek 3
Koniec programu

```

Uruchamiane oraz kończone są wątki po 10 sekundach w różnej kolejności.

2. Usuń wszystkie 3 joiny, skompiluj i sprawdź wyjście

```

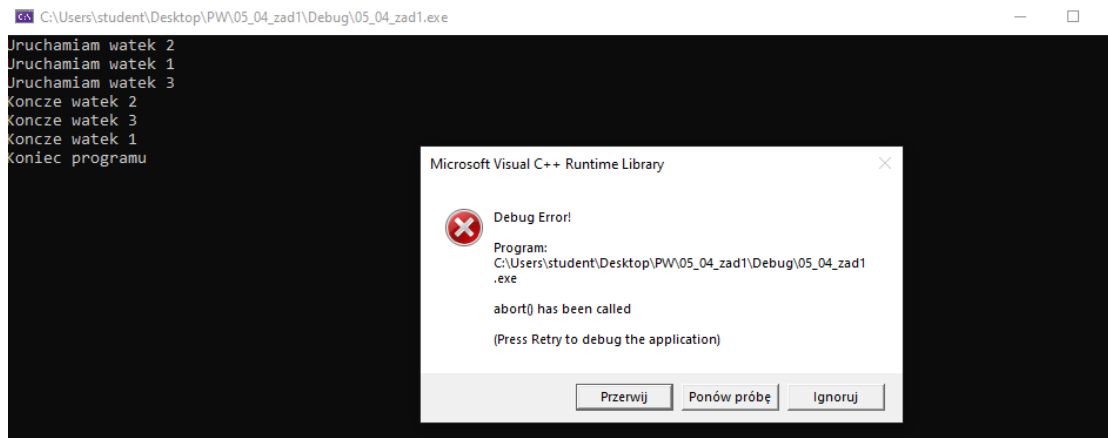
Koniec programu
Uruchamiam watek 1
Uruchamiam watek 2
Uruchamiam watek 3

C:\Users\student\Desktop\PW\05_04_zad1\Debug\05_04_zad1.exe (proces 8452) zakończono z kodem 3.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie -> Automatycznie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...

```

Program nie poczeka na wykonanie 10 sekundowego sleepa.

3. Usuń tylko jeden join, skompiluj i sprawdź co zostanie wyświetlone (najlepiej więcej niż raz).



Otrzymamy błąd „abort() has been called”

4. Zmodyfikuj program tak by wypisywane było id uzyskane z funkcji `get_id()`

```
5  int action(int id) {
6      std::thread::id id_watku = std::this_thread::get_id();
7      printf("Uruchamiam watek %d\n", id_watku);
8      Sleep(10 * 1000); //10 sekund
9      printf("Koncze watek %d\n", id_watku);
10     return 0;
11 }
12
13 int main() {
14     //tworzenie watku
15     std::thread t1(action, 1); //konstruktor klasy t1 przyjmuje minimum wskaźnik na funkcję do wykonania
16     std::thread t2(action, 2); //funkcja ta może coś zwracać, ale może zwracać też void
17     std::thread t3(action, 3); //dalsze parametry zostaną przekazane do podanej funkcji
18
19     t1.join(); //synchronizacja
20     t2.join(); //wątek główny ma tu poczekać na te 3 wątki
21     t3.join(); //inaczej program by się zakończył wcześniej bo wątki trwają minimum 10 sekund
22
23     printf("Koniec programu \r\n");
24
25     return 0;
26 }
```

5. Zmodyfikuj funkcję `action` w taki sposób by czas uśpienia dostawał jako parametr, następnie przetestuj zmiany. Sprawdź: - w jakiej kolejności zostaną zamknięte wątki względem podanych czasów.

```
1  #include <cstdio>
2  #include <thread>
3  #include <windows.h>
4
5  int action(int id, int czas_uspienia) {
6      printf("Uruchamiam watek %d\n", id);
7      Sleep(czas_uspienia * 1000); //10 sekund
8      printf("Koncze watek %d\n", id);
9      return 0;
10 }
11
12 int main() {
13     //tworzenie watku
14     std::thread t1(action, 1, 5); //konstruktor klasy t1 przyjmuje minimum wskaźnik na funkcję do wykonania
15     std::thread t2(action, 2, 5); //funkcja ta może coś zwracać, ale może zwracać też void
16     std::thread t3(action, 3, 5); //dalsze parametry zostaną przekazane do podanej funkcji
17
18     t1.join(); //synchronizacja
19     t2.join(); //wątek główny ma tu poczekać na te 3 wątki
20     t3.join(); //inaczej program by się zakończył wcześniej bo wątki trwają minimum 10 sekund
21
22     printf("Koniec programu \r\n");
23
24     return 0;
25 }
```

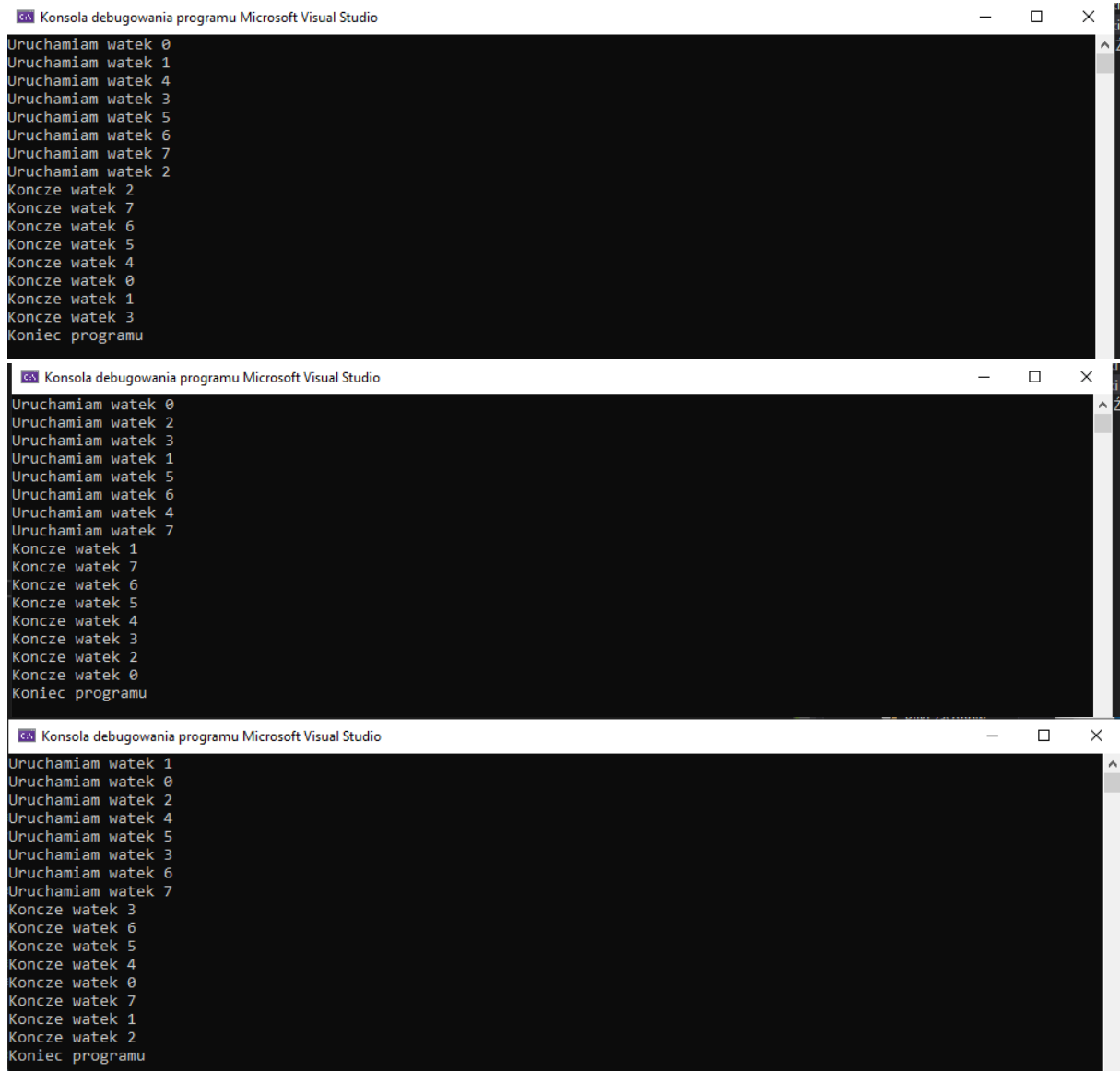
Konsola debugowania programu Microsoft Visual Studio

```
Uruchamiam watek 2
Uruchamiam watek 1
Uruchamiam watek 3
Koncze watek 3
Koncze watek 1
Koncze watek 2
Koniec programu
```

Zadania do KOD2

1. Uruchom program wiele razy i porównaj wyjście.

Tak samo jak w zadaniu 1 uruchamiane są wątki i kończone po ustawionym czasie w różnej kolejności.



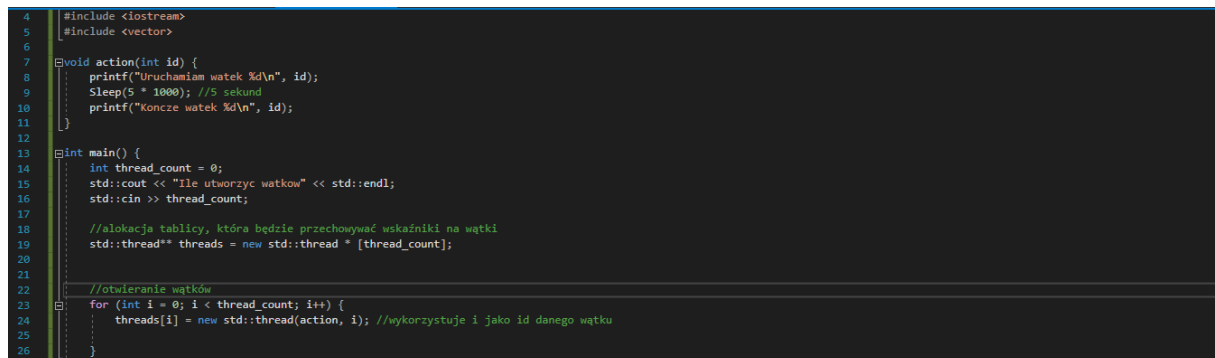
The image shows three screenshots of the Visual Studio debug console, each displaying the output of a program that launches 8 threads (0-7) and prints their start and end times. The output shows that the threads complete their execution in a non-deterministic order across the three runs.

```
Konsola debugowania programu Microsoft Visual Studio
Uruchamiam watek 0
Uruchamiam watek 1
Uruchamiam watek 4
Uruchamiam watek 3
Uruchamiam watek 5
Uruchamiam watek 6
Uruchamiam watek 7
Uruchamiam watek 2
Koncze watek 2
Koncze watek 7
Koncze watek 6
Koncze watek 5
Koncze watek 4
Koncze watek 0
Koncze watek 1
Koncze watek 3
Koniec programu

Konsola debugowania programu Microsoft Visual Studio
Uruchamiam watek 0
Uruchamiam watek 2
Uruchamiam watek 3
Uruchamiam watek 1
Uruchamiam watek 5
Uruchamiam watek 6
Uruchamiam watek 4
Uruchamiam watek 7
Koncze watek 1
Koncze watek 7
Koncze watek 6
Koncze watek 5
Koncze watek 4
Koncze watek 3
Koncze watek 2
Koncze watek 0
Koniec programu

Konsola debugowania programu Microsoft Visual Studio
Uruchamiam watek 1
Uruchamiam watek 0
Uruchamiam watek 2
Uruchamiam watek 4
Uruchamiam watek 5
Uruchamiam watek 3
Uruchamiam watek 6
Uruchamiam watek 7
Koncze watek 3
Koncze watek 6
Koncze watek 5
Koncze watek 4
Koncze watek 0
Koncze watek 7
Koncze watek 1
Koncze watek 2
Koniec programu
```

2. Zmodyfikuj program tak by ilość wątków była pobierana od użytkownika.



```
4 #include <iostream>
5 #include <vector>
6
7 void action(int id) {
8     printf("Uruchamiam watek %d\n", id);
9     Sleep(5 * 1000); //5 sekund
10    printf("Koncze watek %d\n", id);
11 }
12
13 int main() {
14     int thread_count = 0;
15     std::cout << "Ile utworzyc watkow" << std::endl;
16     std::cin >> thread_count;
17
18     //alokacja tablicy, która będzie przechowywać wskaźniki na wątki
19     std::thread* threads = new std::thread * [thread_count];
20
21
22     //otwieranie wątków
23     for (int i = 0; i < thread_count; i++) {
24         threads[i] = new std::thread(action, i); //wykorzystuje i jako id danego wątku
25     }
26 }
```

```
Konsola debugowania programu Microsoft Visual Studio

Ile utworzyc watkow
5
Uruchamiam watek 4
Uruchamiam watek 0
Uruchamiam watek 2
Uruchamiam watek 3
Uruchamiam watek 1
Koncze watek 1
Koncze watek 3
Koncze watek 2
Koncze watek 0
Koncze watek 4
Koniec programu
```

3. Zmodyfikuj program tak by zamiast tablicy przechowującej wskaźniki na wątki użyć wektora przechowującego wskaźniki na wątki.

```
regulaminu obozow      pliczek      <C++>      Konsola debugowania programu Microsoft Visual Studio

10      printf("Koncze watek %d\n", id);
11      }
12      }
13      int main() {
14          int thread_count = 0;
15          std::cout << "Ile utworzyc watkow" << std::endl;
16          std::cin >> thread_count;
17          //alokacja tablicy, która będzie przechowywać wskaźniki na wątki
18          //std::thread** threads = new std::thread * [thread_count];
19          std::vector<std::thread> threads;
20          std::vector<std::thread> threads;
21          //otwieranie wątków
22          for (int i = 0; i < thread_count; i++) {
23              //threads[i] = new std::thread(action, i); //wykorzystuje i jako id danego wątku
24              std::thread watek = new std::thread(action, i);
25              threads.push_back(watek);
26          }
27          //wątki pracują, ale trzeba je zsynchronizować
28          for (int i = 0; i < thread_count; i++) {
29              threads[i].join();
30          }
31          //alokowaliśmy pamięć więc musimy ją zwolnić
32          for (int i = 0; i < thread_count; i++) {
33              delete threads[i];
34          }
35      }
36      }
37      }
38      }
```

```
Ile utworzyc watkow
5
Uruchamiam watek 3
Uruchamiam watek 1
Uruchamiam watek 2
Uruchamiam watek 4
Uruchamiam watek 0
Koncze watek 0
Koncze watek 2
Koncze watek 4
Koncze watek 1
Koncze watek 3
Koniec programu

H:\PW_rozprowadzone\05_04_zad1\Debug\05_04_zad1.exe (proces 11240) zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie -> Auto-
znie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Zadania do KOD3:

1.Sprawdź ile trwają na Twoim komputerze operacje: otwarcia i zamknięcia pliku

```
1      #include <chrono>
2      #include <cstdio>
3      #include <windows.h>
4      #include <fstream>
5      int main() {
6          std::fstream fs;
7          //otwarcie
8          auto start = std::chrono::steady_clock::now();
9          //długie operacje
10         //Sleep(2000);
11         fs.open("pliczek.txt", std::fstream::in | std::fstream::out);
12
13         auto end = std::chrono::steady_clock::now();
14         //zamknięcie pliku
15         auto start2 = std::chrono::steady_clock::now();
16         fs.close();
17
18         auto end2 = std::chrono::steady_clock::now();
19
20         printf("Czas trwania otwarcia pliku: %llu\n", std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count());
21         printf("Czas trwania zamknięcia pliku: %llu\n", std::chrono::duration_cast<std::chrono::milliseconds>(end2 - start2).count());
22
23         return 0;
24     }
```

```
Konsola debugowania programu Microsoft Visual Studio

Czas trwania otwarcia pliku: 9
Czas trwania zamknięcia pliku: 0

C:\Users\student\Desktop\PW\05_04_zad1\Debug\05_04_zad1.exe (proces 1624) zakończono z kodem 0.
Aby automatycznie zamknąć konsolę po zatrzymaniu debugowania, włącz opcję Narzędzia -> Opcje -> Debugowanie -> Automatyc
znie zamknij konsolę po zatrzymaniu debugowania.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

2. Sprawdź ile trwa na Twoim komputerze wygenerowanie jakiejś większej (np. 40) ilości elementów ciągu fibonacciego.

```
6 int main() {
7     long long a = 0, b = 1;
8
9     auto start = std::chrono::steady_clock::now();
10    for (int i = 0; i < 40; i++)
11    {
12        std::cout << b << " ";
13        b += a;
14        a = b - a;
15    }
16
17    auto end = std::chrono::steady_clock::now();
18
19    printf("Czas trwania: %llu\n", std::chrono::duration_cast<std::chrono::milliseconds>(end - start).count());
20
21    return 0;
22 }
23
24 }
```

Konsola debugowania programu Microsoft Visual Studio

```
1 1 1 2 3 5 8 13 21 34 55 89 144 233 377 610 987 1597 2584 4181 6765 10946 17711 28657 46368 75025 121393 196418 317811 514229 832040 1346269 2178309 3524578 5702887
9227465 14930352 24157817 39088169 63245986 102334155 Czas trwania: 18
```