

| | | | |
|--|--|-----------------------|-------------|
| Akademia Nauk Stosowanych w Nowym Sączu Programowanie Współbieżne i Rozproszone | | | |
| Temat: PWIR_02 | | | |
| Nazwisko i imię: Ciapała Tadeusz | | Ocena sprawozdania | Zaliczenie: |
| | | | |
| Data wykonania ćwiczenia: 28.02.2023 | | Grupa: P3 | |

1. KOD 1:

Biblioteka `std::thread` w C++

Wyżej wymieniona biblioteka, jest jedną z najprostszych metod pracy wielowątkowej. Opiera się na podejściu stworzenia wątku, wykonującego daną funkcję z danymi parametrami.

Link do dokumentacji:

<https://en.cppreference.com/w/cpp/thread/thread>

```
#include <cstdio>
#include <thread>
#include <windows.h>

int action(int id) {
    printf("Uruchamiam watek %d\n", id);
    Sleep(10 * 1000); //10 sekund
    printf("Koncze watek %d\n", id);
    return 0;
}

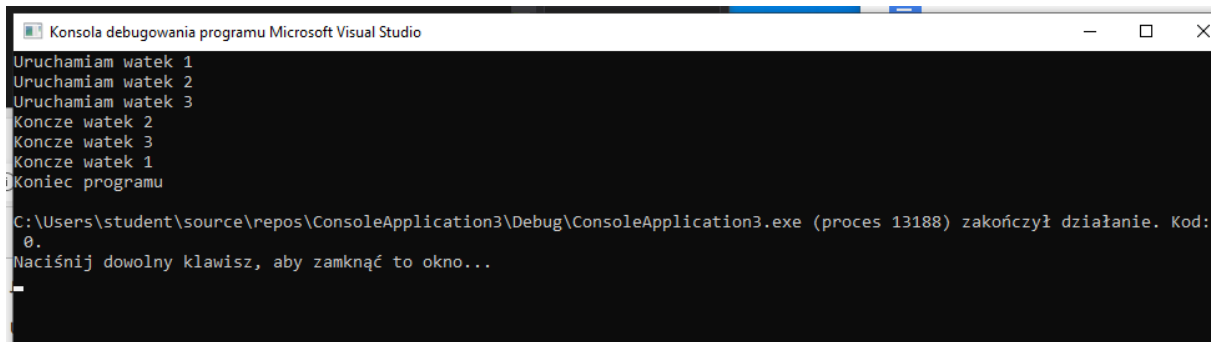
int main() {
    //tworzenie wątku
    std::thread t1(action, 1); //konstruktor klasy t1 przyjmuje minimum wskaźnik na funkcję do wykonania
    std::thread t2(action, 2); //funkcja ta może coś zwracać, ale może zwracać też void
    std::thread t3(action, 3); //dalsze parametry zostaną przekazane do podanej funkcji

    t1.join(); //synchronizacja
    t2.join(); //wątek główny ma tu poczekać na te 3 wątki
    t3.join(); //inaczej program by się zakończył wcześniej bo wątki trwają minimum 10 sekund

    printf("Koniec programu \n\n");

    return 0;
}
```

Listing nr 1: dostarczony kod w materiałach referencyjnych - KOD 1



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

C:\Users\student\source\repos\ConsoleApplication3\Debug\ConsoleApplication3.exe (proces 13188) zakończył działanie. Kod: 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Zrzut 1 - KOD 1 Zad 1: uruchomienie dostarczonego kodu

Zastosowałem zmianę w funkcji action(int), zmieniona została funkcja Sleep na Sleep(1000)

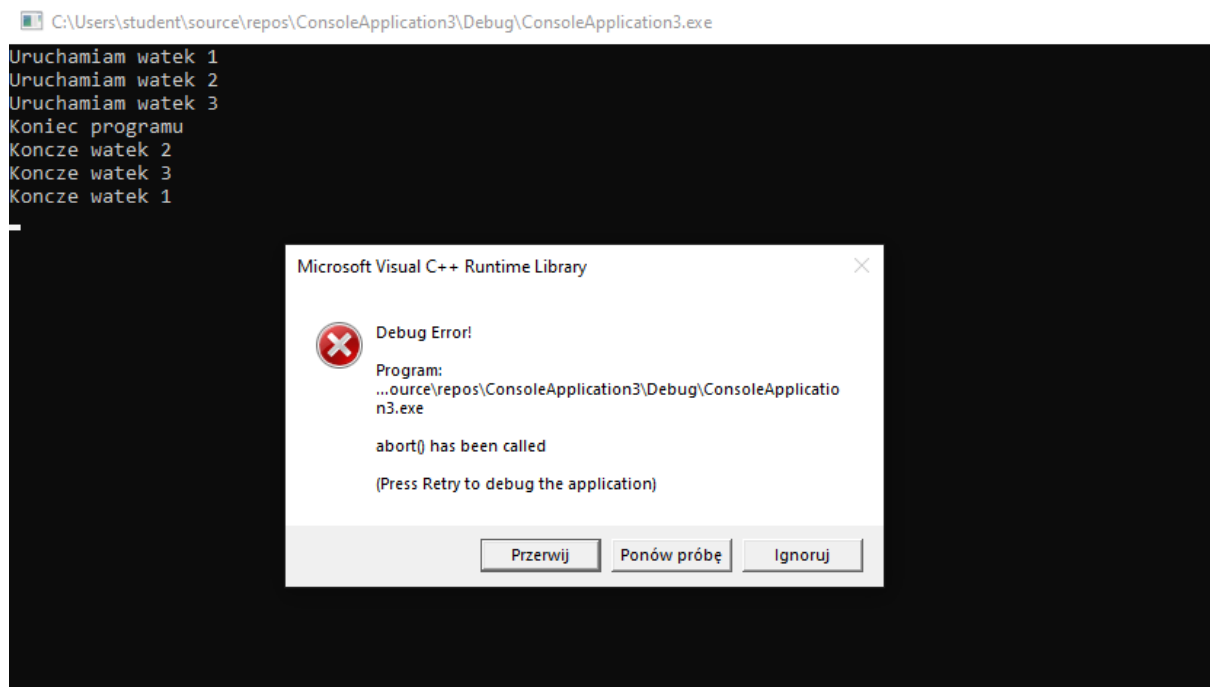


```
Konsola debugowania programu Microsoft Visual Studio
Uruchamiam watek 1
Uruchamiam watek 2
Uruchamiam watek 3
Koncze watek 3
Koncze watek 1
Koncze watek 2
Koniec programu

C:\Users\student\source\repos\ConsoleApplication3\Debug\ConsoleApplication3.exe (proces 11448) zakończył działanie. Kod: 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

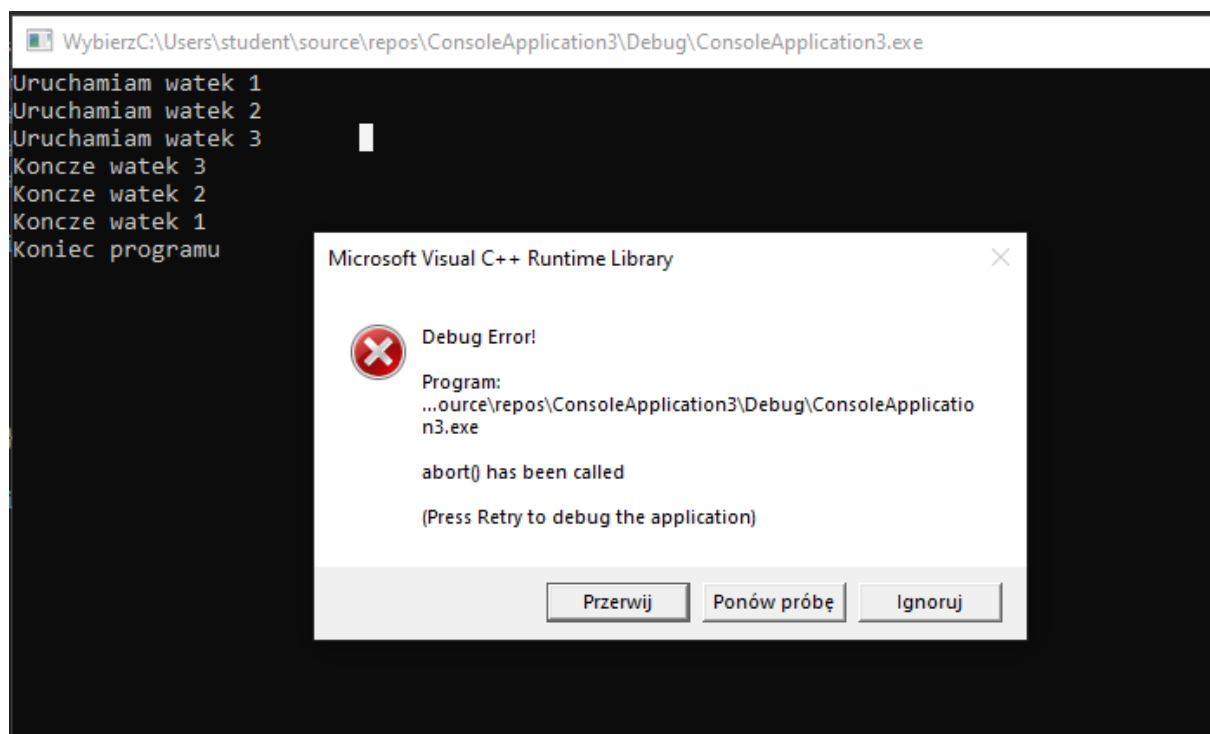
Zrzut 2 - KOD 1 Zad 1: program po kilkukrotnym uruchomieniu - warto zauważyć, że zmieniła się kolejność kończenia wątków.

usunięto z głównej funkcji metody join()



Zrzut 3 - KOD 1 Zad 2: uruchomienie programu bez metod join()

- teraz usuwam tylko jeden join()



Zrzut 4 - KOD 1 Zad 3: uruchomienie po usunięciu jednej metody join()

```

#include <stdio>
#include <thread>
#include <windows.h>
#include <iostream>

int action(int id) {
    printf("Uruchamiam watek %d\n", id);
    Sleep(1000); //
    printf("Koncze watek %d\n", id);
    return 0;
}

int main() {
    //tworzenie wątku
    std::thread t1(action, 1); //konstruktor klasy t1 przyjmuje minimum wskaźnik na funkcję do wykonania
    std::thread t2(action, 2); //funkcja ta może coś zwracać, ale może zwracać też void
    std::thread t3(action, 3); //dalsze parametry zostaną przekazane do podanej funkcji

    std::thread::id t1_id = t1.get_id();
    std::thread::id t2_id = t2.get_id();
    std::thread::id t3_id = t3.get_id();

    std::cout << "Id watek nr 1: " << t1_id << "\n";
    std::cout << "Id watek nr 2: " << t2_id << "\n";
    std::cout << "Id watek nr 3: " << t3_id << "\n";

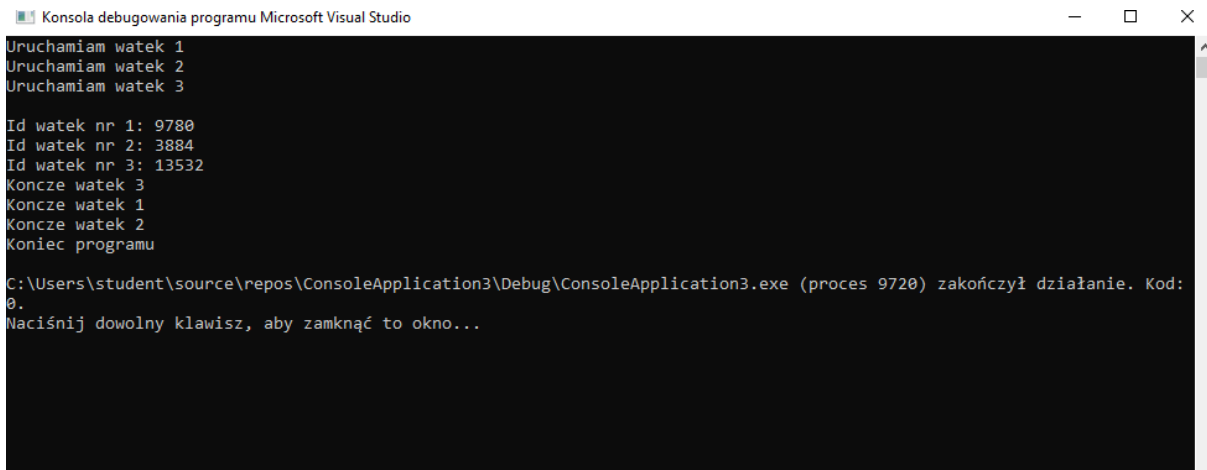
    t1.join(); //synchronizacja
    t2.join(); //wątek główny ma tu poczekać na te 3 wątki
    t3.join(); //inaczej program by się zakończył wcześniej bo wątki trwają minimum 10 sekund

    printf("Koniec programu \r\n");

    return 0;
}

```

Listing nr 2: zmodyfikowany kod do zadania 4



```

Konsola debugowania programu Microsoft Visual Studio

Uruchamiam watek 1
Uruchamiam watek 2
Uruchamiam watek 3

Id watek nr 1: 9780
Id watek nr 2: 3884
Id watek nr 3: 13532
Koncze watek 3
Koncze watek 1
Koncze watek 2
Koniec programu

C:\Users\student\source\repos\ConsoleApplication3\Debug\ConsoleApplication3.exe (proces 9720) zakończył działanie. Kod: 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...

```

Zrzut 5 - KOD 1 Zad 4: wyświetlanie id wątków

```
Konsola debugowania programu Microsoft Visual Studio

Uruchamiam watek 1
Uruchamiam watek 2
Uruchamiam watek 3

Id watek nr 1: 7204
Id watek nr 2: 9600
Id watek nr 3: 13732
Koncze watek 3
Koncze watek 1
Koncze watek 2
Koniec programu

C:\Users\student\source\repos\ConsoleApplication3\Debug\ConsoleApplication3.exe (proces 8800) zakończył działanie. Kod:
0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Zrzut 6 - KOD 1 Zad 4: ponowne uruchomienie - id się zmienia

```
#include <cstdio>
#include <thread>
#include <windows.h>
#include <iostream>

int action(int id, int s) {
    printf("Uruchamiam watek %d\n", id);
    Sleep(s*1000);
    printf("> Koncze watek %d , uplynelo %d sekund\n", id, s);
    return 0;
}

int main() {
    //tworzenie wątku
    std::thread t1(action, 1, 1); //konstruktor klasy t1 przyjmuje minimum wskaźnik na funkcję do
    wykonania
    std::thread t2(action, 2, 5); //funkcja ta może coś zwracać, ale może zwracać też void
    std::thread t3(action, 3, 10); //dalsze parametry zostaną przekazane do podanej funkcji

    std::thread::id t1_id = t1.get_id();
    std::thread::id t2_id = t2.get_id();
    std::thread::id t3_id = t3.get_id();

    std::cout << "\nId watek nr 1: " << t1_id << "\n";
    std::cout << "Id watek nr 2: " << t2_id << "\n";
    std::cout << "Id watek nr 3: " << t3_id << "\n";

    t1.join(); //synchronizacja
    t2.join(); //wątek główny ma tu poczekać na te 3 wątki
    t3.join(); //inaczej program by się zakończył wcześniej bo wątki trwają minimum 10 sekund

    printf("Koniec programu \n\n");

    return 0;
}
```

Listing nr 3: kod po zmianach z zadania nr 5

```
Konsola debugowania programu Microsoft Visual Studio

Uruchamiam watek 1
Uruchamiam watek 2
Uruchamiam watek 3

Id watek nr 1: 4236
Id watek nr 2: 5380
Id watek nr 3: 13088
-> Koncze watek 1 , uplynelo 1 sekund
-> Koncze watek 2 , uplynelo 5 sekund
-> Koncze watek 3 , uplynelo 10 sekund
Koniec programu

C:\Users\student\source\repos\ConsoleApplication3\Debug\ConsoleApplication3.exe (proces 6196) zakończył działanie. Kod:
0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Zrzut 7 - KOD 1 Zad 5: action przyjmuje ilość sekund jako parametr

```
Konsola debugowania programu Microsoft Visual Studio

Uruchamiam watek 1
Uruchamiam watek 2
Uruchamiam watek 3

Id watek nr 1: 9104
Id watek nr 2: 9804
Id watek nr 3: 2860
-> Koncze watek 2 , uplynelo 1 sekund
-> Koncze watek 3 , uplynelo 3 sekund
-> Koncze watek 1 , uplynelo 10 sekund
Koniec programu

C:\Users\student\source\repos\ConsoleApplication3\Debug\ConsoleApplication3.exe (proces 2940) zakończył działanie. Kod:
0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Zrzut 8 - KOD 1 zad 5: kolejne uruchomienie z innymi czasami trwania wątków - inna kolejnosc zakończenia

2. KOD 2:

Otwieranie bardzo dużej ilości wątków:

```
#include <stdio>
#include <thread>
#include <windows.h>

void action(int id){
    printf("Uruchamiam watek %d\n", id);
    Sleep(5*1000); //5 sekund
    printf("Koncze watek %d\n", id);
}

int main(){
    int thread_count = 8;

    //alokacja tablicy, która będzie przechowywać wskaźniki na wątki
    std::thread** threads = new std::thread*[thread_count];

    //otwieranie wątków
    for(int i = 0; i < thread_count; i++){
        threads[i] = new std::thread(action, i); //wykorzystuje i jako id danego wątku
    }

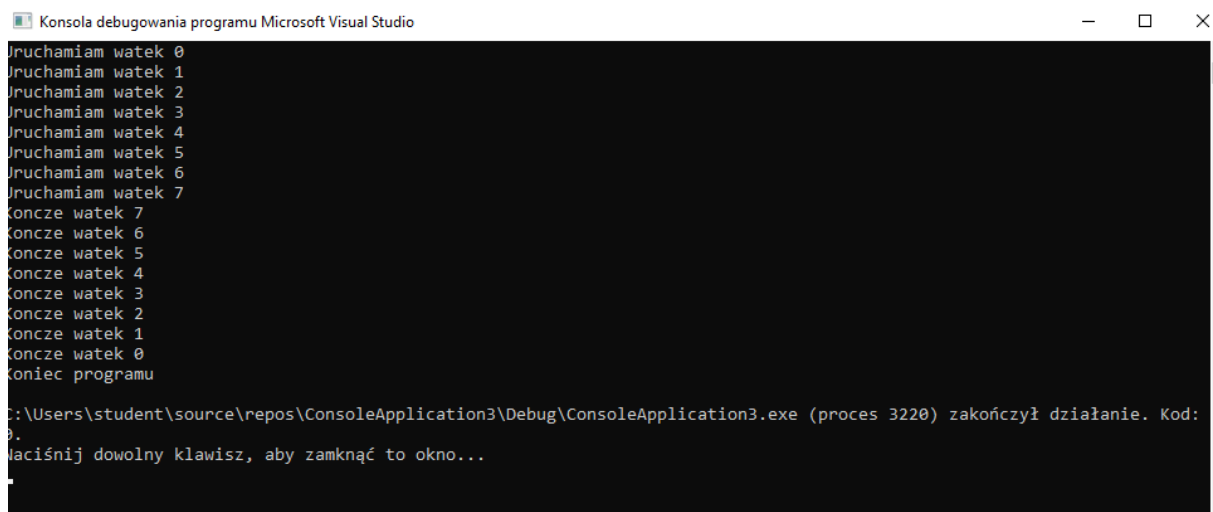
    //wątki pracują, ale trzeba je zsynchronizować
    for(int i = 0; i < thread_count; i++){
        threads[i] -> join();
    }

    //alokowaliśmy pamięć więc pasuje ją zwolnić
    for(int i = 0; i < thread_count; i++){
        delete threads[i];
    }
    delete[] threads;

    printf("Koniec programu \r\n");

    return 0;
}
```

Listing nr 4: dostarczony kod w materiałach referencyjnych - KOD 1



```
Konsola debugowania programu Microsoft Visual Studio

Uruchamiam watek 0
Uruchamiam watek 1
Uruchamiam watek 2
Uruchamiam watek 3
Uruchamiam watek 4
Uruchamiam watek 5
Uruchamiam watek 6
Uruchamiam watek 7
Koncze watek 7
Koncze watek 6
Koncze watek 5
Koncze watek 4
Koncze watek 3
Koncze watek 2
Koncze watek 1
Koncze watek 0
Koniec programu

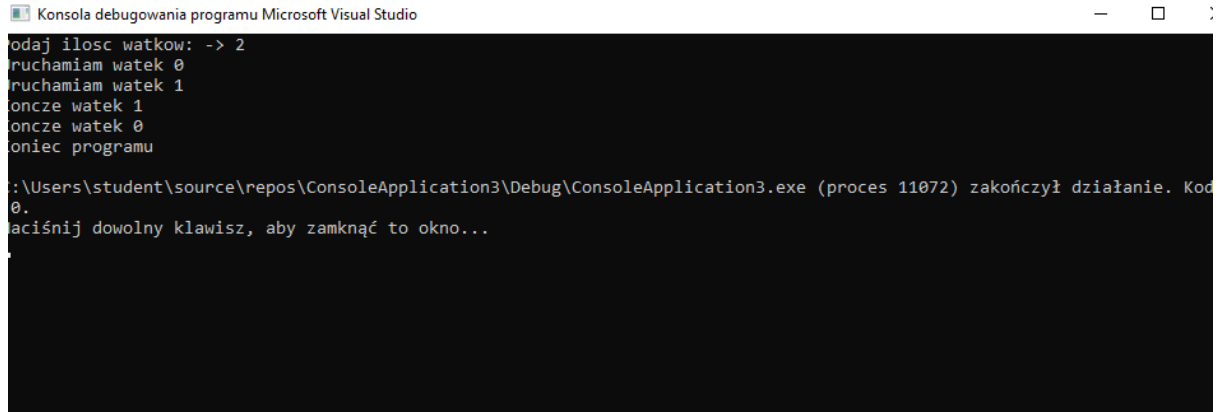
C:\Users\student\source\repos\ConsoleApplication3\Debug\ConsoleApplication3.exe (proces 3220) zakończył działanie. Kod: 0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Zrzut 9 - KOD 2 zad 1: pierwsze uruchomienie danego programu z zadania

zmodyfikowano fragment kodu aby pobierał ilość wątków do uruchomienia od użytkownika.
Na początku funkcji main() dodano kod z listingu poniżej:

```
int thread_count = 0;
std::cout << "Podaj ilosc watkow: -> ";
std::cin >> thread_count;
```

Listing nr 5: modyfikacje w funkcji main()



Konsola debugowania programu Microsoft Visual Studio

```
Podaj ilosc watkow: -> 2
Uruchamiam watek 0
Uruchamiam watek 1
Koncze watek 1
Koncze watek 0
Koniec programu

C:\Users\student\source\repos\ConsoleApplication3\Debug\ConsoleApplication3.exe (proces 11072) zakończył działanie. Kod
0.
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Zrzut 10 - KOD 2 zad 2: podano 2 watki do stworzenia

Zmodyfikowano program tak, aby zamiast tablicy użyć vector do przechowania wskaźników na wątki.

kod programu

```
#include <stdio>
#include <thread>
#include <windows.h>
#include <iostream>
#include <vector>

void action(int id) {
    printf("Uruchamiam watek %d\n", id);
    Sleep(1000); //1 sekund
    printf("Koncze watek %d\n", id);
}

int main() {
    int thread_count = 0;

    std::cout << "Podaj ilosc watkow: -> ";
    std::cin >> thread_count;

    std::vector<std::thread*> threads(thread_count);
    //std::thread** threads = new std::thread*[thread_count];

    //otwieranie wątków
    for (int i = 0; i < thread_count; i++)
    {
        threads.at(i) = new std::thread(action, i);
    }

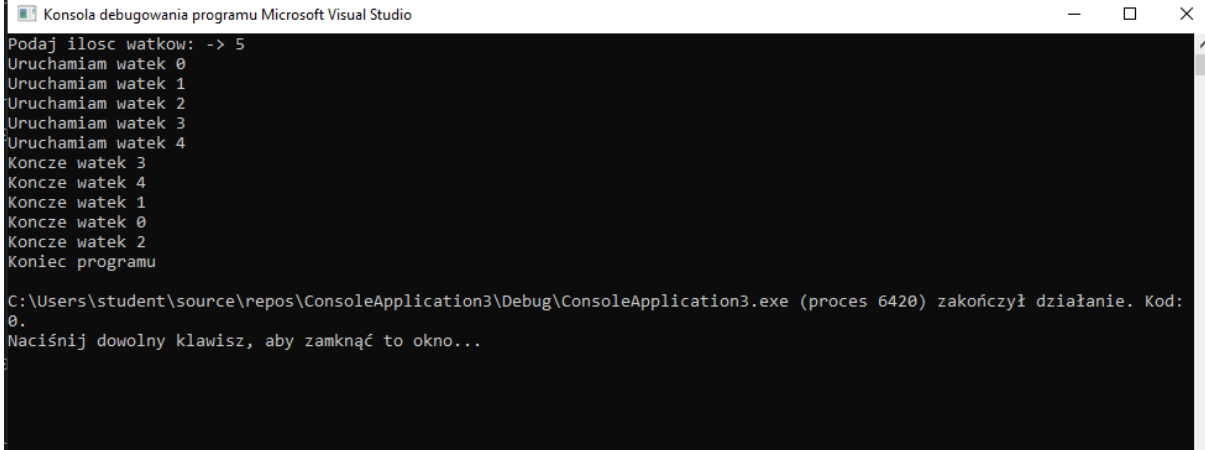
    //watkı pracują, ale trzeba je zsynchronizować
    for (int i = 0; i < thread_count; i++)
    {
        threads.at(i)->join();
    }

    //alokowaliśmy pamięć więc pasuje ją zwolnić
    for (int i = 0; i < thread_count; i++) {
```



```
        delete threads.at(i);  
    }  
    threads.clear();  
  
    printf("Koniec programu \r\n");  
  
    return 0;  
}
```

Listing nr 6: Kod z vector'ami<>



Konsola debugowania programu Microsoft Visual Studio

```
Podaj ilosc watkow: -> 5  
Uruchamiam watek 0  
Uruchamiam watek 1  
Uruchamiam watek 2  
Uruchamiam watek 3  
Uruchamiam watek 4  
Koncze watek 3  
Koncze watek 4  
Koncze watek 1  
Koncze watek 0  
Koncze watek 2  
Koniec programu  
  
C:\Users\student\source\repos\ConsoleApplication3\Debug\ConsoleApplication3.exe (proces 6420) zakończył działanie. Kod:  
0.  
Naciśnij dowolny klawisz, aby zamknąć to okno...
```

Zrzut 11 - KOD 2 zad 3: wykorzystanie kolekcji Vector