|  |  |
| --- | --- |
| Akademia Nauk Stosowanych w Nowym Sączu  Wydział Nauk Inżynieryjnych | |
| Imię i nazwisko: | Michał Bernardy |
| Grupa: | P1 |
| Ocena: |  |

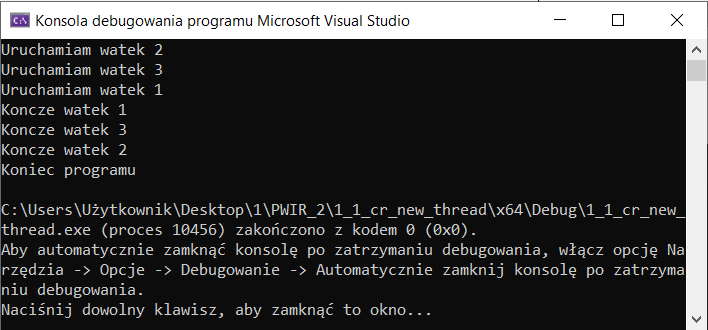
**Wstęp**

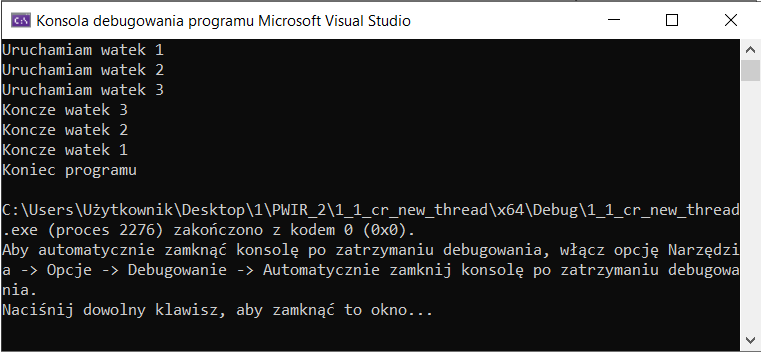
Sprawozdanie dotyczy analizy oraz implementacji programów wielowątkowych w środowisku C++. Celem zadań było zrozumienie zasad działania wątków, ich synchronizacji oraz zarządzania czasem uśpienia. W ramach eksperymentów przeprowadzono szereg modyfikacji kodu, mających na celu zbadanie wpływu różnych podejść do tworzenia i obsługi wątków.

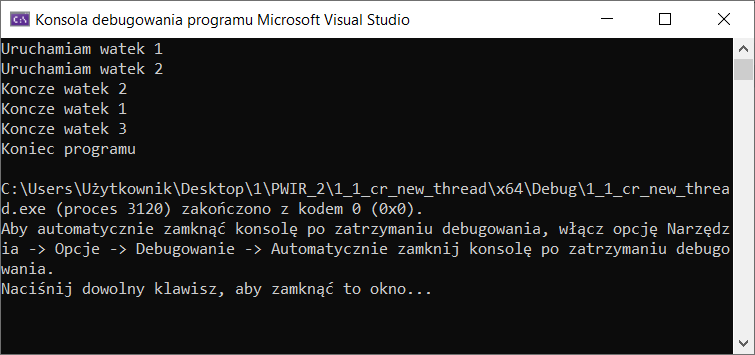
**Opis problematyki poszczególnych zadań**

**KOD1**

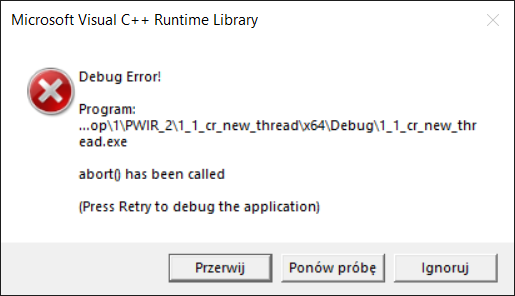
1. W pierwszym zadaniu zauważono, że kompilacja kodu wielokrotnie skutkuje różny rezultatami, ponieważ kolejność zakończenia wątków jest losowa. Wynika to z mechanizmu przydziału czasu procesora przez system operacyjny.



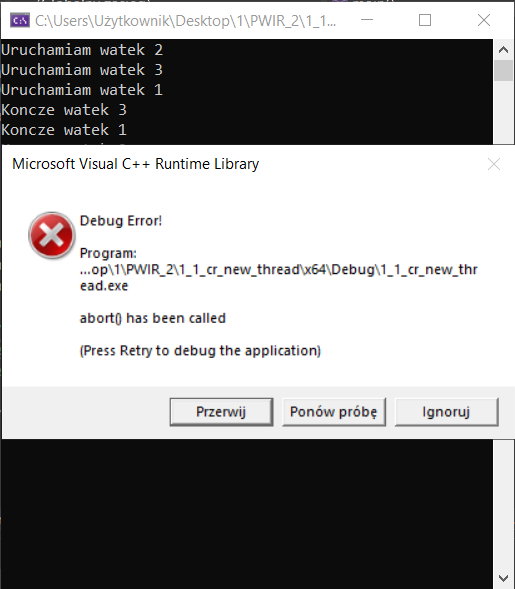




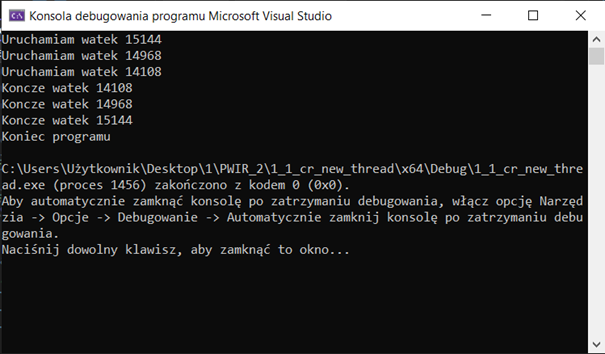
1. Usunięcie wywołań join skutkuje tym, że wątek główny może zakończyć działanie przed wątkami pobocznymi, co powoduje błędy w działaniu programu.



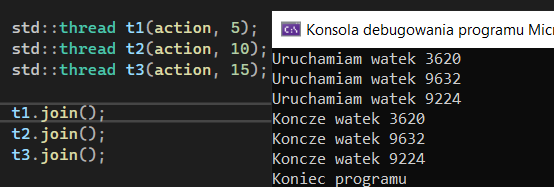
1. Usunięcie jednego z wywołań join skutkuje tym, że możemy mieć tak że zostanie ukończony 0/1/2 wątki i dopiero dostaniemy błąd.



1. Następna modyfikacja polegała na wprowadzeniu funkcji get\_id() o, co pozwalało na pobieranie prawdziwych id procesów .

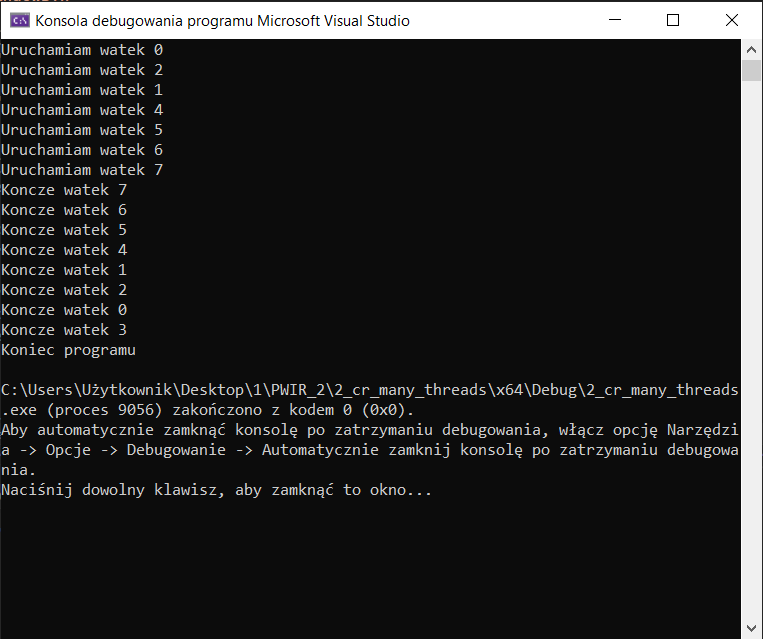
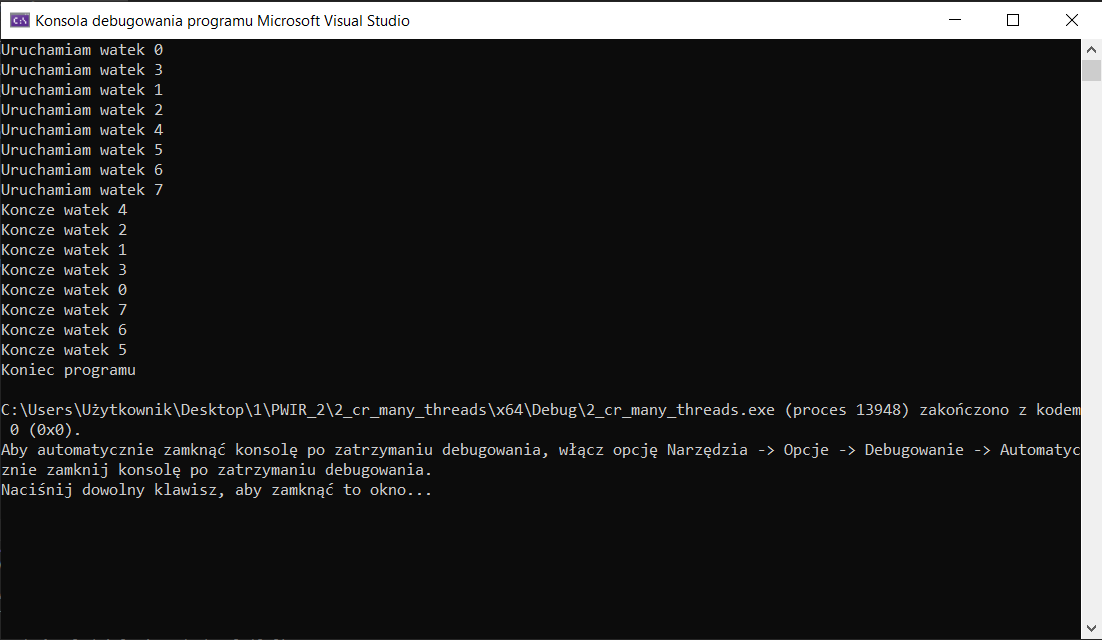
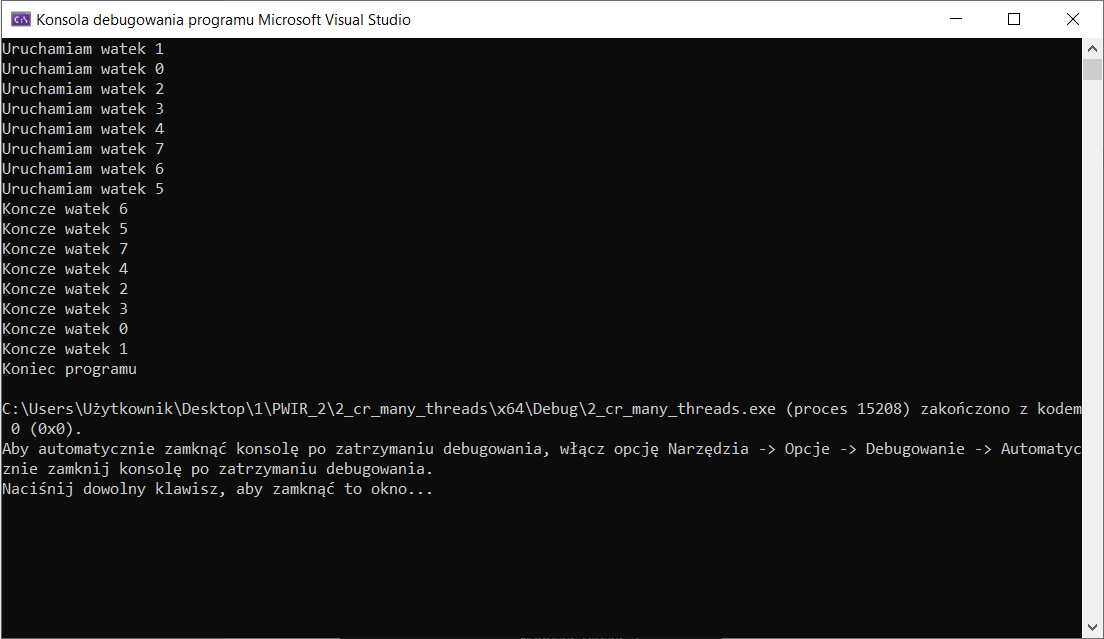
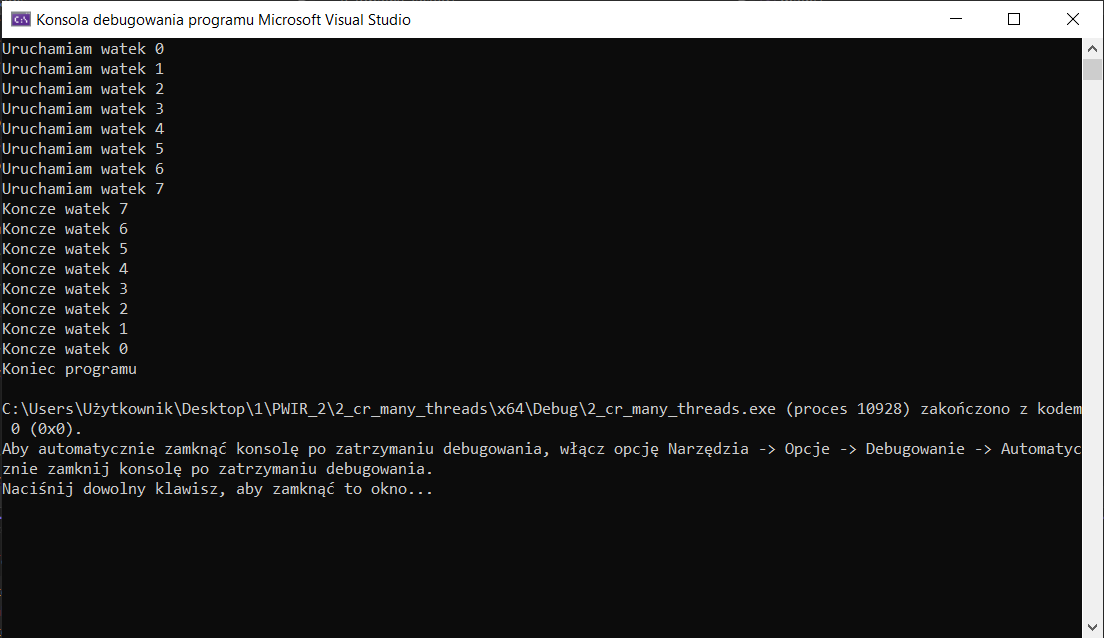


1. Ostatnia modyfikacja polegała na wprowadzeniu parametru sleep\_time, co pozwalało na określanie czasu uśpienia wątków.

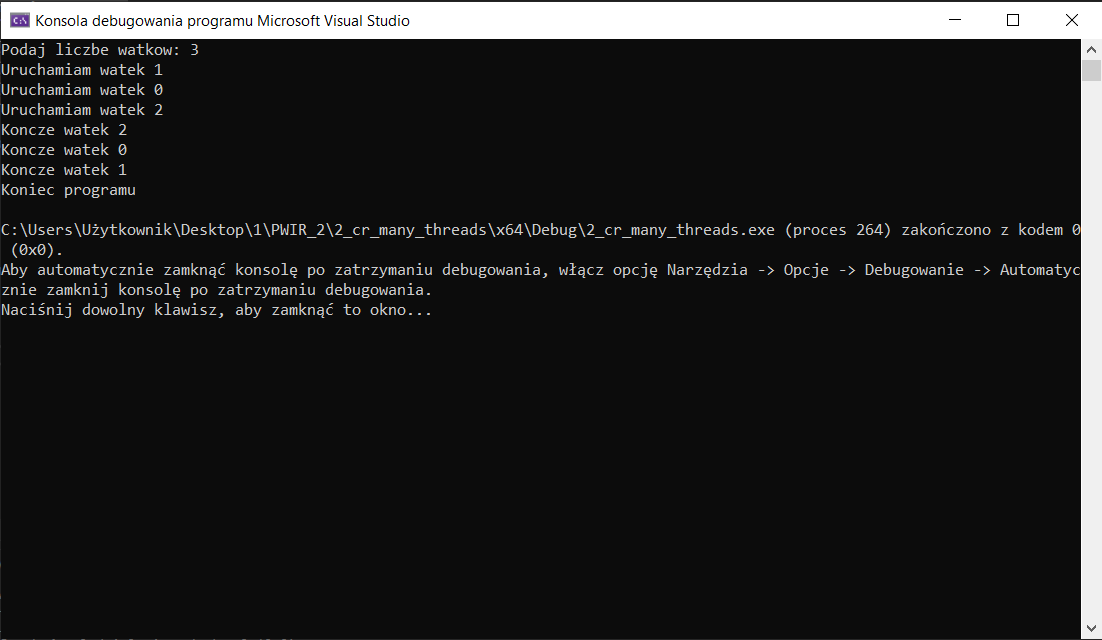
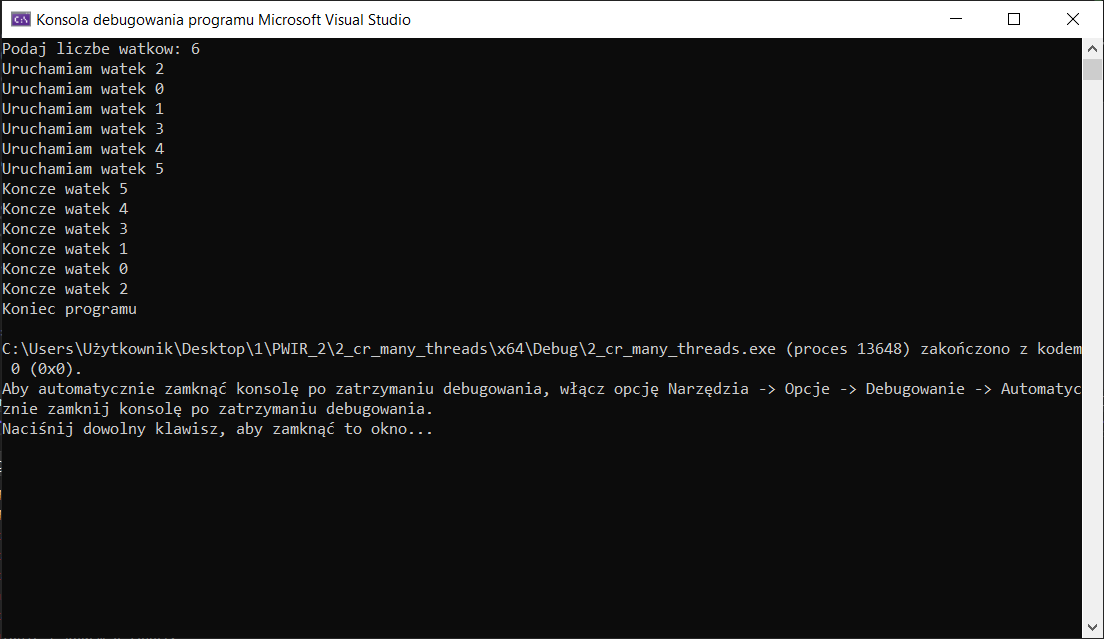
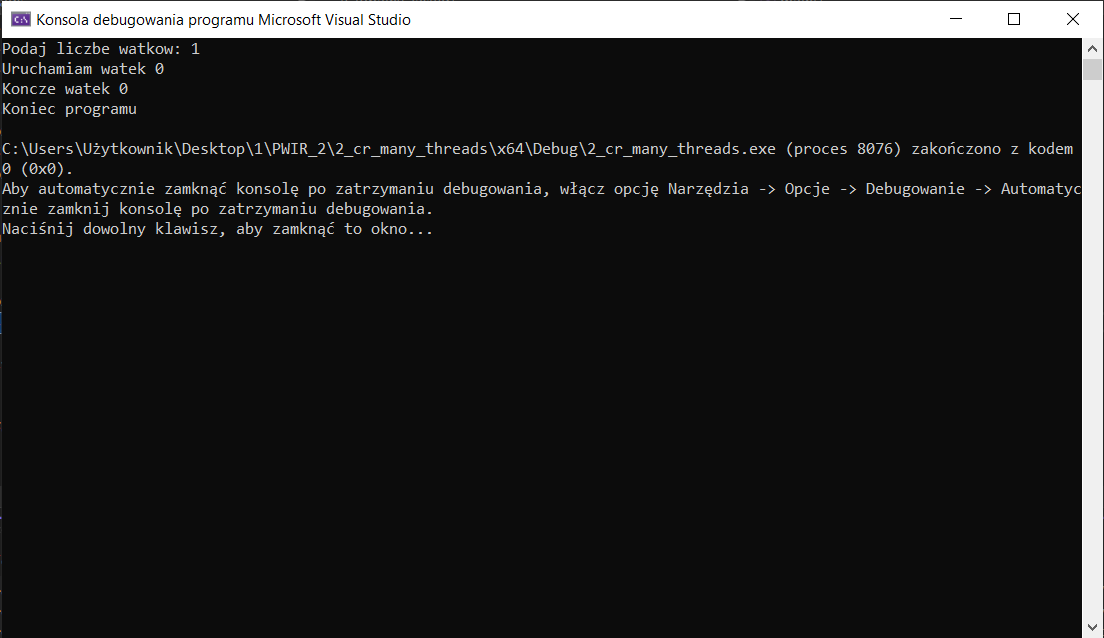


**KOD2**

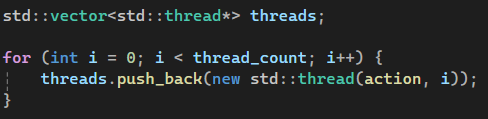
1. Drugie zadanie dotyczyło obsługi wielu wątków. Również w tym przypadku zaobserwowano, że kolejność ich wykonywania nie jest deterministyczna, co jest wynikiem mechanizmu zarządzania czasem procesora.

1. Zmodyfikowano kod tak, aby liczba wątków była pobierana od użytkownika, co pozwala na dynamiczne tworzenie wątków i lepsze zarządzanie pamięcią.

1. Kolejna modyfikacja polegała na zastosowaniu std::vector zamiast statycznej tablicy wskaźników, co umożliwiło lepsze zarządzanie pamięcią oraz dynamiczne dodawanie i usuwanie wątków.



**Wnioski**

Przeprowadzone eksperymenty potwierdziły, że zarządzanie wątkami w C++ wymaga uwzględnienia mechanizmu przydziału czasu procesora przez system operacyjny. Ponadto, synchronizacja wątków jest kluczowa dla uniknięcia błędów i nieprzewidywalnych wyników. Użycie dynamicznych struktur danych, takich jak std::vector, pozwala na efektywne zarządzanie wątkami oraz pamięcią.

**Podsumowanie**

Zadania przedstawione w sprawozdaniu pozwoliły na lepsze zrozumienie zasad pracy z wątkami w C++. Przeanalizowano losowość wyników, problematykę synchronizacji oraz efektywne zarządzanie pamięcią w programach wielowątkowych. Uzyskane rezultaty pokazały, jak istotne jest odpowiednie zarządzanie wątkami i ich synchronizacja dla poprawnego działania aplikacji.