Для реалізації мультипоточності було використано std::thread – інструментарій, що надається стандартною бібліотекою C++.

Використання бібліотечного класу std::thread дозволяє дуже просто створити і виконати додатковий потік. Для цього в якості аргументу достатньо передати назву функції, що буде виконуватись в новому потоці, та список аргументів цієї функції. Переваги: незалежність від OS.

В моїй програмі створюється два додаткові потоки: перший виконує швидке сортування над переданим масивом структур, другий виконує сортування виборкою над переданим масивом структур.

Обидва потоки працюють з двома різними масивами даних в пам’яті, тому додаткові механізми синхронізації не застосовуються.

Для очікування завершення обох потоків застосовано метод join класу std::thread. В програмі було реалізовано доволі просту схему паралельного обчислювання, тому для спрощення коду за основу було взято такий сценарій:

* основний потік створює два додаткових потоки;
* основний потік чекає завершення роботи цих двох потоків;
* після цього основний потік продовжує процес обробки (щойно отриманих) даних.

Використання потоків реалізує концепцію паралельного обчислення. Це дозволяє прискорити виконання програми шляхом задіяння всієї потужності обчислювальної системи. Перевагами такого підходу є зменшення часу виконання програми (або конкретно певного обчислювання), можливість виконувати різні підсистеми незалежно одна від одної (робота з мережею, інтерфейс користувача, тощо). Із недоліків можна назвати необхідність синхронізації виконання потоків (уникнення race condition - стану гонки, забезпечення цілісності об’єктів, тощо), що в свою чергу потенційно може привести до відомих помилок синхронізації, як то взаємне блокування (deadlock). Також при досягненні певної кількості потоків, що вже працюють, додавання нових тільки зменшить загальну продуктивність, і це напряму залежить від апаратних умов.