Przemysław Szymoniak

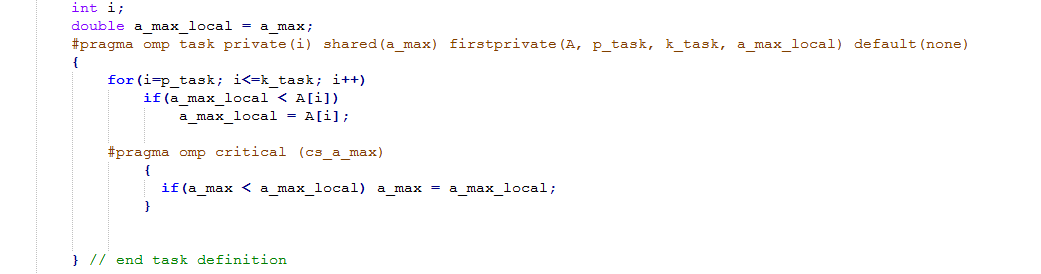
Przetwarzanie współbieżne. Programowanie równoległe i rozproszone.

Sprawozdanie z laboratorium 11.

Celem laboratorium było nabycie umiejętności pisania programów w środowisku OpenMP z wykorzystaniem puli wątków.

W ramach zajęć zrealizowałem następujące kroki:

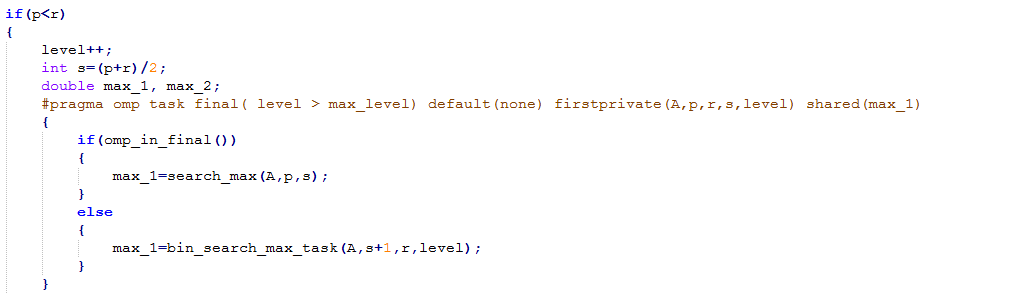
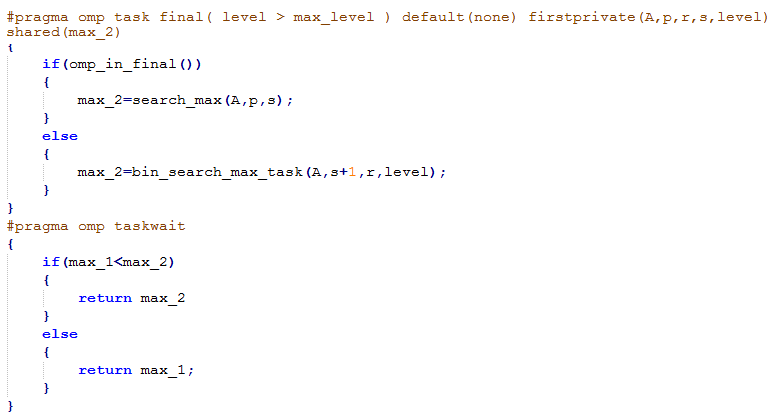
* Pobrałem i rozpakowałem wszystkie wymagane pliki, utworzyłem wymagane katalogi oraz uruchomiłem program.
* Dodanie definicji zadań dla wersji równoległej wyszukiwania liniowego:



* W funkcji „merge\_sort\_openmp” realizującej sortowanie przez scalanie zdefiniowano rozrost drzewa na 2 poziomy.

#pragma omp task final(poziom>max\_poziom) default(none) firstprivate(A,p,r,q1,poziom) – generujemy zadania do wykonania przez wątki w sposób asynchroniczny. Dopóki poziom mniejszy od max\_poziom kolejne zadania wykonywane sa od razu, default(none) to ustawienie, że domyślnie żadna zmienna nie jest prywatna ani wspólna, firstprivate zaś to zmienne inicjowane przy użyciu wartości zmiennych o nazwach jak w wątku głównym. #pragma omp taskwait wymusza oczekiwanie na zakończenie wykonywania wcześniej utworzonych zadań.

* Funkcja „merge\_sort\_openmp\_2” różni się od poprzedniej posiadaniem 3 poziomów rozrostu oraz wykorzystaniem funkcji sortowania szybkiego dla finałowego wykonania zadania ( sprawdzanie w warunku if(omp\_in\_final()) ) w innym wypadku wywołuje samą siebie dzieląc podzbiór na mniejsze części. Po skończeniu podziału algorytm zaczyna scalać tablice w posortowanej kolejności.
* Funkcja „merge\_sort\_openmp\_4” wykorzystuje zagnieżdżenia aktywowane przez omp\_set\_nested(1);   
  #pragma omp parallel sections default(none) firstprivate(A,p,r,q1) – jedyną zmianą w stosunku do poprzedników jest użycie dyrektywy sections służy ona do nie iteracyjnego podziału bloku pomiędzy grupę wątków. Każdy blok tworzymy przez użycie dyrektywy section. W taki właśnie sposób tworzymy 2 bloki a każdy z nich dzielimy później przy użyciu tych samych dyrektyw na 2 następne bloki.
* Algorytm sortowania binarnego z wykorzystaniem sortowania liniowego po przekroczeniu poziomu:



Wnioski:

* Dyrektywa task służy do generowania  
  zadań do wykonania przez wątki w   
  sposób asynchroniczny.
* Wykorzystanie hybrydowego sortowania (łączącego sortowanie binarne/merge i liniowe/szybkie) w znaczący sposób przyśpiesza prace algorytmu.
* Funkcja omp\_in\_final() zwraca true

Jeśli znajdujemy się w zadaniu które jako ostatnie spełnia warunek podany w dyrektywie final.