Пухов Максим Станиславович (ИС-242)

Отчёт

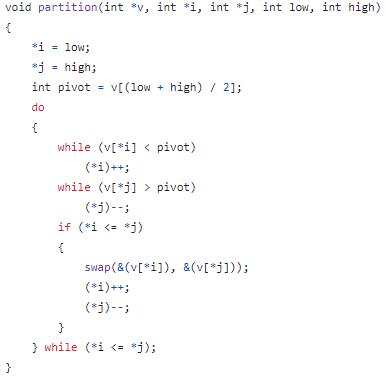
Лабораторная работа 5. Параллелизм задач

Задание

* На базе директив #pragma omp task реализовать многопоточный рекурсивный алгоритм быстрой сортировки (QuickSort). Опорным выбирать центральный элемент подмассива (функция partition, см. слайды к лекции). При достижении подмассивами размеров THREASHOLD = 1000 элементов переключаться на последовательную версию алгоритма.
* Выполнить анализ масштабируемости алгоритма для различного числа сортируемых элементов и порогового значения THRESHOLD.

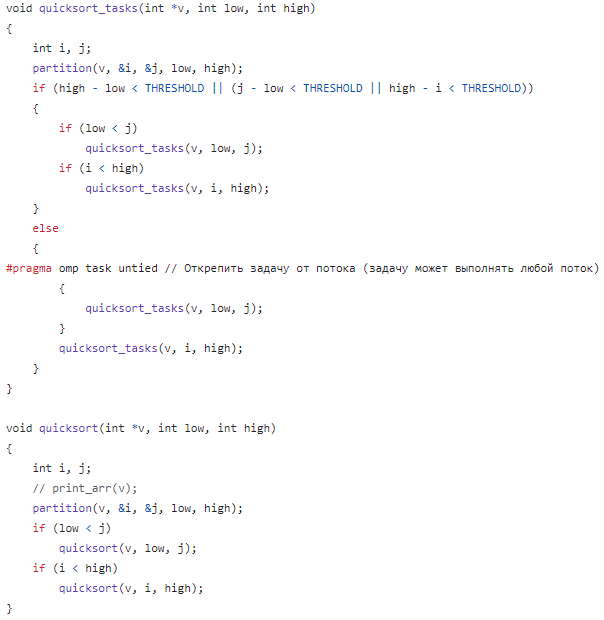
Описание работы программы

### Функция разбиения для быстрой сортировки



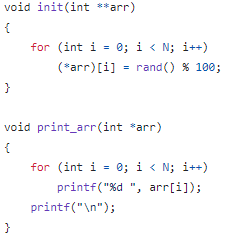
- \*\*`partition`\*\*: Выполняет разбиение массива на две части относительно опорного элемента (pivot).

### Параллельная и последовательная быстрая сортировка

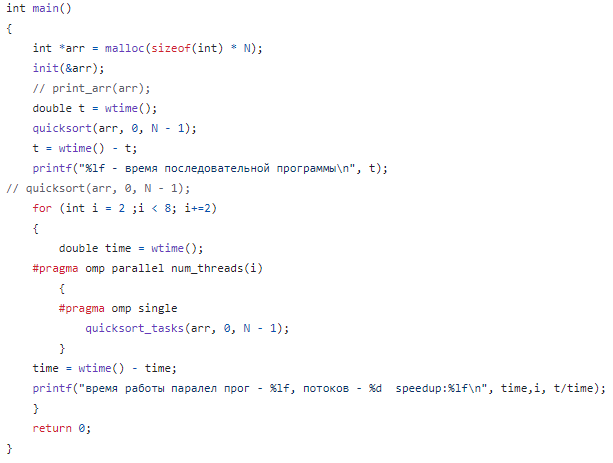


- \*\*`quicksort\_tasks`\*\*: Параллельная версия быстрой сортировки с использованием OpenMP задач.  
- \*\*`quicksort`\*\*: Последовательная версия быстрой сортировки.

### Инициализация и печать массива

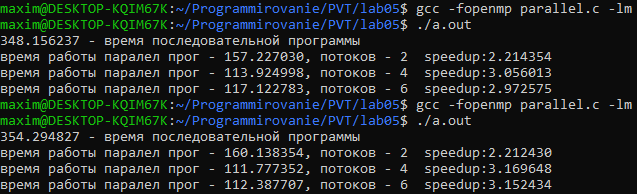


- \*\*`init`\*\*: Заполняет массив случайными числами от 0 до 99.  
- \*\*`print\_arr`\*\*: Печатает массив (не используется в основной программе, оставлено для отладки).  
  
### Основная функция

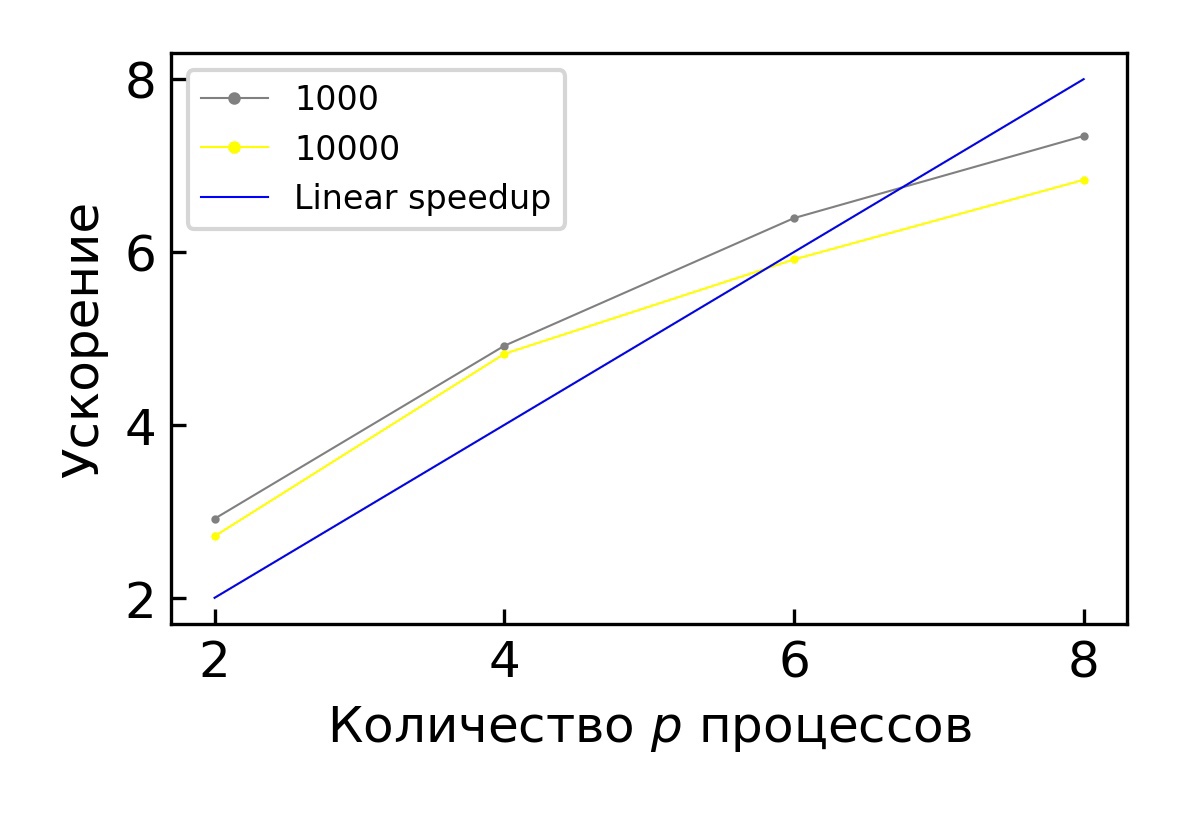


- \*\*`main`\*\*: Основная функция программы, где:  
- Выделяется память для массива.  
- Массив инициализируется случайными числами.  
- Выполняется последовательная быстрая сортировка и измеряется время выполнения.  
- Выполняется параллельная быстрая сортировка с различным количеством потоков (от 2 до 6) и измеряется время выполнения и ускорение.  
  
Ключевой момент заключается в использовании OpenMP для параллельного выполнения быстрой сортировки с задачами (`[#pragma](https://vk.com/im?sel=152867536&st=%23pragma) omp task`). Это позволяет эффективно использовать многопоточность для больших массивов, уменьшая время выполнения.

Работа программы



График



Характеристики процессора



