OTUS

C++ Developer. Professional

otus.ru

Защита проекта
Тема: Подход организации сортировки массивов строк с использованием многопоточности

Новикова Анастасия

Проблема

Разрабатывается desktop приложение, предназначенное для работы с иерархическими структурами данных. При разработке виджетов отображения данных столкнулись с проблемой медленной сортировки данных. Необходимо решить задачу ускорения сортировки данных.

Сортировка иерархической структуры данных заключается в последовательной сортировке множества линейных массивов элементов. Для каждого элемента задано строковое поле, которое отображается в интерфейсе и по которому происходит сравнение.

Для упрощения рассматриваемой задачи элементы были заменены на строки (так как сравнение элементов сводится к сравнению строк).

Цели проекта

1. Рассмотреть возможность использования многопоточности при сортировке.
2. Разработать подход деления массивов между потоками
3. Реализовать сортировку

Что планировалось

1. Реализовать пул потоков, в которых будет выполняться сортировка массивов
2. Для организации параллельных вычислений можно использовать парадигму «портфель задач». На первом этапе создается «портфель задач» – массив указателей на массивы, которые необходимо отсортировать.
3. Изучить алгоритмы сортировки большого числа элементов в одном массиве
4. Проверить работоспособность полученных алгоритмов

Что получилось

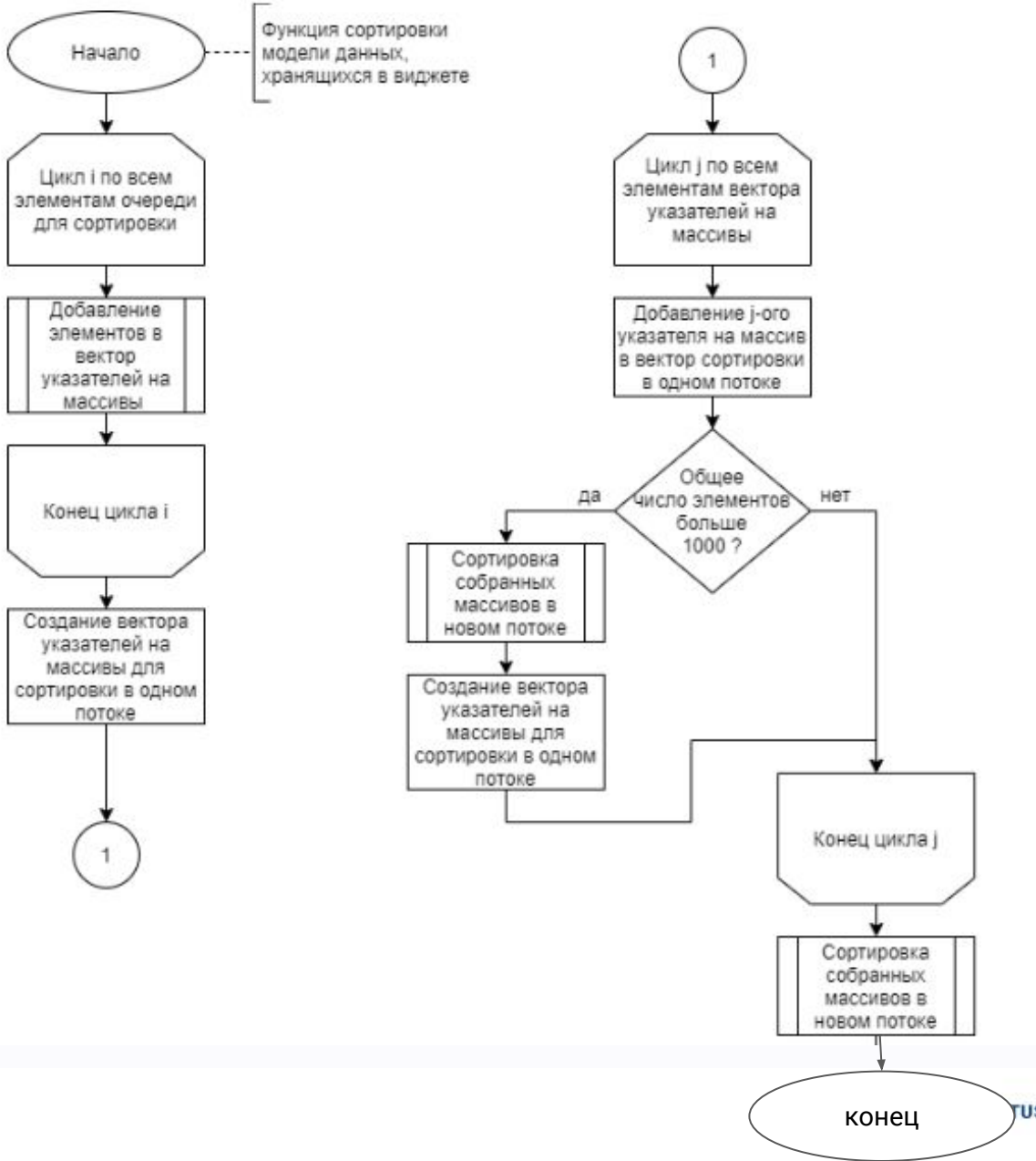
1. Изначально для организации многопоточных вычислений использовались классы QtConcurrent. Для исключения из сборки библиотек Qt организация многопоточности была заменена на класс std::thread.

2. Создается очередь из массивов элементов для сортировки. Если количество элементов для сортировки в очереди превышает минимальное значение, то будет создан поток для сортировки данных элементов.

3. Для сортировки массивов большого размера (больше 10000 элементов) был рассмотрен и реализован алгоритм Timsort – гибридный алгоритм сортировки, сочетающий сортировку вставками и сортировку слиянием.

4. При тестировании работы программы с многопоточностью были выявлены следующие закономерности:
   - При небольшом количестве массивов и количестве элементов для сортировки (до 300 элементов в массиве) создание отдельного потока занимает больше времени, чем последовательная сортировка
   - Если число массивов с количеством элементов до 300 превышает 1000, то имеет смысл запускать отдельные потоки.

Блок-схема алгоритма сортировки массивов строк



Что получилось

Собранная программа для linux:
https://github.com/anastasiyanovikova/otusCourseHW13/releases/tag/6

При запуске программы можно задать два парметра:
максимальное возможное число элементов в массиве: max_size
число создаваемых массивов для тестирования: N

В программе будет создано N массивов. Размер каждого массива будет задан случайным образом в интервале от 0 до max_size. Массивы будут заполнены случайными строками длиной 12 символов. Затем будет вызвана функция сортировки массивов и функция проверки отсортированных массивов.

Выводы и планы по развитию

1. Использование многопоточности позволяет ускорить сортировку массивов по сравнению с последовательными алгоритмами.
2. Для небольшого числа элементов затраты на создание потока превышают выигрыш от параллельных вычислений, поэтому использование многопоточности в этом случае не имеет смысла.
3. В планах на развитие создание пула потока, работа над потокобезопасностью с использованием класса (std::thread)
4. Реализация возможности задавать тип сортировки (по возрастанию, по убыванию). На данный момент реализована и протестирована сортировка по возрастанию (в timsort реализована сортировка только по возрастанию)

Спасибо за внимание!