**Отчет**

Результаты выполнения программы:

Алгоритм Эратосфена:

Mean result: число опытов - 10 число элементов - 10000, количество простых - 1229, время выполнения - 0,0869 ms

Mean result: число опытов - 10 число элементов - 1000000, количество простых - 78498, время выполнения - 11,0593 ms

Mean result: число опытов - 10 число элементов - 10000000, количество простых - 664579, время выполнения - 146,2097 ms

Модифицированный алгоритм Эратосфена:

Mean result: число опытов – 10 число элементов - 10000, количество простых - 1229, время выполнения - 0,1019 ms

Mean result: число опытов - 10 число элементов - 1000000, количество простых - 78498, время выполнения - 15,4565 ms

Mean result: число опытов – 10 число элементов - 10000000, количество простых - 664579, время выполнения - 193,9065 ms

Алгоритм №1 - декомпозиция по данным:

Mean result: число опытов - 10, число потоков - 2, число элементов - 10000, количество простых - 1229, время выполнения - 20,1323 ms

Mean result: число опытов - 10, число потоков - 2, число элементов - 1000000, количество простых - 78498, время выполнения - 28,1968 ms

Mean result: число опытов - 10, число потоков - 2, число элементов - 10000000, количество простых - 664579, время выполнения - 115,467 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000, количество простых - 1229, время выполнения - 42,1904 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 1000000, количество простых - 78498, время выполнения - 25,3004 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000000, количество простых - 664579, время выполнения - 86,7115 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000, количество простых - 1229, время выполнения - 90,2999 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 1000000, количество простых - 78498, время выполнения - 79,4581 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000000, количество простых - 664579, время выполнения - 101,2179 ms

Алгоритм №2 - декомпозиция набора простых чисел:

Средний результат: число опытов - 10, число потоков - 2, число элементов - 10000, количество простых - 1229, время выполнения - 20,8340 ms

Средний результат: число опытов - 10, число потоков - 2, число элементов - 1000000, количество простых - 78498, время выполнения - 34,5455 ms

Средний результат: число опытов - 10, число потоков - 2, число элементов - 10000000, количество простых - 664579, время выполнения - 165,1775 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000, количество простых - 1229, время выполнения - 43,4019 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 1000000, количество простых - 78498, время выполнения - 38,7795 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000000, количество простых - 664579, время выполнения - 174,4203 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000, количество простых - 1229, время выполнения - 82,6189 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 1000000, количество простых - 78498, время выполнения - 62,2042 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000000, количество простых - 664579, время выполнения - 187,9866 ms

Алгоритм №3 - применение пула потоков:

Средний результат: число опытов - 10, число элементов - 10000, количество простых - 1229, время выполнения - 0,1795 ms

Средний результат: число опытов - 10, число элементов - 1000000, количество простых - 78498, время выполнения - 9,9858 ms

Средний результат: число опытов - 10, число элементов - 10000000, количество простых - 664579, время выполнения - 96,7569 ms

Алгоритм №4 - последовательный перебор простых чисел:

Средний результат: число опытов - 10, число потоков - 2, число элементов - 10000, количество простых - 1229, время выполнения - 12,9964 ms

Средний результат: число опытов - 10, число потоков - 2, число элементов - 1000000, количество простых - 78498, время выполнения - 22,9215 ms

Средний результат: число опытов - 10, число потоков - 2, число элементов - 10000000, количество простых - 664579, время выполнения - 136,0706 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000, количество простых - 1229, время выполнения - 27,0802 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 1000000, количество простых - 78498, время выполнения - 27,9723 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000000, количество простых - 664579, время выполнения - 115,3809 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000, количество простых - 1229, время выполнения - 81,4425 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 1000000, количество простых - 78498, время выполнения - 74,0585 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000000, количество простых - 664579, время выполнения - 154,9296 ms

**Ответы на вопросы**

**Вопрос №1:** какими достоинствами и недостатками обладает каждый вариант распараллеливания?

*Последовательный алгоритм.*

Алгоритм показывает хорошие результаты расчета при небольшом числе входных данных. В таких случаях многопоточная обработка уступает за счет использования методов синхронизации и подготовки данных к вычислительным задачам, а также инициализацию самих потоков выполнения.

*№1 - Декомпозиция по данным.*

*№2 - Декомпозиция набора простых чисел.*

Вариант алгоритма 1 быстрее поскольку каждый поток работает со всем набором простых чисел на отделенном ему участке …*N.* В то время как алгоритм 2 дает только определенное количество простых чисел потокам и заставляет проверять весь массив …*N.*

*№3 - Применение пула потоков.*

Данный алгоритм оказался самым эффективным из исследуемых.

*№4 - Последовательный перебор простых чисел.*

Данный вариант оказался эффективным по сравнению с алгоритмами 1 и 2. Из-за механизма синхронизации через lock секцию решается проблема гонки данных и не происходит лишней обработки. Через критическую секцию происходит ограничение доступа к блоку кода если поток начал выполнять операции.

**Вопрос №2:** Какие средства синхронизации можно использовать вместо конструкции lock?

Какой вариант будет более эффективным?

Ниже приведены примеры выполнения алгоритма при использовании конструкций: lock, ConcurrentQueue, AutoResetEvent. Все приведенные методы имеют примерно одинаковое время выполнения.

Применение конструкции lock:

Алгоритм №4 - последовательный перебор простых чисел:

Средний результат: число опытов - 10, число потоков - 2, число элементов - 10000, количество простых - 1229, время выполнения - 12,9964 ms

Средний результат: число опытов - 10, число потоков - 2, число элементов - 1000000, количество простых - 78498, время выполнения - 22,9215 ms

Средний результат: число опытов - 10, число потоков - 2, число элементов - 10000000, количество простых - 664579, время выполнения - 136,0706 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000, количество простых - 1229, время выполнения - 27,0802 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 1000000, количество простых - 78498, время выполнения - 27,9723 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000000, количество простых - 664579, время выполнения - 115,3809 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000, количество простых - 1229, время выполнения - 81,4425 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 1000000, количество простых - 78498, время выполнения - 74,0585 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000000, количество простых - 664579, время выполнения - 154,9296 ms

Применение потокобезопасной очереди:

Алгоритм №4 - последовательный перебор простых чисел:

Средний результат: число опытов - 10, число потоков - 2, число элементов - 10000, количество простых - 1229, время выполнения - 12,1621 ms

Средний результат: число опытов - 10, число потоков - 2, число элементов - 1000000, количество простых - 78498, время выполнения - 25,3745 ms

Средний результат: число опытов - 10, число потоков - 2, число элементов - 10000000, количество простых - 664579, время выполнения - 133,5580 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000, количество простых - 1229, время выполнения - 38,1874 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 1000000, количество простых - 78498, время выполнения - 27,9022 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000000, количество простых - 664579, время выполнения - 117,5436 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000, количество простых - 1229, время выполнения - 65,0992 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 1000000, количество простых - 78498, время выполнения - 75,7979 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000000, количество простых - 664579, время выполнения - 159,0862 ms

Применение AutoResetEvent:

Алгоритм №4 - последовательный перебор простых чисел:

Средний результат: число опытов - 10, число потоков - 2, число элементов - 10000, количество простых - 1229, время выполнения - 6,2845 ms

Средний результат: число опытов - 10, число потоков - 2, число элементов - 1000000, количество простых - 78498, время выполнения - 18,3658 ms

Средний результат: число опытов - 10, число потоков - 2, число элементов - 10000000, количество простых - 664579, время выполнения - 139,6247 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000, количество простых - 1229, время выполнения - 30,9872 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 1000000, количество простых - 78498, время выполнения - 25,6241 ms

Средний результат: число опытов - 10, число потоков - 4, число элементов - 10000000, количество простых - 664579, время выполнения - 114,7924 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000, количество простых - 1229, время выполнения - 71,3385 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 1000000, количество простых - 78498, время выполнения - 81,6049 ms

Средний результат: число опытов - 10, число потоков - 8, число элементов - 10000000, количество простых - 664579, время выполнения - 155,4082 ms

Кроме выше приведенных способов синхронизации можно использовать конструкции, включающие в себя Semaphore, а также можно использовать атомарные методы синхронизации Interlocked. Использование Interlocked требует небольшое изменение алгоритма: использование целочисленного счетчика индексов и замену Queue на массив.