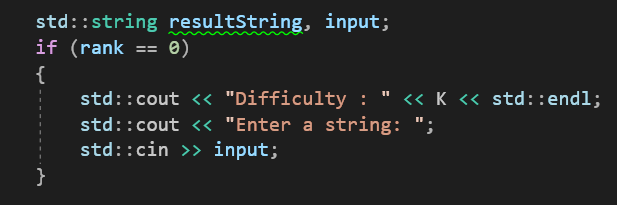
## **Отчёт по лабораторной работе №3**

Цель: реализовать программу формирования доказательства для алгоритма консенсуса PoW.

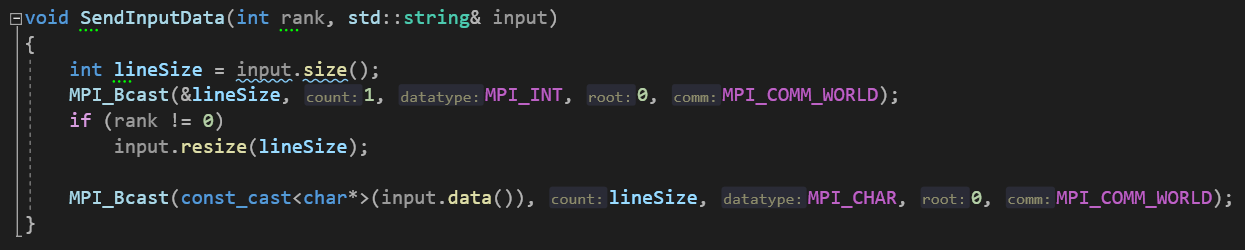
**Программная реализация:**

Инициализация строк и получения значения из консоли

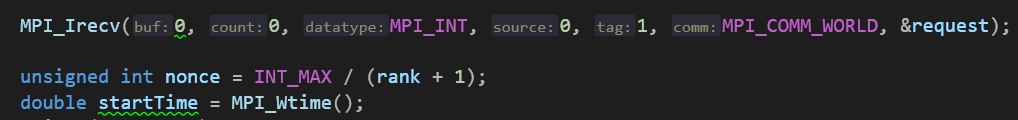
****

Костыль для отправки string, т.к. MPI\_Bcast работает только с C строками, а не С++. Из-за этого программа продолжала работать, когда нужно было рассылать данные.

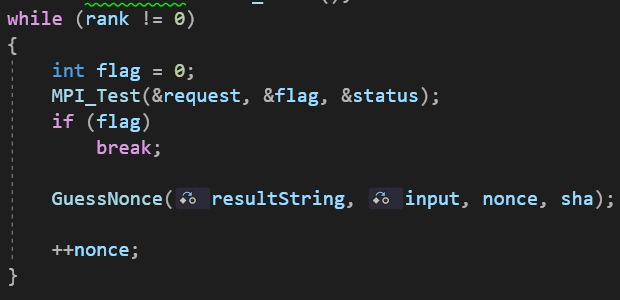


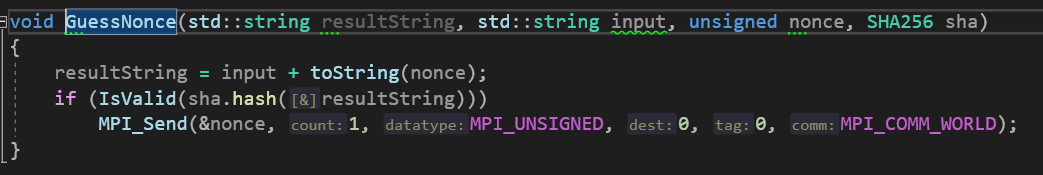


MPI\_Irecv для асинхронного получения request, чтобы остановить выполнение цикла для поиска nonce. Собственно, здесь инициализация самого nonce и запуск таймера.



Выполнение перебора ненулевыми процессами, постоянный мониторинг флага, зависящего от полученного request, а также проверка хэша на валидность для конкретной сложности

****

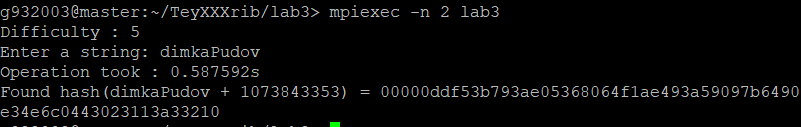
****

Так как нулевой процесс не задействован в переборе, он мониторит ситуацию с nonce. Как только получает от какого-либо процесса nonce, он начинает проверять его на валидность, то есть формирует доказательство. Если это оказалось успешным, то он отправляет request на все процессы, чтобы те остановили перебор nonce и вышли из цикла, чтобы программа завершилась.

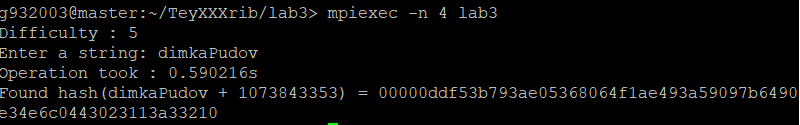
****

**Вывод программы:**

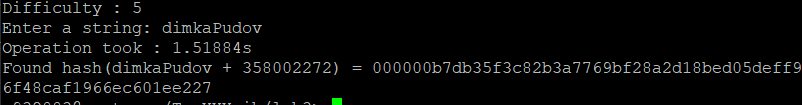
**Processes: 2**

****

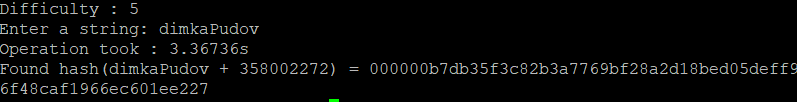
**Processes: 4**

****

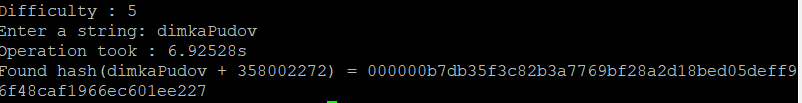
**Processes: 8**

****

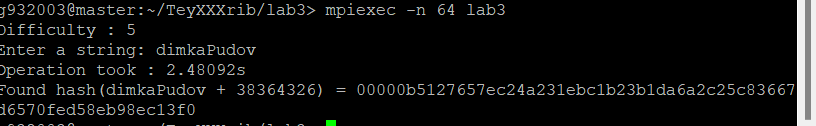
**Processes: 16**

****

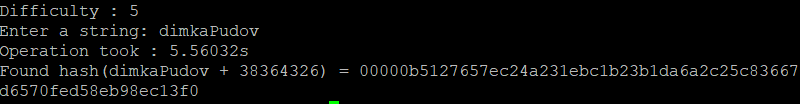
**Processes: 32**

****

**Processes: 64**

****

**Processes: 128**

****

**Заключение:**

Как можно увидеть из скриншотов с замерами выполнения операции, с повышением количества процессов повышается время, но **НЕ ВСЕГДА**. В основном это просто случайность, а также косяк программы в плане разного начального nonce, который мы инициализируем в зависимости от номера процесса