ОТЧЕТ

о выполнении ИДЗ 4

Вариант 6

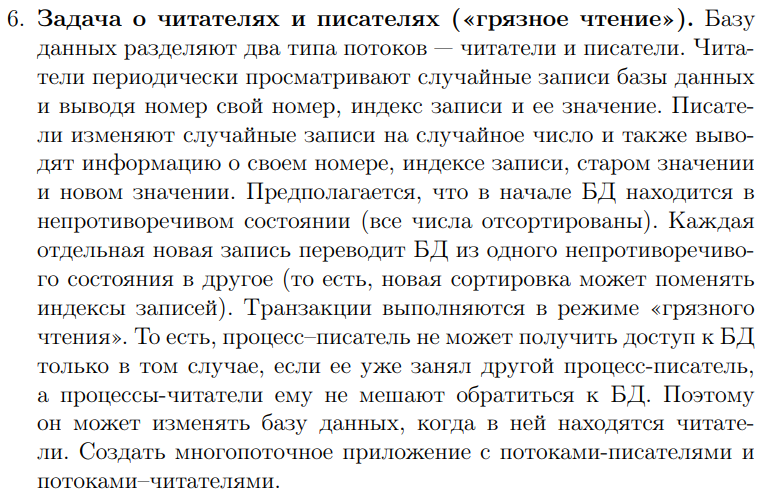
Работа на 10 баллов

Выполнила

студентка 2 курса ПИ БПИ228

Гакал Анжелика Сергеевна

Условие задачи варианта 6



Для разработки программы, описанной в задании, использовалась модель построения многопоточных приложений «Производители и потребители».

**Структура разработанной программы**

Реализация программы представлена 4 файлами:

* main.cpp,
* multithreading\_solution1.h (**основная** реализация программы),
* multithreading\_solution2.h (реализация для критерия **на 9 баллов**),
* multithreading\_solution3.h (реализация для критерия **на 10 баллов**).

Чтобы вызвать ту или иную реализацию необходимо подключить нужный header файл и в функции main вызвать соответствующую стартовую функцию:

* start\_program\_mutex – для multithreading\_solution1.h (реализация с использованием мьютексов),
* start\_program\_rwlock – для multithreading\_solution2.h (реализация с использованием rwlock),
* start\_program\_omp – для multithreading\_solution3.h (реализация с использованием OpenMP).

**Описание сценария работы программы.**

1. Вначале программа запрашивает число 1 или 2, где 1 – считывать вариативные параметры из консоли или командной строки, 2 – считать из консоли или командной строки путь к файлу, где прописаны вариативные параметры, и считать их оттуда.
2. Если введено 2, то нужно ввести 4 параметра: количество читателей, количество писателей, размер буфера, путь к файлу, куда будут дублироваться результаты из консоли. Иначе если введено 1, необходимо ввести путь к файлу, где будут прописаны вышеописанные параметры. Если было введено другое значение (не 1 и не 2), то программа выведет сообщение о некорректности данных и программа завершится.
3. Логика реализации заключается в том, чтобы контролировать то, что только один из потоков-писателей получает доступ к буферу, и то, что при выводе в консоль и записи в файл сообщения не будут перебиты разными потоками.

В основной реализации (файл multithreading\_solution1.h), используются мьютексы. mutex\_write – для регулирования работы писателей (писатель может записать значение в буфер, если только другие писатели не имеют в нему доступ); mutex\_output – для обеспечения корректности записи сообщений в консоль и в файл (чтобы сообщения читателей и писателей не перебивали друг друга)

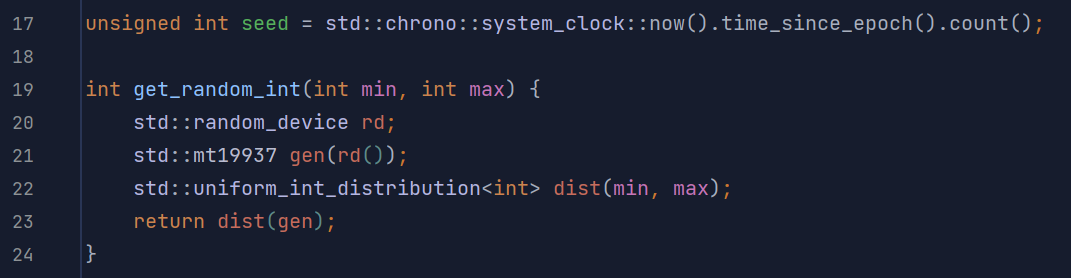
В реализации с иными синхропримитивами логика прежняя, но используются rwlock.

В третьей реализации используются возможности OpenMP. Работа писателей находится в критической секции.

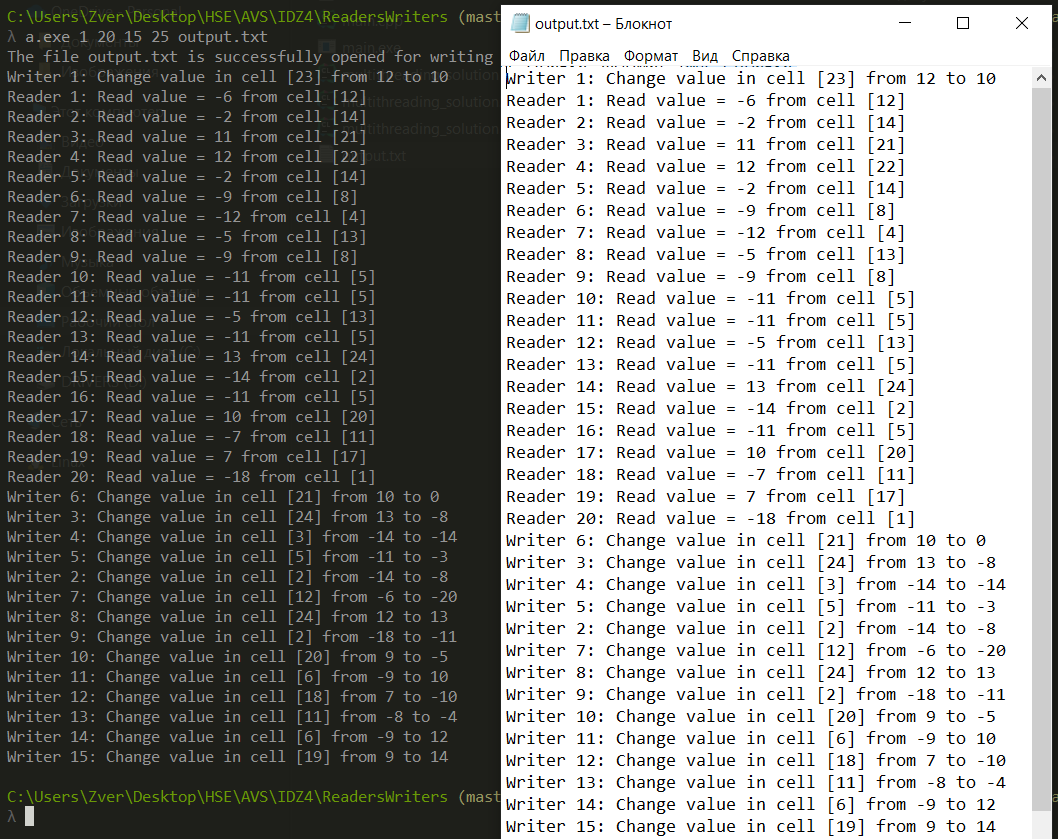
В условии задачи указано, что данные должны находиться в непротиворечивом состоянии, то есть быть отсортированы. Для этого после того, как какой-либо поток-писатель меняет значение в буфере, выполняется сортировка.

**Описание программы, подтверждающее выполнение всех критериев на 10 баллов:**

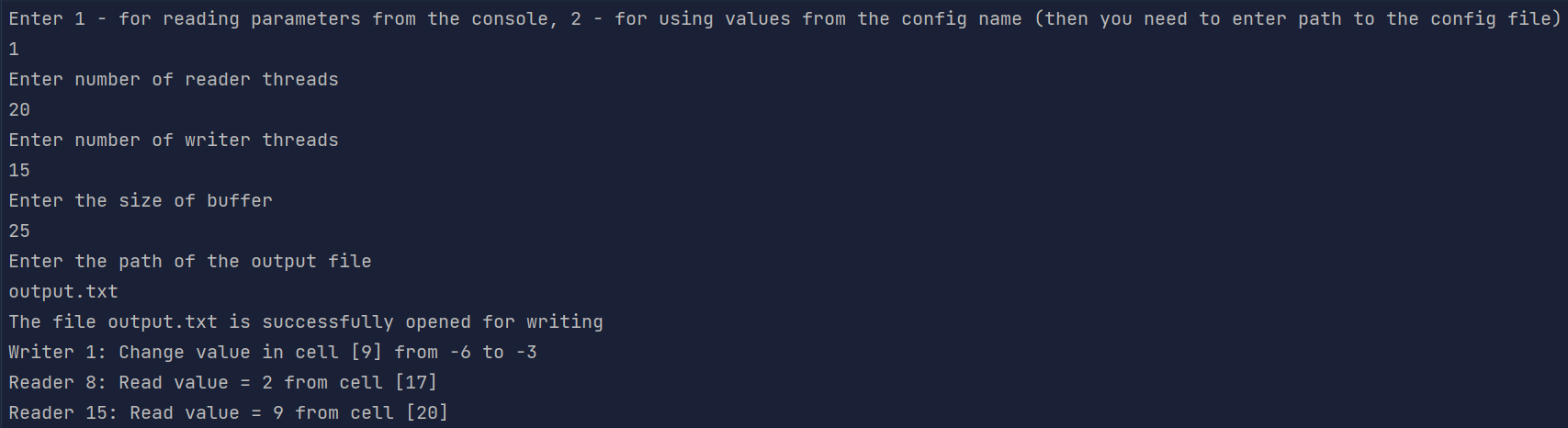
* Для генерации случайных чисел (нужно для генерации индекса элемента, к которому обратится поток, и для генерации самого элемента) используется генератор случайных чисел из равномерного распределения, seed привязан в текущему времени.

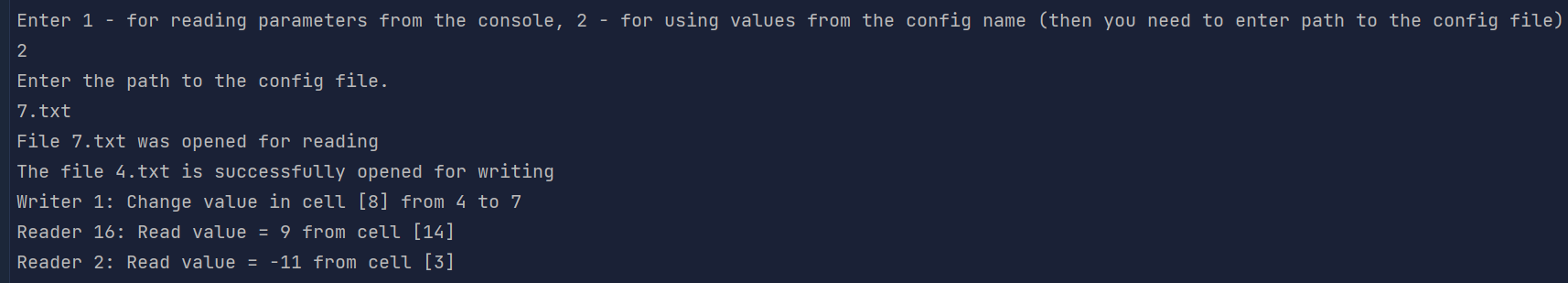


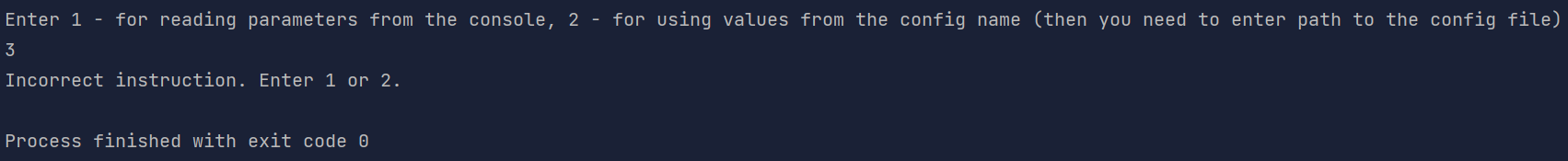
* Информация о последовательности работы программы выводится не только в консоль, но и записывается в файл, путь в которому указан последним аргументов в параметрах при вызове программы.



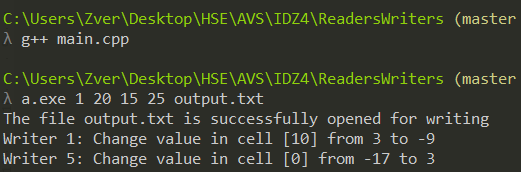
* Сценарии ввода необходимых данных из консоли

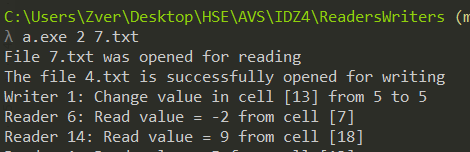


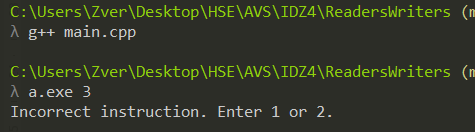




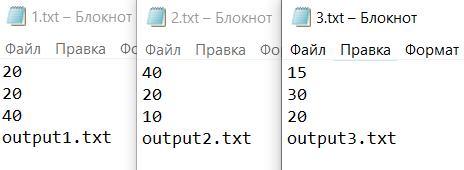
* Также реализован ввод исходных данных из командной строки. Возможные сценарии:

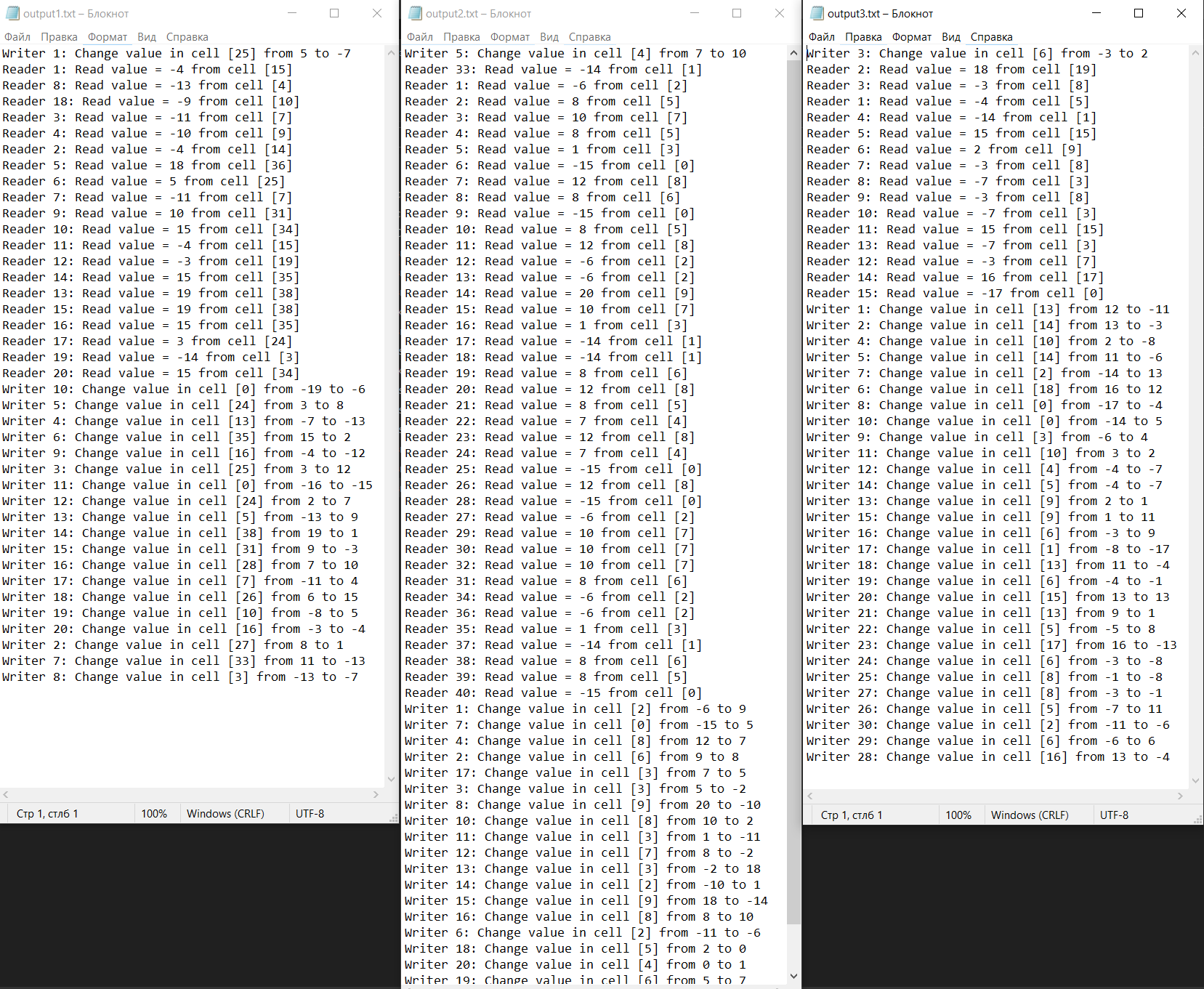




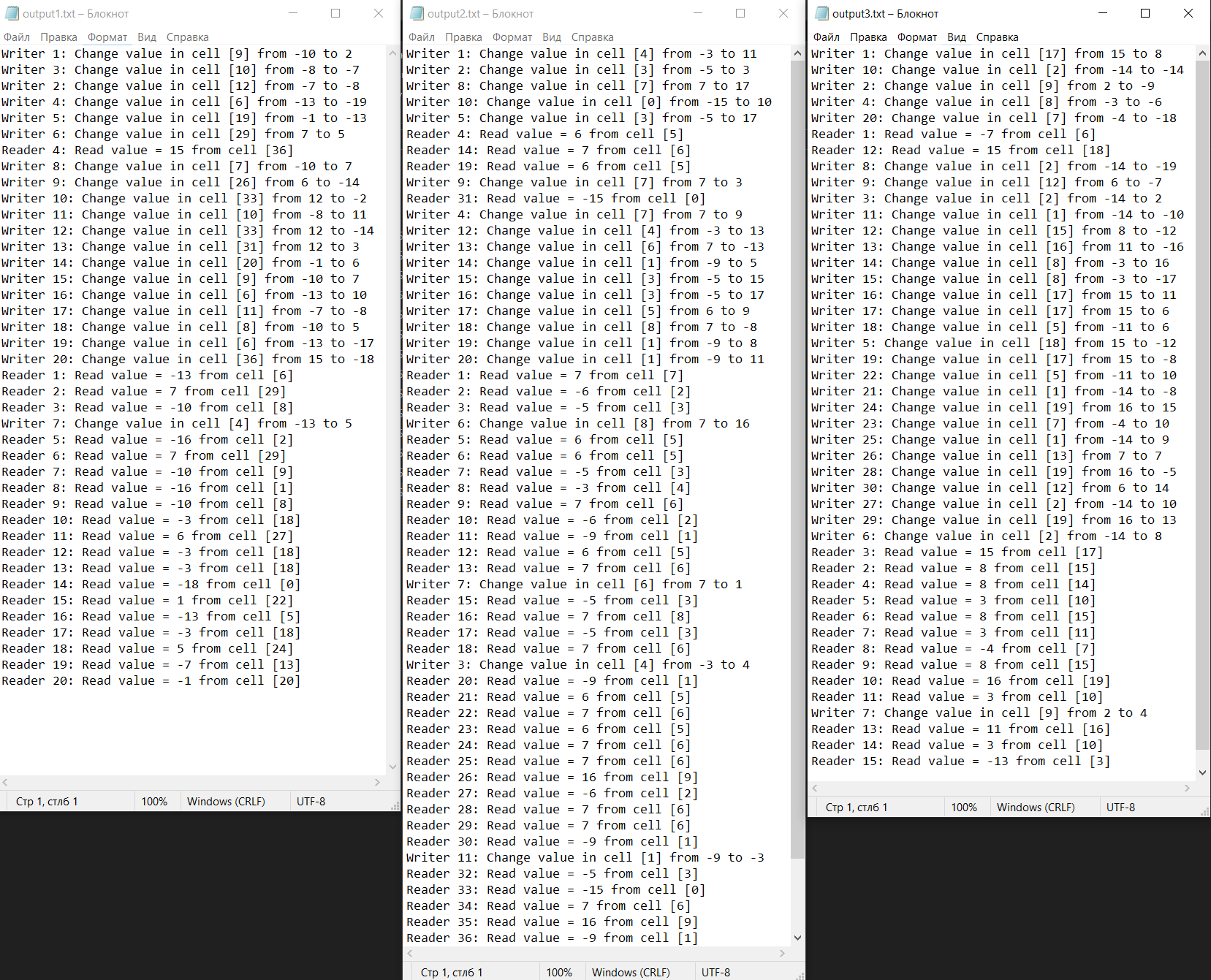


* Создано 3 файла с исходными данными: 1.txt, 2.txt, 3.txt. Информация о работе программы с основной первой реализацией на этих данными записана в файлы output1.txt, output2.txt, output3.txt соответственно.





* Лог работы программы на тех же входных данных второй реализации программы (с использованием иных синхропримитивов)



Видно, что с изменением используемых синхропримитивов логика работы программы не изменилось, программа работает корректно, вывод аналогичен приведенному раннее.

* В реализации программы с использованием OpenMP, если omp\_get\_thread\_num() возвращает четное число, то создается поток-писатель, иначе поток писатель.

Пример работы программы:

