

ABC | ИДЗ-4 | Вариант 7

Потякин Арсений Юрьевич, БПИ237

TODO:

Задача о читателях и писателях («подтвержденное чтение»). Базу данных разделяют два типа процессов — читатели и писатели. Читатели периодически просматривают случайные записи базы данных и выводя номер свой номер, индекс записи и ее значение. Писатели изменяют случайные записи на случайное число и также выводят информацию о своем номере, индексе записи, старом значении и новом значении. Предполагается, что в начале БД находится в непротиворечивом состоянии (все числа отсортированы). Каждая отдельная новая запись переводит БД из одного непротиворечивого состояния в другое (то есть, новая сортировка может поменять индексы записей). Транзакции выполняются в режиме «подтвержденного чтения», то есть процесс-писатель не может получить доступ к БД в том случае, если ее занял другой процесс-писатель или процесс-читатель. К БД может обратиться одновременно сколько угодно процессов-читателей. Процесс читатель получает доступ к БД, даже если ее уже занял процесс-писатель. Создать многопоточное приложение с потоками-писателями и потоками-читателями.

Код можно найти здесь: [GitHub](#)

Сценарий, описывающий одновременное поведение:

Допустим, у нас есть БД, состоящая из упорядоченных чисел. Тогда с этой базой работают два типа "сотрудников":

1. **Читатели:** при обращении к БД они выбирают одну случайную "запись" и смотрят ее содержимое. Читатель просто смотрит, ничего не записывая. Его можно воспринимать как сотрудника, заглядывающего в справочную систему.
2. **Писатели:** при обращении к БД они изменяют одну случайную "запись" (заменит число на новое значение). После этого они приводят базу в порядок — заново сортируют для дальнейшего удобства чтения. Его можно воспринимать как редактора, который правит справочник.

К тому же важным является тот факт, что читатели могут просматривать базу данных параллельно (в реальном мире к справочнику может обратиться несколько человек одновременно). С другой стороны, только один редактор может изменять базу данных одновременно. Пока редактор не закончит, никто не может изменить базу данных или даже прочитать ее. Как итог: читатели всегда видят актуальную информацию в удобной для восприятия форме (отсортированной), а читатели гарантируют целостность базы (отсортированность).

Модель параллельных вычислений:

Модель является стандартным подходом к SMM (Shared Memory Model) с использованием POSIX Thread. Все потоки запускаются внутри одного процесса и имеют доступ к базе данных. Для управления одновременным доступом применяется синхронизация (мьютексы и условные переменные), которые отвечают за регулирование процессов чтения/записи: читать может сколько угодно читателей одновременно, но при записи никто не может ни читать, ни записывать.

Входные данные программы:

При запуске необходимо ввести три числа:

1. **Количество записей в БД (n):** это целое положительное число. Разумный диапазон - от 1 до нескольких тысяч. В зависимости от n меняются индексные позиции для случайного чтения и записи
2. **Количество читателей:** это целое положительное число, или же количество потоков, которые одновременно будут читать БД. При увеличении числа увеличивается количество одновременно читающих базу данных. Разумный диапазон - от 1 до десятков.
3. **Количество писателей:** это целое положительное число, или же количество потоков, которые будут пытаться изменять БД. Разумный диапазон - от 1 до десятков.

Генераторы случайных чисел:

1. **Случайных индекс (для чтения и записи):** генерируется так: $idx = \text{rand}() \% n$, где n - размер массива. Таким образом, $idx \in [0, n - 1]$. Интерпретация: читатель или писатель случайным образом выбирают запись из БД для чтения/записи соответственно. В предметной области это может выглядеть как непредсказуемый интерес к разным частям базы данных.

2. **Случайное новое значение для записи:** генерируется так: `new_value = rand() % 1000`, т.е. `new_value` $\in [0, 999]$. Интерпретация: писатель заменяет старое значение на новое. Это может быть как статистика, новый код, так и новая цена.

Подробный обобщенный алгоритм:

1) Компоненты

- **База данных (database).** Это общий ресурс, доступный для чтения и записи. В программе он представлен вектором с набором чисел от 1 до $n-1$. БД сохраняет свою внутреннюю согласованность путем сортировки после каждой записи.
- **Читатели (reader_thread).** Читатели представлены в виде потоков, которые периодически просматривают случайные записи в БД. Поток читателя вызывает функция синхронизации `start_read()` и `end_read()` для гарантии корректного доступа к ресурсу.
- **Писатели (writer_thread).** Писатели представлены в виде потоков, которые периодически изменяют случайные записи в БД. Поток читателя вызывает функция синхронизации `start_write()` и `end_write()` для гарантии корректного доступа к ресурсу, то есть одновременно изменять БД может лишь один поток писателя.
- **Механизмы синхронизации: мьютексы (pthread_mutex_t) и условные переменные (pthread_cond_t).** Мьютексы используются для блокировки доступа к критическим секциям. Условные переменные обеспечивают ожидание потоков, пока ресурс (БД) недоступна, а именно: `cond_readers` используется для блокировки читателей, пока писатель пишет; `cond_writers` используется для блокировки писателей, пока другой писатель пишет или кто-то читает.
- **Дополнительный мьютекс для вывода (print_mutex).** Синхронизирует вывод данных в консоль, чтобы избежать перемешивания сообщений от разных потоков.

2) Обобщённый алгоритм

1. Программа начинает выполнение и запрашивает три параметра (n , число читателей и число писателей)
2. Инициализируется база данных (вектор)

3. Создаются R потоков-читателей и W потоков-писателей
4. Обработывают потоки-читатели и потоки-писатели (подробнее об их работе ниже)
5. Главный поток ожидает завершения всех потоков с помощью `pthread_join`
6. Программа выводит итоговый вектор (БД) и завершает работу

Подробное описание потоков:

1. **Потоки-читатели:** каждый поток выполняет чтение `reader_operations` раз:
 - (a) Поток вызывает `start_read()`, проверяет флаг `writer_active`: если писатель активен, то ожидает условной переменной `cond_readers`; если писатель неактивен, увеличивает счетчик активных читателей `read_count`
 - (b) Генерируется случайный индекс `idx`
 - (c) Читается значение из базы данных по этому `idx`
 - (d) Читатель блокирует мьютекс `print_mutex` и выводит информацию о чтении
 - (e) Читатель освобождает мьютекс
 - (f) Поток-читатель вызывает `end_read()`, тем самым уменьшая текущее количество читателей БД
 - (g) Если это был последний читатель - вызывается `cond_reader` (для завершения программы)
2. **Потоки-писатели:** каждый поток выполняет запись `writer_operations` раз:
 - (a) Поток вызывает `start_write()`, проверяет флаг `writer_active`: если писатель активен или есть активные читатели, то поток ожидает условной переменной `cond_writer`, иначе блокирует мьютекс и устанавливает флаг `writer_active = true`
 - (b) Генерируется случайный индекс `idx` и новое значение `new_value`
 - (c) Изменяется значение из базы данных по этому `idx` на новое
 - (d) БД проводит балансировку (сортируется)
 - (e) при помощи `std::find` находится новый индекс элемента

- (f) Писатель блокирует мьютекс `print_mutex` и выводит информацию о записи (идентификатор писателя, старое и новое значение, новый индекс)
- (g) Писатель освобождает мьютекс
- (h) Поток-читатель вызывает `end_write()`, тем самым снимая флаг записи
- (i) Подается сигнал всем ожидающим открытия БД

Критерий на 9 баллов – другой синхропримитив:

Ключевые изменения: теперь используются барьеры для синхронизации между писателями и читателями. Барьер позволяет потокам не ждать пока какое-то количество потоков достигнет точки синхронизации, после чего они продолжают выполнение. Кроме того, вместо использования `pthread_cond_t` используется обычный булев флаг, указывающий состояние писателя. Значения генерируются случайно, как и в предыдущей версии.

Алгоритм работы:

1. Читатели:

- Проверяют флаг писателя, если неактивен, то увеличивается счетчик активных читальщиков и выполняется чтение. Если активен, то читатель ждет на барьере
- После чтения счетчик читателей уменьшается. Если это был последний читатель, то писатель может начать работу (если он ждал на барьере)

2. Писатели:

- Проверяют сколько читателей читают БД. Если 0, то ставят флаг читателя
- После записи убирают флаг читателя

Все остальные этапы работы совпадают с первой версией программы.

Сравнительный анализ:

Корректность поведения: Первая версия программы:

1. Использует мьютексы для защиты критических секций и условные переменные для блокировки потоков до выполнения некоторых условий

2. Читатели блокируются, если писатель активен. Писатели блокируются, если активны читатели или другой писатель

Новая версия программы:

1. Вместо условных переменных используется барьер
2. Читатели проверяют флаг писателя. Если он активен, то они ждут на барьере
3. Писатель ждет пока число читателей будет равным 0 и пока флаг писателя будет неактивным. В таком случае писатель устанавливает флаг писателя

Идентичность:

1. Обе программы гарантируют, что у писателя будет эксклюзивный доступ к БД, а у читателей - совместный
2. Вывод программ при одинаковых входных данных идентичен (ниже приведены скриншоты), так как порядок выполнения операций идентичен

Критерий на 10 баллов: OpenMP:

Сравнительный анализ:

POSIX Threads:

1. Использует мьютексы и условные переменные для синхронизации потоков
2. Использует низкоуровневые инструменты (pthreads)

OpenMP:

1. Использует OpenMP с директивами для создания параллельных потоков и синхронизации
2. Используются критические секции и атомарные операции, а также флаг для координации потоков

Обе программы гарантируют корректность выполнения и аналогичность ответов при одних и тех же входных данных.

СКРИНШОТЫ:

```
Введите количество записей в базе данных: 2
Введите количество читателей: 2
Введите количество писателей: 2
[Читатель 1] читает: индекс=0, значение=0
[Читатель 2] читает: индекс=1, значение=1
[Писатель 1] записал: старый_индекс=0, старое_значение=0, новое_значение=224, новый_индекс=1
[Писатель 2] записал: старый_индекс=0, старое_значение=1, новое_значение=714, новый_индекс=1
[Читатель 1] читает: индекс=1, значение=714
```

Рис. 1: При запуске программы необходимо ввести три значения, после чего программа начинает выполняться (со случайными значениями)

```
[Читатель 2] читает: индекс=0, значение=611
[Писатель 2] записал: старый_индекс=1, старое_значение=971, новое_значение=84, новый_индекс=0
[Писатель 1] записал: старый_индекс=1, старое_значение=611, новое_значение=209, новый_индекс=1
[Читатель 2] читает: индекс=1, значение=209
[Читатель 1] читает: индекс=0, значение=84
[Писатель 2] записал: старый_индекс=0, старое_значение=84, новое_значение=711, новый_индекс=1
[Писатель 1] записал: старый_индекс=1, старое_значение=711, новое_значение=648, новый_индекс=1
[Писатель 2] записал: старый_индекс=1, старое_значение=648, новое_значение=6, новый_индекс=0
[Писатель 1] записал: старый_индекс=1, старое_значение=209, новое_значение=915, новый_индекс=1
Все потоки завершили работу. Итоговый массив:
6 915
```

Рис. 2: (часть вывода обрезана), после выполнения программы в консоль выводится ответ. Вывод в консоль является подробным и отражает все изменения, происходящие с БД

```
// Находим новый индекс вставленного значения
int new_idx = (int)(std::find(first: database.begin(), last: database.en

// Вывод под защитой print_mutex
pthread_mutex_lock(&print_mutex);
std::cout << "[Писатель " << id << "] записал: старый_индекс=" << :
    << ", старое_значение=" << old_value
    << ", новое_значение=" << new_value
    << ", новый_индекс=" << new_idx << std::endl;
pthread_mutex_unlock(&print_mutex);
```

Рис. 3: В коде присутствуют комментарии, объясняющие логику выполнения некоторых блоков кода

```

7035137/af13e217/05101f05e5c3/kob_1b2_4/эмке_вогса_савоу/kob_1b2_4
Введите количество записей в базе данных: 2
Введите количество читателей: 2
Введите количество писателей: 2
Выберите режим ввода (1 - ручной, 2 - случайный, 3 - из файла): |

```

Рис. 4: При этом программа дает выбор: ввести данные вручную (в консоль), создать случайные данные или собрать данные из файла, по порядку:

```

Выберите режим ввода (1 - ручной, 2 - случайный, 3 - из файла): 1
[Ручной ввод] Читатель 1, операция чтения 1: введите индекс для чтения [0..1]: 0
[Ручной ввод] Читатель 1, операция чтения 2: введите индекс для чтения [0..1]: 1
[Ручной ввод] Читатель 1, операция чтения 3: введите индекс для чтения [0..1]:

```

Рис. 5: При выборе ручного ввода программа предлагает ввести значения для чтения, для записи (и новое значение впоследствии) случайно

```

Введите количество записей в базе данных: 2
Введите количество читателей: 2
Введите количество писателей: 2
Выберите режим ввода (1 - ручной, 2 - случайный, 3 - из файла): 2
[Читатель 1] читает: индекс=0, значение=0
[Читатель 2] читает: индекс=0, значение=0
[Писатель 1] записал: старый_индекс=1, старое_значение=1, новое_значени
[Писатель 2] записал: старый_индекс=1, старое_значение=77, новое_знач

```

Рис. 6: Случайный ввод данных (генерация)

```

Введите количество записей в базе данных: 2
Введите количество читателей: 2
Введите количество писателей: 2
Выберите режим ввода (1 - ручной, 2 - случайный, 3 - из файла): 3
Введите имя файла ввода: pupupu.txt
Не удалось открыть файл ввода.

```

Рис. 7: Ввод данных из файла. В случае, если файла нет или его невозможно открыть, то программа уведомляет пользователя и завершает работу


```
Введите количество записей в базе данных: 1
Введите количество читателей: 1
Введите количество писателей: 1
Выберите режим ввода (1 - ручной, 2 - случайный, 3 - из файла): 3
Введите имя файла ввода: /Users/arseniy/CLionProjects/ACS-IDZ-4/input1.txt
Ошибка в формате входных данных для читателей.
```

Рис. 8: Ввод данных из файла. В случае, если файла нет или его невозможно открыть, то программа уведомляет пользователя и завершает работу

```
1 [Читатель 2] читает: индекс=1, значение=1
2 [Читатель 1] читает: индекс=0, значение=0
3 [Писатель 1] записал: старый_индекс=0, старое_значение=0, новое_значение=1
4 [Писатель 2] записал: старый_индекс=1, старое_значение=963, новое_значение=1
5 [Читатель 1] читает: индекс=1, значение=44
6 [Читатель 2] читает: индекс=1, значение=44
7 [Писатель 1] записал: старый_индекс=1, старое_значение=44, новое_значение=1
8 [Писатель 2] записал: старый_индекс=1, старое_значение=176, новое_значение=1
9 [Читатель 1] читает: индекс=1, значение=687
10 [Читатель 2] читает: индекс=1, значение=687
11 [Читатель 1] читает: индекс=0, значение=1
12 [Читатель 2] читает: индекс=0, значение=1
13 [Писатель 1] записал: старый_индекс=1, старое_значение=687, новое_значение=1
14 [Писатель 2] записал: старый_индекс=0, старое_значение=1, новое_значение=1
15 [Читатель 1] читает: индекс=1, значение=717
16 [Читатель 2] читает: индекс=1, значение=717
17 [Писатель 1] записал: старый_индекс=1, старое_значение=717, новое_значение=1
18 [Писатель 2] записал: старый_индекс=1, старое_значение=321, новое_значение=1
19 [Писатель 1] записал: старый_индекс=1, старое_значение=421, новое_значение=1
20 [Писатель 2] записал: старый_индекс=1, старое_значение=911, новое_значение=1
21 Все потоки завершили работу. Итоговый массив:
22 215 224
23
```

Run ACS_IDZ_4

```
[Читатель 2] читает: индекс=1, значение=717
[Писатель 1] записал: старый_индекс=1, старое_значение=717, новое_значение=1
[Писатель 2] записал: старый_индекс=1, старое_значение=321, новое_значение=1
[Писатель 1] записал: старый_индекс=1, старое_значение=421, новое_значение=1
[Писатель 2] записал: старый_индекс=1, старое_значение=911, новое_значение=1
Все потоки завершили работу. Итоговый массив:
215 224
```

Рис. 9: Вне зависимости от типа ввода данные будут выведены и в консоль, и в файл output.txt (на скриншоте снизу консоль, сверху - открытый файл, содержимое идентично)

```

~/CLionProjects/ACS-IDZ-4 git:(main)±32 (0.717s)
clang++ -std=c++20 main.cpp -o program

~/CLionProjects/ACS-IDZ-4 git:(main)±32 (1.409s)
./program 2 2 2 output10.txt
[Читатель 1] читает: индекс=1, значение=1
[Читатель 2] читает: индекс=1, значение=1
[Писатель 1] записал: старый_индекс=0, старое_значение=0, новое_значение=405, новый_индекс=1
[Писатель 2] записал: старый_индекс=0, старое_значение=1, новое_значение=347, новый_индекс=0
[Читатель 1] читает: индекс=1, значение=405
[Читатель 2] читает: индекс=0, значение=347
[Писатель 2] записал: старый_индекс=1, старое_значение=405, новое_значение=263, новый_индекс=0
[Писатель 1] записал: старый_индекс=1, старое_значение=347, новое_значение=639, новый_индекс=1
[Читатель 2] читает: индекс=1, значение=639
[Читатель 1] читает: индекс=1, значение=639
[Читатель 1] читает: индекс=0, значение=263
[Читатель 2] читает: индекс=1, значение=639
[Писатель 2] записал: старый_индекс=0, старое_значение=263, новое_значение=250, новый_индекс=0
[Писатель 1] записал: старый_индекс=0, старое_значение=250, новое_значение=68, новый_индекс=0
[Читатель 2] читает: индекс=0, значение=68
[Читатель 1] читает: индекс=1, значение=639
[Писатель 2] записал: старый_индекс=1, старое_значение=639, новое_значение=534, новый_индекс=1
[Писатель 1] записал: старый_индекс=0, старое_значение=68, новое_значение=495, новый_индекс=0
[Писатель 1] записал: старый_индекс=1, старое_значение=534, новое_значение=61, новый_индекс=0
[Писатель 2] записал: старый_индекс=1, старое_значение=495, новое_значение=313, новый_индекс=1
Все потоки завершили работу. Итоговый массив:
61 313

```

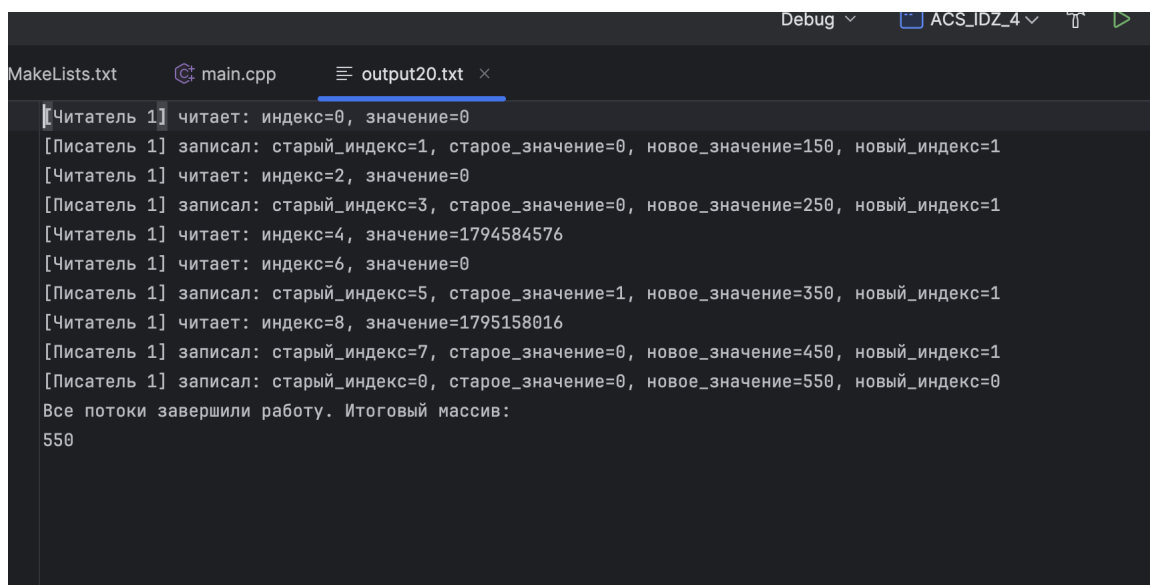
Рис. 10: Программу можно запустить из консоли. При вводе значений N, R, W (первые три аргумента) программа находится в "случайном" режиме и генерирует данные случайно, при этом данные выводятся как в консоль, так и в указанный файл

```
CMakeLists.txt  main.cpp  output10.txt x
[Читатель 2] читает: индекс=1, значение=1
[Писатель 1] записал: старый_индекс=0, старое_значение=0, новое_значение=405, новый_индекс=1
[Писатель 2] записал: старый_индекс=0, старое_значение=1, новое_значение=347, новый_индекс=0
[Читатель 1] читает: индекс=1, значение=405
[Читатель 2] читает: индекс=0, значение=347
[Писатель 2] записал: старый_индекс=1, старое_значение=405, новое_значение=263, новый_индекс=0
[Писатель 1] записал: старый_индекс=1, старое_значение=347, новое_значение=639, новый_индекс=1
[Читатель 2] читает: индекс=1, значение=639
[Читатель 1] читает: индекс=1, значение=639
[Читатель 1] читает: индекс=0, значение=263
[Читатель 2] читает: индекс=1, значение=639
[Писатель 2] записал: старый_индекс=0, старое_значение=263, новое_значение=250, новый_индекс=0
[Писатель 1] записал: старый_индекс=0, старое_значение=250, новое_значение=68, новый_индекс=0
[Читатель 2] читает: индекс=0, значение=68
[Читатель 1] читает: индекс=1, значение=639
[Писатель 2] записал: старый_индекс=1, старое_значение=639, новое_значение=534, новый_индекс=1
[Писатель 1] записал: старый_индекс=0, старое_значение=68, новое_значение=495, новый_индекс=0
[Писатель 1] записал: старый_индекс=1, старое_значение=534, новое_значение=61, новый_индекс=0
[Писатель 2] записал: старый_индекс=1, старое_значение=495, новое_значение=313, новый_индекс=1
Все потоки завершили работу. Итоговый массив:
61 313
```

Рис. 11: Идентичные данные в файле

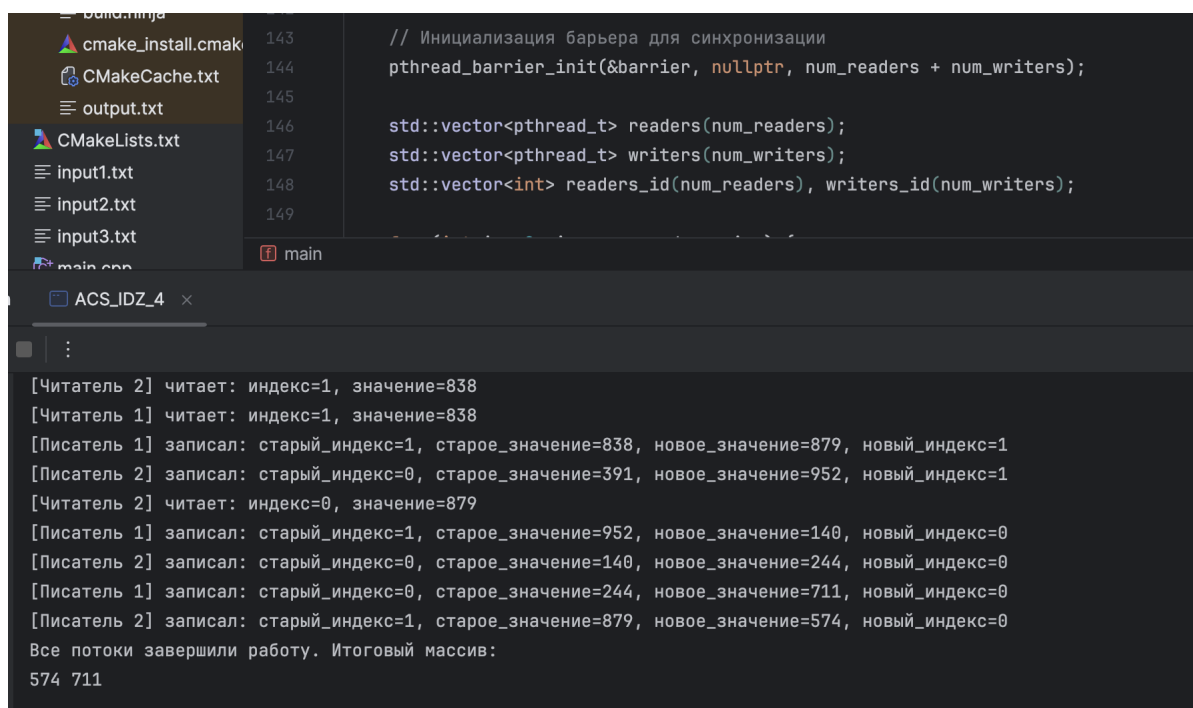
```
~/CLionProjects/ACS-IDZ-4 git:(main)±32 (1.075s)
./program /Users/arseniy/CLionProjects/ACS-IDZ-4/input1.txt output20.txt
[Читатель 1] читает: индекс=0, значение=0
[Писатель 1] записал: старый_индекс=1, старое_значение=0, новое_значение=150, новый_индекс=1
[Читатель 1] читает: индекс=2, значение=0
[Писатель 1] записал: старый_индекс=3, старое_значение=0, новое_значение=250, новый_индекс=1
[Читатель 1] читает: индекс=4, значение=1794584576
[Читатель 1] читает: индекс=6, значение=0
[Писатель 1] записал: старый_индекс=5, старое_значение=1, новое_значение=350, новый_индекс=1
[Читатель 1] читает: индекс=8, значение=1795158016
[Писатель 1] записал: старый_индекс=7, старое_значение=0, новое_значение=450, новый_индекс=1
[Писатель 1] записал: старый_индекс=0, старое_значение=0, новое_значение=550, новый_индекс=0
Все потоки завершили работу. Итоговый массив:
550
```

Рис. 12: Если же указывать два параметра - имя входного и выходного файла, то программа будет считывать данные из файла. Аналогично, вывод происходит как в консоль, так и в указанный файл



```
MakeLists.txt  main.cpp  output20.txt x
[Читатель 1] читает: индекс=0, значение=0
[Писатель 1] записал: старый_индекс=1, старое_значение=0, новое_значение=150, новый_индекс=1
[Читатель 1] читает: индекс=2, значение=0
[Писатель 1] записал: старый_индекс=3, старое_значение=0, новое_значение=250, новый_индекс=1
[Читатель 1] читает: индекс=4, значение=1794584576
[Читатель 1] читает: индекс=6, значение=0
[Писатель 1] записал: старый_индекс=5, старое_значение=1, новое_значение=350, новый_индекс=1
[Читатель 1] читает: индекс=8, значение=1795158016
[Писатель 1] записал: старый_индекс=7, старое_значение=0, новое_значение=450, новый_индекс=1
[Писатель 1] записал: старый_индекс=0, старое_значение=0, новое_значение=550, новый_индекс=0
Все потоки завершили работу. Итоговый массив:
550
```

Рис. 13: Вывод в файл



```
cmake_install.cmake 143
CMakeCache.txt 144
output.txt 145
CMakeLists.txt 146
input1.txt 147
input2.txt 148
input3.txt 149
main.cpp main
ACS_IDZ_4 x
[Читатель 2] читает: индекс=1, значение=838
[Читатель 1] читает: индекс=1, значение=838
[Писатель 1] записал: старый_индекс=1, старое_значение=838, новое_значение=879, новый_индекс=1
[Писатель 2] записал: старый_индекс=0, старое_значение=391, новое_значение=952, новый_индекс=1
[Читатель 2] читает: индекс=0, значение=879
[Писатель 1] записал: старый_индекс=1, старое_значение=952, новое_значение=140, новый_индекс=0
[Писатель 2] записал: старый_индекс=0, старое_значение=140, новое_значение=244, новый_индекс=0
[Писатель 1] записал: старый_индекс=0, старое_значение=244, новое_значение=711, новый_индекс=0
[Писатель 2] записал: старый_индекс=1, старое_значение=879, новое_значение=574, новый_индекс=0
Все потоки завершили работу. Итоговый массив:
574 711
```

Рис. 14: Версия для критерия 9. Как видно, версия с барьерами имеет аналогичный вывод как и версия с мьютексами

```
~/CLionProjects/ACS-IDZ-4 git:(main)±32 (0.53s)
clang++ -Xpreprocessor -fopenmp -I/opt/homebrew/opt/libomp/include -L/opt/homebrew/opt/libomp/lib main.cpp -o program -lomp
main.cpp:124:20: warning: range-based for loop is a C++11 extension [-Wc++11-extensions]
    124 |     for (int value : database) {
        |                ^
1 warning generated.

~/CLionProjects/ACS-IDZ-4 git:(main)±32 (11.537s)
./program 2 2 2 output3.txt
Введите количество записей в базе данных: 2
Введите количество читателей: 2
Введите количество писателей: 2
[Читатель 2] читает: индекс=1, значение=1
[Читатель 1] читает: индекс=0, значение=0
[Писатель 2] записал: старый_индекс=1, старое_значение=1, новое_значение=617
[Писатель 1] записал: старый_индекс=0, старое_значение=0, новое_значение=858
[Читатель 2] читает: индекс=1, значение=858
[Читатель 1] читает: индекс=1, значение=858
[Писатель 1] записал: старый_индекс=1, старое_значение=858, новое_значение=968
[Писатель 2] записал: старый_индекс=1, старое_значение=858, новое_значение=899
[Читатель 1] читает: индекс=1, значение=968
[Читатель 2] читает: индекс=1, значение=968
[Читатель 1] читает: индекс=1, значение=968
[Читатель 2] читает: индекс=0, значение=617
[Писатель 1] записал: старый_индекс=1, старое_значение=968, новое_значение=856
[Писатель 2] записал: старый_индекс=0, старое_значение=617, новое_значение=809
[Читатель 1] читает: индекс=0, значение=809
[Читатель 2] читает: индекс=1, значение=856
[Писатель 1] записал: старый_индекс=1, старое_значение=856, новое_значение=19
[Писатель 2] записал: старый_индекс=1, старое_значение=19, новое_значение=775
[Писатель 1] записал: старый_индекс=0, старое_значение=775, новое_значение=639
[Писатель 2] записал: старый_индекс=0, старое_значение=639, новое_значение=345
Все потоки завершили работу. Итоговый массив:
345 809
```

Рис. 15: Версия для критерия 10. Как видно, версия с OpenMP имеет аналогичный вывод как и версия с мьютексами и барьерами. К сожалению, OpenMP не хочет компилироваться в моем IDE (CLion), поэтому запуск проводится из терминала. Перед запуском программы производится сборка с соответствующими параметрами.