В этой задаче вам предстоит реализовать одну из фундаментальных структур - “канал” (channel). Канал позволяет организовать передачу данных от приемника к источнику при помощи следующих методов:

- send(value) —- отправить value через канал.
- recv() —- получить очередное значение из канала.
- close() —- закрыть канал для передачи и приема.

Важнейшей особенностью такого канала является возможность сразу нескольких источников писать в один канал данные и сразу нескольким получателям их принимать без всякой дополнительной синхронизации с их стороны. Это позволяет удобным образом организовывать вычислительные конвейеры – каждая функция в конвейере принимает данные из одного канала и пишет результат их обработки в другой канал для следующей функции. При этом каждая функция может исполняться сразу несколькими потоками, а разработчику даже не нужно думать о распараллеливании вычислений (заниматься распределением данных между потоками, следить за синхронизацией) – всю эту работу канал берет на себя.

Гарантируется, что T имеет 2, а также оператор перемещающего присваивания.

Буферизованный канал (асинхронные операции)

Для буферизованных каналов в момент создания определяется емкость некоторого внутреннего буфера (buffer_size), работающего по принципу FIFO. Алгоритм работы:

Send(value) – если текущее количество элементов в очереди равно buffer_size (т.е. очередь заполнена), то поток блокируется до тех пор, пока в очереди не появится свободное место. После этого value записывается в конец очереди.

Recv() – если очередь пуста, то поток блокируется до тех пор, пока в очередь не будет что-нибудь записано. После этого из головы очереди извлекается очередное значение и возвращается как результат. Таким образом, каждое значение может быть получено только в 1 экземпляре, но получающий поток (из тех, что сделали recv) может быть любым.

Close() – закрывает канал. Если в очереди еще содержатся какие-то элементы, то канал продолжает заниматься их отправкой, однако прочие попытки сделать send после закрытия канала приводят к ошибке (send должен бросать исключение типа std::runtime_error при попытке отправить через закрытый канал). В то же время, Recv() после закрытия дочитывает оставшиеся в очереди элементы, а все дальнейшие вызовы должны возвращать значение, сигнализирующее о закрытии канала (второй параметр в возвращаемой паре - флаг закрытия (значение false если реального значения нет), первый при закрытом канале - объект, созданный конструктором по умолчанию).

На проверку сдаётся файл buffered_channel.h с исходным кодом решения.

Потоки и блокировки
Условные переменные