ПРАВИТЕЛЬСТВО РОССИЙСКОЙ ФЕДЕРАЦИИ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет компьютерных наук
Департамент программной инженерии

МНОГОПОТОЧНОЕ ПРИЛОЖЕНИЕ ДЛЯ СИМУЛЯЦИИ РАБОТЫ СУПЕРМАРКЕТА.

Пояснительная записка

Исполнитель:
студент группы БПИ199
_____ Т. С. Кочарян
«_____» _____ 2020 г.

Москва 2020

Оглавление:

1. Текст задания.

Задача о супермаркете. В супермаркете работают два кассира, покупатели заходят в супермаркет, делают покупки и становятся в очередь к случайному кассиру. Пока очередь пуста, кассир спит, как только появляется покупатель, кассир просыпается. Покупатель спит в очереди, пока не подойдет к кассиру. Создать многопоточное приложение, моделирующее рабочий день супермаркета.

2. Применяемые расчетные методы.

Для симуляции работы, согласно условиям, в реализации используются 3 вида потоков. Первый поток - это поток работы первого кассира, второй вид потоков - это поток работы второго кассира, и, наконец, третий вид потоков - поток работы типичного покупателя. Каждый поток имеет семафоры и мютексы для регулирования использование потоков переменных связанных с потоком.

Для реализации многопоточности была выбрана модель “Клиенты и серверы”. Клиентским потоком здесь являются клиенты магазина, которые встают в очередь, а серверным потоком является кассир. Это обусловлено тем, что кассир находится в ожидании запроса от клиента, и когда клиент встает в очередь, он пробуждает поток кассира и сам засыпает в ожидании запроса от кассира, который затем действует в соответствии с поступившим запросом. Таким образом, с помощью данной модели реализуется многопоточность.

Для реализации работы потоков используется библиотека `thread.`

Для реализации работы семафоров используется - `semaphore.h,` а для мютексов - `mutex.`

Для хранения семафоров потоков клиентов используется контейнер `std::queue`.

Для хранения векторов используется контейнер `std::vector`.

Для ввода/вывода в консоль используется библиотека `iostream`.

3. Список используемых источников.

[1] Инструкция по составлению пояснительной записки [Электронный ресурс]. //URL: http://softcraft.ru/edu/comparch/tasks/mp01/ (Дата обращения: 07.12.2020, режим доступа: свободный)

[2] Статья “Потоки, блокировки и условные переменные в C++11” [Электронный ресурс]. //URL: https://habr.com/ru/post/182610/ (Дата обращения: 07.12.2020, режим доступа: свободный)

[3] Руководство std::mutex [Электронный ресурс]. //URL: https://en.cppreference.com/w/cpp/thread/mutex (Дата обращения: 07.12.2020, режим доступа: свободный)

[4] Multithreading in C++ [Электронный ресурс]. //URL: https://www.geeksforgeeks.org/multithreading-in-cpp/ (Дата обращения: 07.12.2020, режим доступа: свободный)

[5] Статья “Такие удивительные семафоры“ [Электронный ресурс]. //URL: https://habr.com/ru/post/261273/ (Дата обращения: 07.12.2020, режим доступа: свободный)

[6] Статья “Клиент-серверный процессор командной строки“ [Электронный ресурс]. //URL: https://wm-help.net/lib/b/book/3539737877/284 (Дата обращения: 07.12.2020, режим доступа: свободный)

4. Описание работы программы.

Описание инициализации:

В самом начале программа просит пользователя ввести число N - количество клиентов, которых 2 кассира должны обслужить. Это число должно быть типа int и быть >= 0. Если число равно 0, то программа завершается с сообщением, что очереди пусты. Если же число < 0, то программа завершает работа с сообщением об ошибке. В противном случае, программа начинает работу.

В глобальные переменные вынесена информация о каждом кассире и о каждой очереди. Кассир представлен семафором, а очередь контейнером queue и мютексом.

Чтобы выбрать, в какую же очередь все-таки пойдет клиент, программа генерирует число от 0 до 1. Каждое число соответствует очереди: 0 - первая очередь, 1 - вторая очередь.

Для корректной работы семафоров кассиров, они инициализируется в самом начале работы программы. Затем, с помощью цикла for, программа генерирует N потоков клиентов. Каждый клиент имеет семафор, который при инициализации добавляется в queue.

Затем создаются потоки кассиров, которые занимаются обслуживанием клиентов из queue.

После создания этих потоков, они объединяются с основным потоком с помощью метода join().

Взаимодействие потоков:

Как уже описывалось ранее, взаимодействие потоков происходит согласно модели “клиент-сервер” [см. 6]

Метод, переданный в поток клиента, инициализирует семафор и добавляется в очередь. В момент добавления, вызывается `sem_post()`, который “будит” семафор кассира. Затем, вызывается метод `sem_wait()`, который ожидает дальнейших действий кассира. Во время данных действий используются мютексы, которых не позволяют нескольким потокам одновременно вызывать `sem_post(&first_cashier)`.

Затем потоки кассиров проверяют, пуста ли очередь. Если да, то завершают свою работу. Если нет, то пока очередь будет иметь хотя бы одного клиента, кассир этой очереди будет ожидать, пока его разбудят, чтобы обслужить. Напомню, что при инициализации, клиент вызывает `sem_post(&cashier)`, что пробуждает поток кассира, а сам клиент засыпает. Таким образом, пробужденный поток кассира пробуждает поток первого клиента в очереди queue. Пробужденный клиент выводит в консоль информацию о том, какой кассир его обслужил и какой у клиента был номер.

Таким образом, кассиры пробуждаются сами и пробуждают клиентов в порядке очереди и засыпают, если в очереди никто их не пробуждает.

Замечание:

Хочется отметить, что операции `sem_post()`, `queue.front()` и `queue.pop()`, а также cout окружены мютексами, которые не позволяют потокам конфликтовать друг с другом.

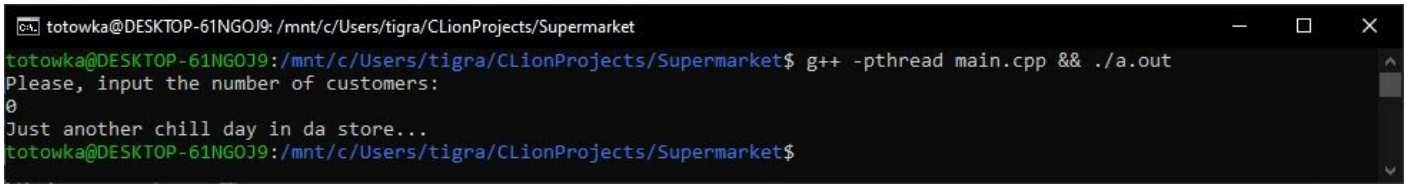
5. Тестирование программы.

Крайними случаями в данной программе могут быть некорректные данные о количестве клиентов, считанные из консоли. Данная программа корректно обрабатывает такие случаи.

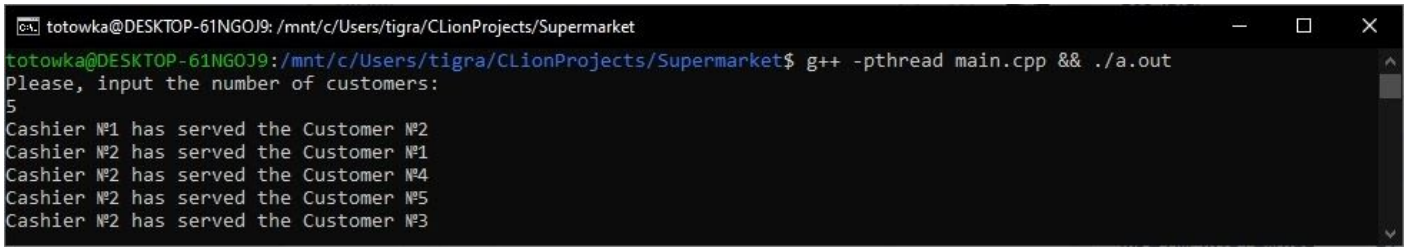
Пример 1: Некорректное (отрицательное) значение количества клиентов. Вывод сообщения об ошибке.

Пример 2: Некорректное (строковое) значение количества клиентов. Вывод сообщения о том, что магазин сегодня не работает.

Пример 3: Ввод количества клиентов равному 0. Сообщение о завершении работы магазина на сегодня.

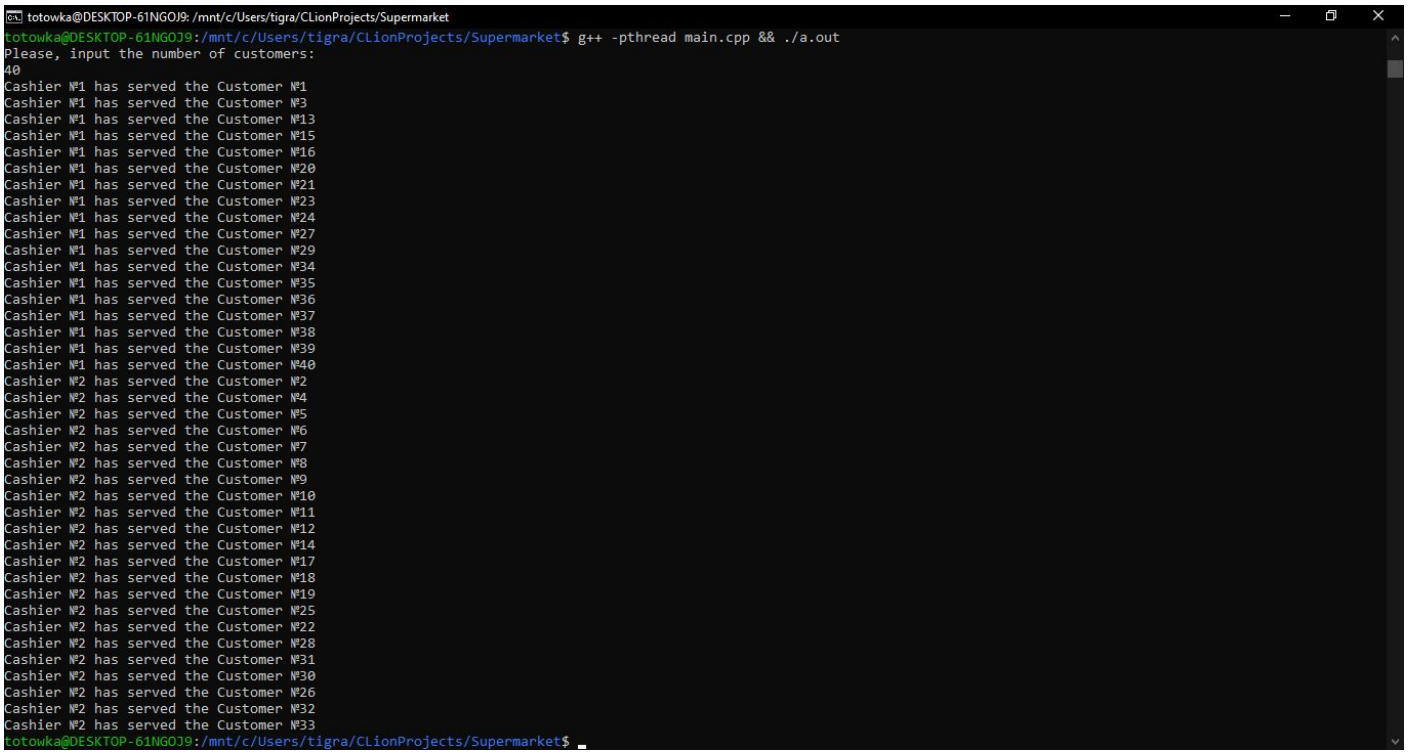


Пример 4: Ввод количества клиентов равного 5. Разделение клиентов по потокам и их обслуживание.



Пример 5: Повторный ввод количества клиентов равного 5. Разделение клиентов по потокам и их обслуживание. Заметим, что клиенты разделились по другим очередям и обслуживались отлично от того, как это было в прошлом примере.

Пример 6: Количество клиентов равно 40. Клиенты распределились по очереди и были обслужены кассирами.



Пример 7: Количество клиентов равно 40. Клиенты распределились по очереди в другом порядке и были обслужены кассирами.