Текстовый вариант видеоурока из предыдущего шага

Пришло время поговорить о каналах в языке Go. Смотрите, как мы сказали ранее: горутина у нас стартует с отдельным stack-ом. То есть если, например, у нас есть stack вызовов main(), потом вызывается функция foo(), а foo() вызовет в горутине функцию bar, то у функции bar() будет отдельный свой stack. Таким образом, в языке горутины сделали достаточно изолированными сущностями. И пришлось придумать механизм, который позволит разным горутинам синхронизироваться друг с другом и строить какое-то общение. Для этих целей были созданы такие типы данных как каналы.

Для того чтобы в Go создать канал, нужно воспользоваться конструкцией типа данных chan. То есть мы можем объявить нашу переменную с типом данных chan и создать при этом канал, который позволит нам общаться между разными горутинами. Давайте посмотрим на примерах, как с этим можно работать.

Разберем функцию readChan, которую я написал, и как мы видим, она как параметр получается тип переменной chan. И мы говорим, что это канал, состоящий из int-а. То есть в данной переменной ch, которая является каналом, должны передаваться (НРЗ 01:37).

Смотрите, что мы делаем в этой функции: в Go есть такой оператор «<-ch», что значит, что мы читаем из канала. То есть данная операция достает из нашего канала значение и помещает его в эту переменную. После того, как мы достали значение, – мы просто выводим это на экран.

Что происходит в функции main()? По традиции мы здесь просто выводим, что стартуем функцию main(), дальше мы объявляем нашу переменную ch как chan из int-ов, то есть это канал, который содержит внутри int. И здесь с помощью уже знакомой нам функции make мы создаем канал.

Здесь есть 2 варианта создания каналов: каналы могут быть буферизированные и небуферизированные. В данном случае мы создали небуферизированный канал – об этом чуть позже я расскажу. Что это значит? Это значит ровно то, что ёмкость у этого канала никакая. То есть, по сути, у неё нет ёмкости, и она не может в себе содержать много значений.

Как это отражается на нашей программе? Смотрите, дальше мы в горутине вызываем нашу функцию readChan и передаем туда наш канал. И как мы видели, в нашей функции происходило чтение путем помещения оператора «<-ch». А здесь мы помещаем оператор «ch <-», что значит: «запись в канал». То есть в данной строчке мы число 77 кладём в канал, после этого мы выводим, что функция main завершилась. Давайте запустим нашу программу и посмотрим на результат. И всё, на самом деле, отработало, как и планировалось: мы сначала стартанули main(), потом в горутине у нас вывелся канал значение 77, и потом мы завершили нашу программу.

Какие здесь есть нюансы? Смотрите, на самом деле, чтение из канала в данном случае является блокирующим, что это значит? Когда мы запускаем эту горутину, в ней происходит чтение из канала ch, но на этот момент в канале ещё никаких данных нет, потому что мы ещё не положили никаких данных туда. И, на самом деле, эта горутина заблокируется на этой точке и будет ждать, когда в наш канал прилетят какие-то данные. Как только мы кладем в канал какие-то данные – они будут прочитаны в горутине, и произойдет вывод.

Давайте для примера сделаем следующим образом, чтобы было наглядно: допустим, я сделаю здесь Sleep на 1 секунду. Посмотрим, что мы будем видеть в итоге. Как мы видим, мы сначала запустили START MAIN, но горутина ничего не печатала, пока у нас не прошёл Sleep, и мы не положили что-то в канал. После того как в канале что-то появилось, горутина уже в этот момент напечатала нам

77. Здесь вывод немножко произошёл раньше по времени, то есть мы сначала успели напечатать эту строку, потом горутина перед тем как main() завершился, успела напечатать эту строку, поэтому порядок немножко изменился. Но на самом деле и запись, и чтение также блокирует канал, то есть операция записи тоже является блокирующей. Если мы запишем в канал, и никто из него не будет читать, то в этой точке мы тоже заблокируемся.

Для наглядности можно сделать следующим образом: давайте я здесь запишу в канал цифру 100, но смотрите, горутина, которая читает из канала, запустится после того, как я положил значение в канал. Как мы сказали, при записи в канал мы тоже блокируемся, поэтому здесь программа заблокируется, и до выполнения горутины код не дойдет. В этом случае, на самом деле, мы получим ошибку, потому что у нас будет deadlock! Как мы видим, язык… рантайм языка тоже ругается. Почему он ругается? – Потому что, на самом деле, мы здесь будем бесконечно висеть и ждать, пока кто-то прочитает наш канал, но читателей у нас канала нет, поэтому висеть мы будем бесконечно.

Данная вещь очень опасна в горутинах, потому что если мы это будем делать в горутине, то ошибка рантайма не появится, и у нас горутины будут висеть, по сути, заблокированные и ничего не делать. Давайте попробуем воссоздать такую ситуацию, как она может быть. Допустим, мы ничего не будем записывать в наш канал. При этом запуск горутины приведет к тому, что горутина здесь заблокируется и будет ожидать, пока кто-то что-то напишет в канал. Но как мы видим, наша программа запустится в обычном режиме, отработает и завершится.

На протяжении работы функции main() мы так и не узнаем, что у нас горутина readChan висит заблокированная. Если мы так запускаем очень много горутин – они будут забивать память и висеть в заблокированном виде. И эта ситуация называется «утечкой горутин», потому что горутины будут блокироваться, в них

есть deadlock-и! Но существуют при этом горутины, которые запущены и что-то делают, например, у нас функция main() продолжают нашу работу. И эти горутины – они навсегда заблокированы, поэтому эта ситуация называется «утечкой горутин».

В чём же тогда отличие буферизированных каналов от небуферизированных? Для того чтобы с этим разобраться, давайте сначала поймем, как создавать буферизированные каналы. Небуферизированные мы теперь знаем, а буферизированные создаются с помощью передачи в функцию make ещё одного значения, а именно размера нашего канала. Мы можем передать 1-2, любой размер, который нам нужен.

Давайте создадим канал с буфером 1, что это значит? – То, что мы можем в этот канал записать значения, при этом не блокироваться. В прошлый раз мы пытались записать в канал ch значение 100 до того, как вызвали горутину. И при этом мы блокировались и получали deadlock! Давайте попробуем сделать то же самое, но теперь с буферизированным значением, у которого размер 1.

Запишем значение 100, потом вызовем нашу горутину и запишем значение 200 после вызова нашей горутины. Давайте запустим и посмотрим, что произойдет. Смотрите, как обычно, мы стартовали нашу main() функцию, потом записали в канал значение 100, при этом не заблокировались. У нас произошёл запуск функции readChan, мы прочитали значение из канала и вывели его на экран. После этого мы также записали значение в канал, но не заблокировались и вывели окончание нашей функции. При этом последнее значение так никто и не прочитал.

Но как только мы выйдем за рамки нашего размера, мы будем иметь ту же ситуацию, как и с небуферизированным каналом. То есть если мы попытаемся до

чтения из нашего канала положить туда 2 значения, то на 2-й записи в канал мы заблокируемся, потому что размер нашего канала всего лишь 1. Давайте запустим и посмотрим, что у нас получился deadlock! Вот мы видим, что сработал deadlock! потому что на данном этапе мы заблокировались. И мы можем это сделать либо после чтения, чтобы не заблокироваться, либо увеличить размер нашего канала на 2, например, и избежать при этом deadlock!