```
// Verilog

// >>>>>> Модули
// Модуль - основной элемент программы на Verilog.
// Модуль, по смыслу, схож с описанием класса.
// Он, так же, имеет в составе себя как конструкции, отвечающие за хранение информации,
// так и конструкции отвечающие за её обработку
// Кроме того, у модуля есть заголовок, в котором описываются интерфейс, по которому он может общаться с внешним миром.


// Объявление модуля

module (  // Заголовок
        Input       a,          // Входной порт a
        Output      b,          // Выходной порт b
        Input [15:0] c          // Входной порт c. 16-и разрядный.
);

// TODO: Ваш код

endmodule


// Ещё одн пример объявления модуля

module (  // Заголовок
        Input       a,          // Входной порт a
        Output      b,          // Выходной порт b
        Input [15:0] c          // Входной порт c. 16-и разрядный.
);

// TODO: Ваш код

endmodule


// >>>>>> Линии
// Переносят сигнал, но не хранят его. Из Высокоуровневых языков напоминают свойства,
// которые, так же, не хранят информацию сами, а всего лишь возвращают значение переменной или результат работы функции.
// Значения свойствам могут присваиваться как при объявлении, так и после, при помощи оператора assign.
// Передают значение, которое может быть одновременно представлена и как массив бит и как целое число.
// Компилируются в соеденительные линии и коммутаторы.

// Объявление линии.

wire line1;             // Одноразрядная линия. Объявление без присваивания.
wire [7:0] line2;       // Восьмиразрядная линия. Объявление без присваивания.
wire alalalalal = line1 && line2[0] || line2[3];    // Объявление и присваивание значения одноразрядной линии.
wire [5:0] ololo [0:31];      // Массив из 32-х 6-ти разрядных линий.
```

```
assign line1 = 1;                                                    // Присваивание значения линии line1
assign line2 = 8'b01001001 + 2'h3F;                                  // Присваивание значения линии line2
assign ololo[0] = (line1)?6'b010101:{1'he, 2'b00};     // Присвоение 0-му элементу массива значания, вычисляемого условным оператором.  {...,...} - Операция конкатинации

// >>>>>> Регистры
// Хранят информацию. Могет быть присвоены только внутри блока always
// Компилируются  в триггеры и регистры

// Объявление регистра

reg r1;                          // 1-но разрядный регистр. (Один триггер)
reg [9:0] r2;                    // 10-ти разрядный регистр.
reg [31:0] rororo [0:127];       // Массив из 128-ми 32-х разрядных регистров.

// >>>>>> always
// Блоки always представляют собой основные функциональные элементы в модуле.
// Все вычисления, сделанные при помощи логических функций, фуксируются в этом блоке,
// путём сохранения в регистры значений линий, других регистров, логических функций от линий и регистров

// Происходит всегда. Постоянно. 24 часа в сутки, 7 дней в неделю, каждый момент времени.
// Вследствие чего, после компиляции, выраждается в конструкции: wire r1 = 1'b1;
always @* begin
        r1 <= 1'b1;
end



// Будет запускаться по положительному фронту сигнала a.
// a - автоматически интерпритируется, как тактовый сигнал. И всё построение модуля делается изходя из этого.
always @(posedge a) begin
        // Условный оператор, проверяющий факт выставления сигнала сброса
        if (!rst) begin
                // Синициализация начальных значений регистров (reg) пишется тут
        end else begin
                // Тут прочая логика
        end
end
```