Дано:

uint16\_t a = 0xFFFF;

uint16\_t b = 0;

b = ++a + a++;

Чему равно b после выполнения вышеуказанного участка кода?

Чему равно a после выполнения вышеуказанного участка кода?

Дано:

for (int i = 0; i < 10; i += 2)

printf(“%d”, i);

Какие данные выведет программа после выполнения вышеуказанного участка кода?

**На выбор любые 3:**

Задача на самостоятельное выполнение 1:

Дан массив фиксированной длины 1000 из целых чисел. Заполнить числами от 1 до 1000 таким образом, чтобы сначала шли все четные, а затем все нечетные. Написать работающую программу на языке C.

Задача на самостоятельное выполнение 2:

Написать работающий пример программы на языке C с использованием volatile и пояснить для чего это требуется в данном случае.

Задача на самостоятельное выполнение 3:

Написать работающий пример программы на языке C с использованием динамической памяти из кучи.

Задача на самостоятельное выполнение 4:

Написать работающий пример программы на языке C для вычисления факториала числа оптимальным методом на усмотрение программиста, пояснить причину выбора этого метода.

Задача на самостоятельное выполнение 5:

Написать работающий пример программы на языке C для сортировки массива случайных чисел (массив фиксированной длины) от большего к меньшему методом пузырька.

Задача на самостоятельное выполнение 6:

Написать работающий пример программы на языке C содержащий программную реализацию кольцевого буфера фиксированной длинны и привести пример его использования.

1. Переменная a будет равен 1, тк переполнение беззнаковой величины вполне однозначно, а вот с переменной b не однозначно. Два инкремента в одной строке являются undefined behavior, если точнее, то unsequenced expressions modifications.

Для примера у меня при использовании MinGW b равняется 1, а на Clang-cl – 0.

2. Будет напечатано «02468»

Задача 1. приложенный код

Задача 2. классическим примером использования volatile можно считать применение в контексте прерываний у микроконтроллеров. Например:

|  |
| --- |
| Bool a = true;  void handler1(void)  {  if(a){...}  } |
| void handler2(void)  {  a = false;  } |

Код такого типа может работать некорректно, тк в handler1 компилятор может оптимизировать проверку if(a), посчитав что a не меняется, а значит условие всегда правдиво, а значит код в условии будет выполняться в не зависимости от вызова handler2

Задача 4. приложенный код.

Реализовано частично, так как uint64\_t не вмещает больше 20!. Для подсчета больших значений можно подключить модуль вычислений с произвольной разрядностью (например GMP), либо реализовать такой модуль самостоятельно, либо пожертвовать точностью и перейти к вычислению с плавающей точкой.

Задача 5. приложенный код

Задача 6. <https://github.com/4rturO/DiscoveryStm32f4-AD7705> в этом проекте есть пример использования для отправки сообщения.

Пример использования: несколько событий генерируют отправку к верхнему модулю одноуровневых по приоритету сообщений, то их проще отправить в буфер из которого по готовности эти сообщения будут отправляться дальше