1. Когда бы ты предпочел использование enum вместо bool/int?

Ответ: Когда нам необходимо в функции возвращаться не только true/false, а какие-то значения из списка. Например, можно создать перечисление enum с кодами ошибок и в функции “enum название\_перечисления название\_функции (передаваемые переменные)” возвращать коды ошибок. Плюс перечисления хорошо документируют мой код (в случае bool и int придется написать комментарии того, что возвращает функция)

1. Как в языке Си можно получить доступ к биту данных?

Ответ: Использовать битовые поля, которые представляют собой структуру, содержащую информацию о полях регистров по битам (с различной битовостью 1, 2, 4 бита и т.п.). Опасность битового поля заключается в последовательности следования битов в различных система (big и little endian)

1. Что такое объединения union? Где ты видишь их применение?

Ответ: Объединение – это тип данных, хранящий определенный набор элементов. Отличие от структуры в том, что все элементы объединения занимают один и тот же участок памяти. Мы можем инициализировать только первый элемент объединения. Но изменение одного элемента объединения, приводит к изменениям другого эл-та. Применять объединения можно, например, в случае, когда необходимо одной командой сбрасывать все элементы структуры внутри этого объединения, это могут быть и другие типы данных, указатели и так далее.

1. На какие секторы делится программа Си?

Ответ: .text – код программы, хранящийся во FLASH памяти.

.data – инициализированные данные, хранящиеся во FLASH памяти, при запуске копируются в SRAM (значения в глобальной области, но не **const,** а также static данные в локальной области)

.bss – неинициализированные данные, хранящиеся в SRAM

1. Как узнать, сколько FLASH-памяти занимают ненулевые данные в программе?

Ответ: посмотреть, сколько данных сгенерировалось в секции text и data, их сумма будет составлять размер прошивки в бинарном виде

1. Как узнать минимальный потребляемый программой объем ОЗУ?

Ответ: посмотреть, сколько данных занимают .bss и .data. Меньше, чем сумма данных, занимаемая этими секциями, в SRAM быть не может

1. Для чего используется в программе стек? А Куча?

Ответ: Оба сектора являются динамическими и могут при определенных обстоятельствах пересечься, чего допускать нельзя. Можно смотреть на схему распределения данных в SRAM.

.heap – куча и .stack – стек, хранятся в SRAM. Стек используется для вызова функций и его используют как место для переменных, его переполнение приведет к невозможности вызова этих функций.

При каждом вызове новых функций на стек кладется новая запись – стековый фрейм.

При объявлении локальной переменной, она создается на вершине стека. При выходе из функции компилятор снимает со стека ее локальные переменные, в результате чего на вершину поднимаются значения внешней области видимости.

Переполнение стека – ошибка, возникающая при полном заполнении стека.

В куче же выделение памяти требует вызова специальных функций (malloc, calloc и т.п.). Для доступа к содержимому кучи надо использовать указатели.

1. Что такое статическое выделение памяти и динамическое выделение? Назови преимущества и недостатки каждого

Ответ: Динамическое выделение памяти – это резервирование места в куче с помощью специальных функций malloc, calloc и другие. Они принимают размер в байтах того, сколько требуется выделить памяти. Минус динамического выделения в том, что нужно следить за освобождением памяти, не допускать двойного освобождения памяти и в целом внимательно контролировать процесс во избежание пересечения стека и кучи.

Статическое выделение памяти – автоматическое выделение памяти в стеке, память освобождается при выходе из фигурных скобок. Минус в том, что можно зарезервировать больше памяти, чем необходимо. Во встроенных системах чаще всего используют именно статическое выделение памяти для избегания переполнения стека.

1. Какие опасности могут быть при работе с динамически выделенной памятью?

Ответ: Нужно освобождать память; проверять указатель на NULL; не допускать двойного освобождения памяти – освобождение висячего указателя (неопределенное поведение)