1. Почему нужно избегать объявления функций без аргументов, когда она не принимает никаких аргументов: void fn\_name() { … } вместо void fn\_name(void) { … } ?

Ответ: чтобы обозначить то, что функция не принимает никакие аргументы. Если не написать void, то можно передать в функцию аргументы и не получить ошибку об этом.

1. Какие методы передачи значений в функцию ты знаешь? В чем их отличия?

Ответ: Передача по значению – копирование значение аргументов в другую функцию (передаем не переменную, а ее копию). По выходу из функции значения переменных в функции, откуда они копировались, останутся прежними

Передача по ссылке – передается адрес переменной (&x – разыменовывание) в функцию, которая ссылается на эту переменную (\*x). Таким образом, в переменной по адресу x, записываем результат выполнения функции

1. Изучи код. В чем его проблема? Какой результат будет выведен? Постарайся объяснить, в чем его проблема

#include <stdio.h>

#include <string.h>

char \*get\_result\_str(int code)

{

char res\_str[4] = {0};

switch (code) {

case 0:

strcpy(res\_str, "OK");

break;

case 1:

strcpy(res\_str, "ERR");

break;

default:

strcpy(res\_str, "UNK");

}

return res\_str;

}

int main(void)

{

printf("Result: %s", get\_result\_str(0));

return 0;

}

Ответ: Результат выполнения данной программы дает: “Result: (null)”. Функция возвращает адрес локальной переменной. Нужно переопределить переменную res\_str[4] в глобальной области видимости. Тогда результатом будет вывод строки “OK”.

1. Что такое условная компиляция? Для чего она применяется?

Ответ: Условная компиляция – это предварительная обработка текста всей программы. Включаются файлы в текст программы, происходит проверка #define программы с помощью директив #ifndef, #else и #endif, которые проверяют, был ли ранее произведен #define в коде и компилирует либо код после #ifndef (если #define есть), либо код после #else (если #define не был)

1. Что такое рекурсивное (многократное) включение заголовка .h? Каким образом можно защититься от рекурсивного включения?

Ответ: Многократное включение заголовка .h – это подставление в одном файле одного и того же заголовочного файла 2 или более раз. В таком случае может произойдет объявление 2 и более прототипов функции и возникнет ошибка компилятора.

Избежать рекурсивного включения можно с помощью с помощью директивы #ifndef – если заголовочный файл не #define’н, то #define filename.h, в другом же случае define файла не произойдет. Закрываем код с помощью директивы #endif.

1. Зачем применяется ключевое слово static для функций?

Ответ: Ключевое слово static для функции применяется для функций, которые применяются только в конкретном файле исходного кода (.c). По умолчанию область видимости функции – extern. Это надо помнить

!!!В заголовочных файлах никогда не пишется static ни к функциям, ни к переменным, так как заголовочные файлы могут многократно вставляться в разные файлы

1. Как ты думаешь, что такое модуль программы Си?

Ответ: Модульность подразумевает написание файла исходного кода .c и его одноименного заголовочного файла .h, которые связана между собой по смыслу, т.е. выполняют общую функцию. Например, заголовочный файл и файл исходного кода для работы с BMS в проекте по Black Pill.

1. Каким правилам следует придерживаться при построении многофайловой (многомодульной) программы Си?

Ответ: Общий нейминг заголовочного файла и файла исходного кода, помнить про условную компиляцию (использовать директивы для избежания ошибок), static и extern во избежание ошибок, удобная сортировка файлов в проекте (в среде разработки),