1. **Дайте определение символьному типу данных.**

char - single-byte character (стандарт)

Символьный тип (char) — тип данных, предназначенный для хранения одного символа (управляющего или печатного) в определённой кодировке.

1. **Ложен ли символ конца файла? Какое макроопределение соответствует символу конца файла?**

Символ конца файла имеет негативное значение типа int -> не ложен

Макроопределение символа конца файла = EOF

1. **Какой тип имеет символ конца файла?**

Тип int

1. **Ложен ли символ терминального нуля? Если ложен, то только ли он? Каково его представление в памяти?**

Да, ложен, так как представлен в памяти, как 0. Ложен только он

1. **Накладывает ли размер символа какие-либо ограничения на таблицы кодировки?**

Да, количество символов, которые могут быть закодированы не может превышать 2^(размер одного символа)

1. **Какие таблицы кодировки Вам известны? Какие из них — однобайтные?**

Основные таблицы кодировки: ASCII (ASCII, Windows-1251), UNICODE (UTF-8/16/32)

Однобайтные кодировки: ASCII, Windows-1251, UTF-8

1. **Можно ли делать какие-либо предположения о положении тех или иных символов в таблице кодировки?**

Нет, нельзя

1. **Дайте определение строке, оканчивающейся нулём.**

Будем называть оканчивающейся нулём строкой (англ. null‐terminated string) массив символов, последним из которых является символ терминального нуля — символ '\0'.

1. **Поясните, как работают договорённости сообщества на примере договорённости о терминальном нуле. Какие выводы Вы можете сделать из определения строки?**

Библиотеки в языке и работа с типом данных реализуется на основе этих договоренностей

Из определения строки можно сделать вывод, что определение строки в языке Си основано на договоренности о терминальном нуле

1. **Почему рекомендовано использовать функцию fgets при работе с stdin, а не функцию gets? Каким образом функции выводятся из стандарта, зачем нужно в некоторых языках ключевое слово deprecated?**

Рекомендовано использовать функцию fgets, так как она безопасна по сравнению с функцией gets. fgets считывает символы на 1 меньше заданной длины. Это помогает избежать переполнения буфера

Ключевое слово deprecated используется в некоторых языках, чтобы указать на функции, которые являются устаревшими. Эти функции не удаляются из стандарта, чтобы можно было поддерживать старый код

1. **Предложите свою реализацию функции strlen.**

funcs.c

1. **Предложите свою реализацию функции strcpy.**

funcs.c

1. **Валиден ли вызов функции strcpy(str, str + 1)?**

Нет, так как в качестве параметров в функцию передаются строки с параметром restrict. Это значит, что при таком вызове будет undefined behavior

1. **Дайте определение лексикографическому порядку на множестве строк.**

Лексикографический порядок: 1) сначала сравнивается длина строк, у которой длина меньше -> меньше; 2) если длины строк совпадают, то сравниваются коды символов поочередно

1. **Предложите свою реализацию функции strcmp.**

funcs.c

1. **Предложите свою реализацию функции strcat.**

funcs.c

1. **Валиден ли вызов функции strcat(str, str)?**

Нет, так как в качестве параметров в функцию передаются строки с параметром restrict. Это значит, что при таком вызове будет undefined behavior

1. **Предложите свою реализацию функций strchr и strrchr. С чем может быть связано наличие в стандартной библиотеке двух функций?**

funcs.c

Чтобы искать вхождение символов с двух сторон

1. **Существуют ли аналоги функций для работы со строками, безопасные относительно длины строки?**

Да, существуют. Например, strcpy -> strncpy; strcat -> strncat и т. д.

1. **В каких случаях Вы стали бы использовать безопасные аналоги?**

Безопасные аналоги используются, когда строка вводится пользователем

1. **Почему стандартные функции по работе со строками работают с символами, принимаемыми в виде выражений типа int, а не char?**

Это нужно для совместимости при работе с EOF. EOF имеет значение -1, которое не поддерживается типом char. Из-за этого будут возникать проблемы при работе с файлами

1. **Предложите свою реализацию функции strtok, исходя из предположения, что Вы можете пользоваться в данном случае глобальными переменными.**

funcs.c

1. **Изменяет ли strtok передаваемые данные? Какие из трёх «строк» можно передавать в strtok:**

**char \*s1 = "beebos";**

**char s2[] = "bobos";**

**char s3[256] = "seeleebobos";**

strtok изменяет передаваемые данные, вставляя терминальные нули вместо символов разделения

в strtok можно передавать только s2 и s3

1. **Равны ли p1 и p2?**

**char \*p1 = strchr("mama", ‘a’);  
char \*p2 = strchr("mama", ‘a’);**

Мы не можем гарантировать, что две строки “mama” будут указывать на одинаковые участки памяти, а так как strchr возвращает адрес, то не можем гарантировать, что p1==p2

1. **Предложите свою реализацию функции split, возвращающей массив слов, составляющих исходную строку. Сравните функции split и strtok.**

funcs.c

split возвращает массив слов, а strtok - только одно слово. Обе функции принимают саму строку и разделители для этой строки, но из-за реализации strtok при разделении строки на слова можно передавать различные разделители на каждом этапе -> будем получать совершенно различные слова. В split разделители передаются один раз и строка делится по ним только один раз однозначно. Также split выдает нам весь массив слов сразу, в отличие от strtok, где мы можем получить только необходимое нам количество строк и не занимать лишнюю память

1. **Предложите свою реализацию функции join, возвращающей строку, составленную из элементов массива строк. Будете ли Вы контролировать превышение длины строки или возложите ответственность на вызывающую сторону? Почему? Как это связано с безопасными относительно длины строки функциями?**

funcs.c

Я контролирую превышение длины строки, так как чтобы проверить, что все выполнено корректно, пользователю нужно выполнить n действий. Также выполнено условие безопасности относительно длины строки, т.е. со строкой полученной в итоге можно будет безопасно работать, так как не будет переполнения буфера, а также это гарантирует, что будет добавлен символ конца строки в конец функции

1. **Назовите особенности реализации функций с данными, сохраняющимися между вызовами, на примере функции strtok. В качестве ориентира можете воспользоваться указаниями в стандарте насчёт связи реализации функции strtok и реализации других функций по обработке строк.**

Такие функции сохраняют положение указателей на внешние переменные. Из-за этого реализация таких функций должна быть такой, будто ее не вызывает никакая другая функция, потому что в ином случае при работе пользователя пользователя с функцией strtok может быть неопределенно, так как глобальный указатель может уже указывать на какие-то участки памяти. Поведение будет не определенно

1. **Пусть строка в стиле языка Паскаль — массив символов, в котором на первой позиции хранится длина строки в виде символа. Как будут выглядеть аналоги всех функций по обработке строк для этой структуры данных?**

Сначала всегда будет считывать первый символ и запоминаться, как длина строки, а остальные будут считываться пока индекс не достигнет значения длины строки, а не пока она встретит терминальный ноль, как при обработке строк в стиле Си.

1. **Сравните строку, оканчивающуюся нулём, и строку в стиле языка Паскаль.**

Они имеются одинаковую длину. В строке в стиле Паскаль нет явного разделителя строк и явного признака указания на ее длину. Если в строке в стиле Си нам не нужно было знать длину строки при работе с ней, то при работе со строкой в стиле Паскаль нужно будет всегда пересчитывать длину строки, чтобы записать ее

1. **Дайте определение модулю. Чем различаются интерфейс и имплементация модуля?**

Модуль - это отдельно разрабатываемые и поддерживаемые части основной реализации программы. Их можно заменять, отдельно тестировать.

Интерфейс модуля описывает функциональность модуля. Указывает, какие функции и методы доступны пользователю

Имплементация представляет собой реализацию всех описанных в интерфейсе функция

1. **Как осуществляется защита заголовочного файла от многократного включения? Необходима ли она?**

Оно осуществляется с помощью конструкции include guard (макросы #ifndef, #define, #endif)

Это обязательно, так как иначе в многофайловом проекте может возникнуть конфликт при компиляции программы, так как один и тот же файл будет включен несколько раз и произойдет переопределение одних и тех же структур и функций

1. **Защита заголовочного файла является стандартной процедурой или опирается на договорённости сообщества?**

Это стандартная процедура для устранения ошибок компиляции