**Защита лабораторной работы №4**

1. Символьный тип данных – простой тип данных, множество значений которого является подмножеством целых неотрицательных чисел и включает нуль.
2. EOF (end of file), определён в stdio.h, не является ложным (так как в основном реальное значение -1)
3. Ложен, всегда, только он, представление в памяти - просто 0
4. Да, накладывает, зависит от памяти, выделяемой под каждый символ; с char можно использовать таблицы кодировки с 256 символами
5. ASCII - однобайтовая; UTF8 - от 1 до 4 Б, UTF-16: 2 - 4 Б, UTF-32: 4 Б, Unicode: 2 Б
6. Делать какие-либо предположения о положении тех или иных символов в таблице кодировки нельзя, потому что тогда мы перекладываем ответственность на составителей компилятора (чтобы определить чем является символ и какими св-ми он обладает, надо использовать макросы из <ctype>: isdigit, isalpha)
7. Строка, оканчивающаяся нулем - массив символов с нулем в конце (всегда)
8. Все обязаны ставить 0, поэтому все могут на него опираться. Вывод из определения строки: она всегда заканчивается нулем, и поэтому ее длина нам не нужна при их обработке
9. Брешь такой. Вжух. И написал. Очевидно же
10. Брешь такой. Вжух. И написал. Очевидно же
11. Все хорошо. Рестрикт роли не играет.
12. Лексикографический порядок — как в телефонной книге.

(отношение линейного порядка на множестве слов над некоторым упорядоченным алфавитом)

1. Брешь такой. Вжух. И написал. Очевидно же
2. Брешь такой. Вжух. И написал. Очевидно же
3. Невалиден, потому что restrict
4. char \*strchr(const char \*s, int c)

char \*strrchr(const char \*s, int c)

Функция strchr находит *первое*, а strrchr - *последнее* вхождение *c* (преобразованное в символ) в строке, на которую указывает s. Завершающий нулевой символ считается частью строки. Возвращает указатель на расположенный символ или нулевой указатель, если символ не встречается в строке.

1. У многих функций существуют аналоги, безопасные относительно

переполнения буфера. Все они имеют суффикс «n» в названии, например, strcpy/strncpy.

Более безопасными функции сами по себе не являются — при использовании безопасного варианта вызывающая сторона просто переносит ответственность на себя со стороны, передающей данные.

1. Когда рационально запомнить длину строки
2. Брешь такой. Вжух. И написал. Очевидно же
3. 1 – указатель на строковый литерал – НЕВАЛИДНО

2 – массив символов – ВАЛИДНО

3 – массив символов фиксированной длины - ВАЛИДНО

1. Коротко: strtok делает из строки порнографию, а сплит сплитит как завещал дедушка питон.

Сплитом пользуемся, когда можно не экономить память. Стртоком наоборот.

Подробно: strtok сначала пробегается по строке и если все символы из массива разделителей то >> NULL. Далее пробегается и ищет в строке символы из массива разделителей. Если не находит, то >> указ. на эту строку, но последующие вызовы >> NULL. Если находит, то заменяет на ‘\0’ и запоминает позицию, с которой будет начинаться поиск следующей части строки, а >> указ. на начало первой выделенной части строки. Чтобы продолжить разбиение, нужно передать в ф-ю NULL и тогда она вытащит из памяти запомненную позицию и начнет разбиение с нее.

1. Используются глобальные переменные
2. Условие выхода из цикла не терминальный ноль, а превышение кол-ва итераций, равного длине строки
3. В Паскалé нулевой символ – длина строки => ограничение на длину строки зависит от таблицы кодировки
4. Модуль — исходный текст программы, который может быть отдельно оттранслирован
5. Чем различаются интерфейс и имплементация модуля?  
   Интерфейс - взаимосвязь программных объектов (модулей, программ, языков и т.п.), обеспечивающая условия их совместного функционирования. В интерфейсе указываются константы, типы, переменные, процедуры и функции, которые могут быть использованы основной программой (модулем) при вызове этого модуля.  
     
   Имплементация (исполнительная часть) включает все подпрограммы модуля. Она может также включать локальные метки, константы, типы и переменные, недоступные для других программных единиц (естественно, и для интерфейса самого модуля). Это "внутренняя кухня" модуля, где протекают процессы, о которых не надо "знать" другим программным единицам.
6. Защита заголовочного файла от многократного включения  
   Для того чтобы защититься от повторного включения используется конструкция, которая называется «include guard».  
   [#ifndef](https://vk.com/im?sel=284644018&st=%23ifndef) \_GRANDFATHER\_H\_  
   [#endif](https://vk.com/im?sel=284644018&st=%23endif)  
   Для правильной работы «include guard» необходимо позаботится о том, чтобы имена (макросы процессора), которые используются в этой конструкции были уникальны.  
   Защита нужна, если в программе больше 1 модуля.
7. Защита заголовочного файла является стандартной процедурой или опирается на договорённости сообщества?  
   По-моему это масло масленое. Но предположим, что стандартной.

Строковый литерал – последовательность символов, заключенных в двойные кавычки. [c99 6.4.5]

Строковый литерал рассматривается компилятором как массив элементов типа char. Когда компилятор встречает строковый литерал из n символов, он выделяет n+1 байт памяти, которые заполняет символами строкового литерала и завершает нулевым символом.

* Массив, который содержит строковый литерал, существует в течение всего времени выполнения программы.
* В стандарте сказано, что поведение программы не определено при попытке изменить строковый литерал.
* Обычно строковые литералы хранятся в read only секции.

Строка s1 меньше строки s2, если выполнено любое из двух условий:

* первые i символов строк s1 и s2 одинаковы, а символ s1[i+1] меньше символа s2[i+1] (пример, “abc” < “abd” или “abc” < “bcd”);
* все символы строк s1 и s2 одинаковы, но строка s1 короче строки s2 (пример, “abc” < “abcd”).

Функция strcmp сравнивает символы, сравнивая значения кодов, которые представляют эти символы.

* int fputs(const char \* restrict s, FILE \* restrict stream);
* char \*fgets(char \* restrict s, int n, FILE \* restrict stream);
* char \*strcat(char \* restrict s1, const char \* restrict s2);
* char \*strncat(char \* restrict s1, const char \* restrict s2, size\_t n);
* int strcmp(const char \*s1, const char \*s2);
* int strncmp(const char \*s1, const char \*s2, size\_t n);
* char \*strchr(const char \*s, int c);
* char \*strrchr(const char \*s, int c);
* size\_t strcspn(const char \*s1, const char \*s2);
* char \*strpbrk(const char \*s1, const char \*s2);

находит первое вхождение в строке, на которую указывает s1, любого символа из строки, на которую указывает s2.

* char \*strtok(char \* restrict s1, const char \* restrict s2);
* size\_t strlen(const char \*s);
* char \*strcpy(char \* restrict s1, const char \* restrict s2);
* char \*strncpy(char \* restrict s1, const char \* restrict s2, size\_t n);
* char \*strchr(const char \*s, int c);
* char \*strrchr(const char \*s, int c);