🎯 ШПАРГАЛКА ПО СТРОКАМ В C

1. БАЗОВЫЕ ПРИНЦИПЫ

```
c
// Строка — массив char с окончанием \0
char str1[10] = "Hello";    // Можно изменять
char *str2 = "World";       // НЕЛЬЗЯ изменять!

// Важно: всегда есть \0 в конце
// 'H','e','l','l','o','\0'
```

2. ВВОД/ВЫВОД СТРОК

Способ	Плюсы	Минусы	Пример
`scanf("%s", str)`	Простота	Опасен, нет пробелов	`scanf("%s", name)`
`fgets(str, size, stdin)`	Безопасен	Считывает `\n`	`fgets(buf, 100, stdin)`
`gets(str)`	Удобен	ОПАСЕН! Не использовать!	-

```
c
char name[50];
printf("Введите имя: ");
fgets(name, sizeof(name), stdin);
// Удаляем \n
name[strcspn(name, "\n")] = '\0';
```

3. ОСНОВНЫЕ ФУНКЦИИ <string.h>

Длина строки

```
c
size_t len = strlen(str); // Длина БЕЗ \0
```

Копирование

```c
char src[] = "Source", dest[20];
strcpy(dest, src);       // ОПАСНО!
strncpy(dest, src, 19);  // Безопасно
dest[19] = '\0';         // Гарантия конца
```

Конкатенация

```c
char str1[20] = "Hello";
strcat(str1, " World");     // ОПАСНО!
strncat(str1, " World", 10); // Безопасно
```

Сравнение

```c
int result = strcmp(str1, str2);
// result = 0  – равны
// result < 0  – str1 < str2
// result > 0  – str1 > str2
```

4. ФУНКЦИИ ПОИСКА

strchr - поиск символа

```c
char *pos = strchr(str, 'a');
if (pos != NULL) {
    printf("Найден: %s\n", pos); // "abc" если str="xabc"
    printf("Позиция: %ld\n", pos - str); // 1
}
```

strstr - поиск подстроки

```c
char *pos = strstr("Hello world", "world");
if (pos != NULL) {
    printf("Найдено: %s\n", pos); // "world"
}
```

strpbrk - поиск любого из набора

```c
char *pos = strpbrk("email@domain.com", "@.");
if (pos != NULL) {
    printf("Разделитель: %c\n", *pos); // '@'
}
```

strspn - длина сегмента из разрешенных символов

```
c
size_t len = strspn("123abc", "0123456789");
printf("Цифр в начале: %zu\n", len); // 3
```

5. ФУНКЦИИ ПРЕОБРАЗОВАНИЯ

Из строки в число

```
c
int num = atoi("123");       // 123
float f = atof("3.14");      // 3.14
long l = atol("1000000");    // 1000000

// Более безопасно:
int num;
sscanf("256", "%d", &num);   // 256
```

Из числа в строку

```
c
char buffer[20];
int num = 255;
sprintf(buffer, "Number: %d", num); // "Number: 255"
snprintf(buffer, 20, "Num: %d", num); // Безопасно
```

6. РАЗБИЕНИЕ НА ТОКЕНЫ (strtok)

```
c
char text[] = "apple,banana;orange";
char *token = strtok(text, ",;"); // Первый вызов

while (token != NULL) {
    printf("Токен: %s\n", token);
    token = strtok(NULL, ",;"); // Последующие
}

// ВАЖНО: strtok изменяет исходную строку!
```

7. ПОЛЕЗНЫЕ ФУНКЦИИ <ctype.h>

```
c
isalpha(c)  // буква?
isdigit(c)  // цифра?
isalnum(c)  // буква или цифра?
isspace(c)  // пробельный символ?
toupper(c)  // в верхний регистр
tolower(c)  // в нижний регистр
```

8. БЕЗОПАСНОЕ ПРОГРАММИРОВАНИЕ

❌ ОПАСНО:

```
c
char buf[10];
scanf("%s", buf);           // Переполнение!
strcpy(buf, "Very long");  // Переполнение!
```

✅ БЕЗОПАСНО:

```
c
char buf[10];
fgets(buf, sizeof(buf), stdin);        // Ограниченный
ввод
strncpy(buf, src, sizeof(buf) - 1);    // Ограниченное
копирование
buf[sizeof(buf) - 1] = '\0';           // Гарантия конца
строки
snprintf(buf, sizeof(buf), "%s", src); // Самый
безопасный способ
```

9. ТИПИЧНЫЕ ОШИБКИ

1 Забыли \0:

```
c
char s[3] = {'a', 'b', 'c'}; // НЕ строка!
printf("%s", s); // Мусор или крах
```

2 Переполнение буфера:

```
c
char small[5];
strcpy(small, "Hello World"); // КАТАСТРОФА!
```

3 Сравнение через ==:

```
c
if (str1 == str2) // Сравнивает адреса!
// Правильно: if (strcmp(str1, str2) == 0)
```

4 Изменение строкового литерала:

```
c
char *ptr = "Constant";
ptr[0] = 'c'; // КРАХ программы!
```

10. ШАБЛОН БЕЗОПАСНОЙ РАБОТЫ СО СТРОКАМИ

```
c
#include <stdio.h>
#include <string.h>
```

```
int main() {
    char buffer[100];

    // Безопасный ввод
    printf("Введите строку: ");
    fgets(buffer, sizeof(buffer), stdin);

    // Удаление \n
    buffer[strcspn(buffer, "\n")] = '\0';

    // Безопасная обработка
    char result[100];
    snprintf(result, sizeof(result), "Вы ввели: %s",
buffer);

    printf("%s\n", result);
    printf("Длина: %zu\n", strlen(buffer));

    return 0;
}
```

🎯 КОРОТКИЕ EXAMPLES ДЛЯ ЗАПОМИНАНИЯ

```
c
// 1. Длина строки
char s[] = "Hello";
int len = strlen(s); // 5

// 2. Поиск символа
char *p = strchr(s, 'l'); // указывает на первую 'l'

// 3. Копирование
char dest[10];
strncpy(dest, s, 9);
dest[9] = '\0';

// 4. Конкатенация
strcat(dest, "!");

// 5. Сравнение
if (strcmp(s, "Hello") == 0) {
    printf("Строки равны\n");
}
```

Запомните: Работа со строками в C требует внимательности! Всегда проверяйте границы буферов и не забывайте про `\0`.

Теперь можно переходить к задачам! 🚀