

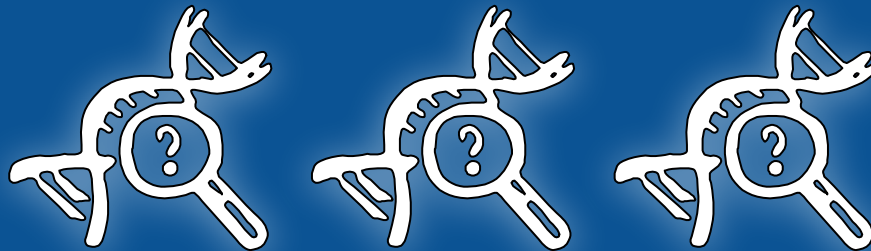


Алгоритмы и Алгоритмические Языки

Семинар #7:

- Представление строк в памяти;
- Ввод строки и переполнение буфера;
- Библиотечные функции работы со строками;
- Пример утилиты для сортировки строк;

Представление строк в памяти



Представление строк в памяти

В языке Си строка – это массив char-ов, заканчивающийся нулевым символом (Null-terminated byte string):

```
const char* str = "Hello, world!\n";  
  
printf("%s", str);
```

Представление строки в памяти:

...	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	...
...	н	е	л	л	о	,		в	о	р	л	д	!	\n	\0	...

Размер строки не учитывает '\0':

```
strlen("Hello, world!\n") = 14
```

Кодировка символов в строке



Символы кодируются значениями char-ов по кодировке ASCII:

dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch	dec	oct	hex	ch
0	0	00	NUL (null)	32	40	20	(space)	64	100	40	@	96	140	60	`
1	1	01	SOH (start of header)	33	41	21	!	65	101	41	A	97	141	61	a
2	2	02	STX (start of text)	34	42	22	"	66	102	42	B	98	142	62	b
3	3	03	ETX (end of text)	35	43	23	#	67	103	43	C	99	143	63	c
4	4	04	EOT (end of transmission)	36	44	24	\$	68	104	44	D	100	144	64	d
5	5	05	ENQ (enquiry)	37	45	25	%	69	105	45	E	101	145	65	e
6	6	06	ACK (acknowledge)	38	46	26	&	70	106	46	F	102	146	66	f
7	7	07	BEL (bell)	39	47	27	'	71	107	47	G	103	147	67	g
8	10	08	BS (backspace)	40	50	28	(72	110	48	H	104	150	68	h
9	11	09	HT (horizontal tab)	41	51	29)	73	111	49	I	105	151	69	i
10	12	0a	LF (line feed - new line)	42	52	2a	*	74	112	4a	J	106	152	6a	j
11	13	0b	VT (vertical tab)	43	53	2b	+	75	113	4b	K	107	153	6b	k
12	14	0c	FF (form feed - new page)	44	54	2c	,	76	114	4c	L	108	154	6c	l
13	15	0d	CR (carriage return)	45	55	2d	-	77	115	4d	M	109	155	6d	m
14	16	0e	SO (shift out)	46	56	2e	.	78	116	4e	N	110	156	6e	n
15	17	0f	SI (shift in)	47	57	2f	/	79	117	4f	O	111	157	6f	o
16	20	10	DLE (data link escape)	48	60	30	0	80	120	50	P	112	160	70	p
17	21	11	DC1 (device control 1)	49	61	31	1	81	121	51	Q	113	161	71	q
18	22	12	DC2 (device control 2)	50	62	32	2	82	122	52	R	114	162	72	r

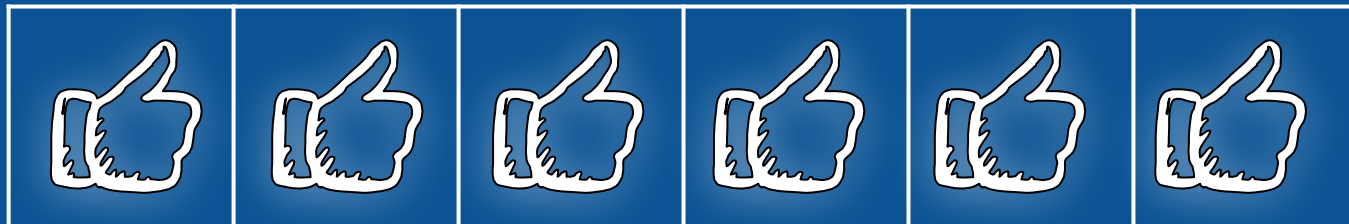


Иные кодировки символов

Иные кодировки:

- Кодировки с немоноширинными символами (multibyte strings);
- Кодировки с широкими символами (wide strings);

Ввод строки и переполнение буфера



Ввод строки от пользователя



Программа ввода строки от пользователя ([07_input_string](#)):

```
#define MAX_ARRAY_SIZE 40U

int main(void)
{
    char input_str[MAX_ARRAY_SIZE];
    scanf("%s", input_str);

    printf("Entered \"%s\"\n", input_str);

    return EXIT_SUCCESS;
}
```

Потенциальные проблемы?

Ввод строки от пользователя

Программа ввода строки от пользователя (чуть получше):

```
#define MAX_ARRAY_SIZE 40U

int main(void)
{
    char input_str[MAX_ARRAY_SIZE];
    int ret = scanf("%s", input_str);
    VERIFY_CONTRACT(ret == 1, "ERROR: expected string\n");

    printf("Entered \"%s\"\n", input_str);

    return EXIT_SUCCESS;
}
```

Потенциальные проблемы?

Защита от переполнения

Программа ввода строки от пользователя (ещё чуть получше):

```
int main(void)
{
    char input_str[MAX_ARRAY_SIZE];

    int ret = scanf("%40s", input_str);
    VERIFY_CONTRACT(ret == 1, "ERROR: expected string\n");

    printf("Entered \"%s\"\n", input_str);

    return EXIT_SUCCESS;
}
```

Потенциальные проблемы?

Детектирование переполнения



Программа ввода строки от пользователя (эквивалентная):

```
#define MAX_ARRAY_SIZE 41U

int main(void)
{
    char input_str[MAX_ARRAY_SIZE];

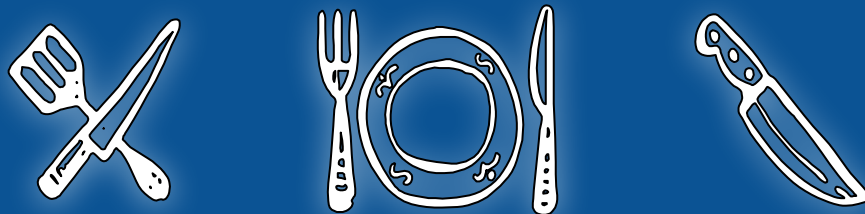
    char* ret = fgets(input_str, MAX_ARRAY_SIZE, stdin);
    VERIFY_CONTRACT(ret != NULL, "ERROR: expected string");

    printf("Entered \"%s\"\n", input_str);

    return EXIT_SUCCESS;
}
```

Потенциальные проблемы?

Библиотечные функции для работы со строками



Функции классификации символов

Вот они: [isdigit](#), [islower](#), [isupper](#), [isalpha](#), ...

ASCII values			characters	isctrl	isprint	isspace	isblank	isgraph	ispunct	isalnum	isalpha	isupper	islower	isdigit	isxdigit
decimal	hexadecimal	octal		iswctrl	iswprint	iswspace	iswblank	iswgraph	iswpunct	iswalnum	iswalpha	iswupper	iswlower	iswdigit	iswxdigit
0-8	\x0-\x8	\0-\10	control codes (NUL, etc.)	≠0	0	0	0	0	0	0	0	0	0	0	0
9	\x9	\11	tab (\t)	≠0	0	≠0	≠0	0	0	0	0	0	0	0	0
10-13	\xA-\xD	\12-\15	whitespaces (\n, \v, \f, \r)	≠0	0	≠0	0	0	0	0	0	0	0	0	0
14-31	\xE-\x1F	\16-\37	control codes	≠0	0	0	0	0	0	0	0	0	0	0	0
32	\x20	\40	space	0	≠0	≠0	≠0	0	0	0	0	0	0	0	0
33-47	\x21-\x2F	\41-\57	!"#\$%&'()*+,-./	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
48-57	\x30-\x39	\60-\71	0123456789	0	≠0	0	0	≠0	0	≠0	0	0	0	≠0	≠0
58-64	\x3A-\x40	\72-\100	: ; < = > ? @	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
65-70	\x41-\x46	\101-\106	ABCDEF	0	≠0	0	0	≠0	0	≠0	≠0	≠0	0	0	≠0
71-90	\x47-\x5A	\107-\132	GHIJKLMNOPQRSTUVWXYZ	0	≠0	0	0	≠0	0	≠0	≠0	≠0	0	0	0
91-96	\x5B-\x60	\133-\140	[\ ^ _ `	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
97-102	\x61-\x66	\141-\146	abcdef	0	≠0	0	0	≠0	0	≠0	≠0	0	≠0	0	≠0
103-122	\x67-\x7A	\147-\172	ghijklmnopqrstuvwxyz	0	≠0	0	0	≠0	0	≠0	≠0	0	≠0	0	0
123-126	\x7B-\x7E	\172-\176	{ } ~	0	≠0	0	0	≠0	≠0	0	0	0	0	0	0
127	\x7F	\177	backspace character (DEL)	≠0	0	0	0	0	0	0	0	0	0	0	0



Конверсия в/из числовые типы

atoi/atol/atoll – преобразование строки в int, long, long long;

strtol strtoll – те же конверсии, но с заданным основанием;

strtof/strtod/strtold – преобразование строки в
float, double, long double;

Обратные преобразования:

- strfromf, strfromf, strfromld;



Копирование строк

Копирование строки:

```
char* strcpy(char* restrict dest, const char* restrict src);
```

Копирование строки с учётом размера буфера dst:

```
char* strncpy(  
    char* restrict dest, const char* restrict src, size_t count);
```

Undefined Behavior в ситуациях:

- Переполнение массива dest;
- Наложение массивов src и dest;
- Отсутствие символа '\0' в строке src.



Склейка строк

Склейка строк:

```
char* strcat(char* restrict dest, const char* restrict src);
```

Склейка строк с учётом размера буфера dst:

```
char* strncat(  
    char* restrict dest, const char* restrict src, size_t count);
```

Undefined Behavior в ситуациях:

- Переполнение массива dest;
- Наложение массивов src и dest;
- Отсутствие символа '\0' в строке src.



Иные полезные функции

Длина строки:

```
size_t strlen(const char* str);
```

Сравнение строк:

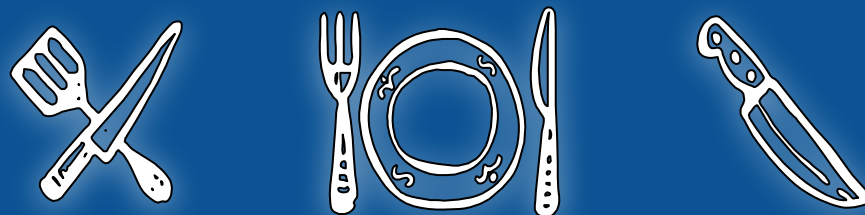
```
int strcmp(const char* lhs, const char* rhs);
```

Токенизация:

```
char* strtok(char* str, const char* delim);
```

Их много!

Пример утилиты для сортировки строк





Разбор утилиты сортировки строк

Разбор утилиты sort:

- [Документация на утилиту](#) (man sort на Linux);
- Собственная реализация – [07_sort_strings](#);

Вопросы?

