Символы и символьные строки. Ввод и вывод информации. Символьный и форматированный ввод-вывод. Примеры.

Символы и символьные строки

Символ - некоторый значок, изображение, это может быть как буква, так и число, знаки препинания и прочее. Компьютером символы используются для хранения текстовой информации. А сами символы хранятся как числовые коды, к примеру, самый популярный набор символов указан в таблице ASCII, где на каждый символ отводится 7 бит, таким образом, всего в ней находятся 128 символов.

Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex	Char	Dec	Oct	Hex
(nul)	0	0000	0x00	(sp)	32	0040	0x20	@	64	0100	0x40	`	96	0140	0x60
(soh)	1	0001	0x01	!	33	0041	0x21	A	65	0101	0x41	a	97	0141	0x61
(stx)	2	0002	0x02	"	34	0042	0x22	B	66	0102	0x42	b	98	0142	0x62
(etx)	3	0003	0x03	#	35	0043	0x23	C	67	0103	0x43	c	99	0143	0x63
(eot)	4	0004	0x04	$	36	0044	0x24	D	68	0104	0x44	d	100	0144	0x64
(enq)	5	0005	0x05	%	37	0045	0x25	E	69	0105	0x45	e	101	0145	0x65
(ack)	6	0006	0x06	&	38	0046	0x26	F	70	0106	0x46	f	102	0146	0x66
(bel)	7	0007	0x07	'	39	0047	0x27	G	71	0107	0x47	g	103	0147	0x67
(bs)	8	0010	0x08	(	40	0050	0x28	H	72	0110	0x48	h	104	0150	0x68
(ht)	9	0011	0x09	)	41	0051	0x29	I	73	0111	0x49	i	105	0151	0x69
(nl)	10	0012	0x0a	*	42	0052	0x2a	J	74	0112	0x4a	j	106	0152	0x6a
(vt)	11	0013	0x0b	+	43	0053	0x2b	K	75	0113	0x4b	k	107	0153	0x6b
(np)	12	0014	0x0c	,	44	0054	0x2c	L	76	0114	0x4c	l	108	0154	0x6c
(cr)	13	0015	0x0d	-	45	0055	0x2d	M	77	0115	0x4d	m	109	0155	0x6d
(so)	14	0016	0x0e	.	46	0056	0x2e	N	78	0116	0x4e	n	110	0156	0x6e
(si)	15	0017	0x0f	/	47	0057	0x2f	O	79	0117	0x4f	o	111	0157	0x6f
(dle)	16	0020	0x10	0	48	0060	0x30	P	80	0120	0x50	p	112	0160	0x70
(dc1)	17	0021	0x11	1	49	0061	0x31	Q	81	0121	0x51	q	113	0161	0x71
(dc2)	18	0022	0x12	2	50	0062	0x32	R	82	0122	0x52	r	114	0162	0x72
(dc3)	19	0023	0x13	3	51	0063	0x33	S	83	0123	0x53	s	115	0163	0x73
(dc4)	20	0024	0x14	4	52	0064	0x34	T	84	0124	0x54	t	116	0164	0x74
(nak)	21	0025	0x15	5	53	0065	0x35	U	85	0125	0x55	u	117	0165	0x75
(syn)	22	0026	0x16	6	54	0066	0x36	V	86	0126	0x56	v	118	0166	0x76
(etb)	23	0027	0x17	7	55	0067	0x37	W	87	0127	0x57	w	119	0167	0x77
(can)	24	0030	0x18	8	56	0070	0x38	X	88	0130	0x58	x	120	0170	0x78
(em)	25	0031	0x19	9	57	0071	0x39	Y	89	0131	0x59	y	121	0171	0x79
(sub)	26	0032	0x1a	:	58	0072	0x3a	Z	90	0132	0x5a	z	122	0172	0x7a
(esc)	27	0033	0x1b	;	59	0073	0x3b	[	91	0133	0x5b	{	123	0173	0x7b
(fs)	28	0034	0x1c	<	60	0074	0x3c	\	92	0134	0x5c	|	124	0174	0x7c
(gs)	29	0035	0x1d	=	61	0075	0x3d	]	93	0135	0x5d	}	125	0175	0x7d
(rs)	30	0036	0x1e	>	62	0076	0x3e	^	94	0136	0x5e	~	126	0176	0x7e
(us)	31	0037	0x1f	?	63	0077	0x3f	_	95	0137	0x5f	(del)	127	0177	0x7f

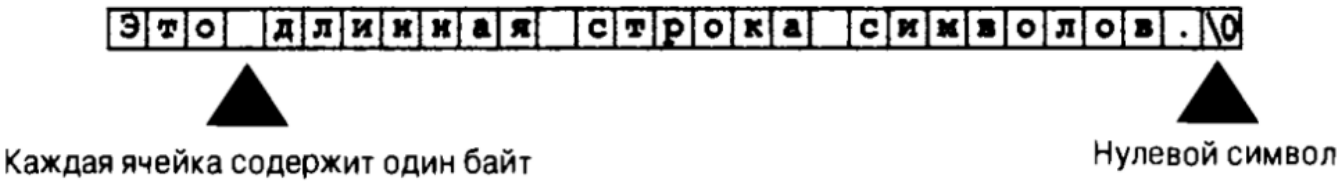
Причем существуют "невидимые символы" под номерами от 0 до 32, которые выполняют разные роли - табуляция, перенос строки, пустой символ и т.д.

В языке Си для хранения символов используется тип `char`.
Так как тип `char` содержит число - можно использовать арифметические операции с ним, наиболее полезными являются операции сложения и вычитания:

```
char l = 66; // Символ "B"
printf("%c", l+1); // Получим символ "C"
printf("%c", l-1); // Получим символ "A"
```

В переменные типа `char` можно записывать как числа, так и сами символы.

Символьные строки
Символьная строка в языке Си – это **последовательность** из одного или большего количества символов, которая заканчивается управляющим символом ‘\0’;
‘\0’ – называют *нулевым символом строки*



В языке Си, двойные(") и одинарные (') кавычки используются в разных случаях. и они невзаимозаменяемые!
“Эта длинная строка символов.” – двойные кавычки определяют строковый литерал, и не считаются частью строки.
‘И’, ‘В’, ‘Т’ – одинарные кавычки идентифицируют 1 символ, который сопоставим с целочисленным типом данных char (1byte).

Символьные строки в Си записываются в массив типа `char`
Каждый символ по отдельности хранится в своей ячейки памяти в естественном виде, также как он и расположен в строке.
На символ конца строки тоже нужно выделять память! То есть:

```
char stroka[16] = "Examen po proge" // Сама строка из 15 символов, но
добавляем                                    // символ конца строки и
получится 16.
```

`char` - тип данных.
`stroka` - название переменной.
`[]` - указывает на использование массива.

`16` - обозначает количество переменных типа `char`, выделяемое в памяти последовательно.
Чтобы узнать сколько байт весит символьная строка или другой тип данных используется `sizeof(<то что нужно измерить>)`.

Символьный ввод-вывод
Символьные операции позволяют работать с отдельными символами. Основные функции для работы с символами:
Функции ввода:

- `getchar()`
  - Читает один символ из стандартного потока ввода (`stdin`).
  - Возвращает считанный символ или `EOF` в случае ошибки.
  - Пример:
  `char c = getchar();`
  Функции вывода:
- `putchar(int c)`
  - Выводит один символ в стандартный поток вывода (`stdout`).
  - Возвращает этот символ или `EOF` в случае ошибки.
  - Пример:
  `putchar('A');
  Особенности:
- `getchar()` и `putchar()` полезны для построчной или побуквенной обработки текста.
- Работают с буферизованным вводом/выводом.

Форматированный ввод-вывод
Для работы со строками и числами в стандартных потоках используются функции `scanf` и `printf`.
Функции форматированного ввода:

- `scanf(const char *format, ...)`
  - Читает данные из `stdin` и сохраняет их в переменные.
  - Форматная строка (`format`) определяет типы считываемых данных.
  - Примеры спецификаторов:
    - `%d` — целое число.
    - `%f` — число с плавающей точкой.
    - `%s` — строка (до первого пробела).

- `%c` — символ.

- Пример:

```
int a;
float b;
char str[100];
scanf("%d %f %s", &a, &b, str);
```

Функции форматированного вывода:

- `printf(const char *format, ...)`
  - Выводит данные в стандартный поток вывода (`stdout`).
  - Форматная строка (`format`) определяет, как выводятся данные.
  - Примеры спецификаторов:
  - `%d` — целое число.
  - `%f` — число с плавающей точкой (можно указать точность, например `%.2f`).
  - `%s` — строка.
  - `%c` — символ.
  - Пример:
  `c int a; float b; char str[100]; scanf("%d %f %s", &a, &b, str);`
  Работа с форматными строками
  Форматная строка определяет, как обрабатываются данные:
- **Пример вывода с шириной и точностью:**
  `printf("%10.2f\n", 123.456); // Ширина 10, точность 2`
- **Спецификаторы для чисел:**
  - `%x` — шестнадцатеричное представление.
  - `%o` — восьмеричное представление.
  - `%e` — экспоненциальный формат.
- **Пример ввода:**
  - Ввод в определённом формате:

```
int a, b;
scanf("%2d %3d", &a, &b); // Читает 2 цифры в a и 3 цифры в b
```

Важные моменты

- **Буферизация ввода:**
  - После ввода строки или числа в `scanf`, в буфере могут остаться символы новой строки (`\n`), что может мешать дальнейшему вводу.

- *Решение*: использовать `getchar()` для очистки буфера.

- **Ограничение длины строки:**
  - В `scanf` для строк можно задавать ограничение длины:

```
char str[20];
scanf("%19s", str); // Считает максимум 19 символов + символ конца строки
```

Примеры

- **Подсчёт символов:**

```
int count = 0;
char c;
while ((c = getchar()) != EOF) {
    count++;
}
printf("Всего символов: %d\n", count);
```

- **Обратный порядок строки:**

```
char str[100];
printf("Введите строку: ");
scanf("%s", str);
for (int i = strlen(str) - 1; i >= 0; i--) {
    putchar(str[i]);
}
putchar('\n');
```

- **Форматированный вывод таблицы(число-квадрат-куб):**

```
for (int i = 1; i <= 10; i++) {
    printf("%-3d %5d %10d\n", i, i * i, i * i * i);
}
```