Операторы ввода/вывода в языке Си

Базовые функции ввода/вывода

Операции ввода/вывода в языке Си организованы посредством различных библиотечных функций.

Функция printf( ), прототип которой содержится в файле stdio.h, обеспечивает форматированный вывод. Ее можно записать в следующем формальном виде:

printf ("управляющая строка", список вывода);

Функция printf

Список вывода содержит перечисленные через запятую имена выводимых переменных, т. е. показывает, что выводить. В список вывода можно включать не только переменные, но и произвольные выражения (в частном случае константы).

```
1  #include <stdio.h>
2  int main()
3  {
4  int x=12;
5  printf("%d      %d          ",x,2*x*x-24);
6  printf("x=%d      2*x*x-24=%d",x,2*x*x-24);
7  return 0;
8  }
```

```
12      264          x=12      2*x*x-24=264
Process returned 0 (0x0)   execution time : 0.188 s
Press any key to continue.
```

Функция printf

Управляющая строка содержит компоненты трех типов:

обычные символы, которые просто копируются в стандартный выходной поток (выводятся на экран дисплея);

спецификации преобразования, каждая из которых вызывает вывод на экран очередного аргумента из списка;

управляющие символьные константы.

Управляющая строка показывает, в каком виде значения переменных будут выведены на экран. В простейшем случае управляющая строка - это строковая константа, т.е. она ограничена двойными кавычками. Каждая спецификация преобразования начинается со знака % и заканчивается некоторым символом, задающим преобразование.

Функция printf

Для вывода значений на экран нужно в функции написать правильный **спецификатор формата**. Ниже представлены спецификаторы формата для целых чисел:

спецификатор	назначение
%d	для вывода целых чисел в десятичной форме
%u	для вывода целых чисел в десятичной форме без знака
%o	для вывода целых чисел без знака в восьмеричной форме
%x	для вывода целых чисел без знака в шестнадцатеричном формате

Функция printf

Для вывода значений на экран нужно в функции написать правильный **спецификатор формата**. Ниже представлены спецификаторы формата для целых чисел.

here | Exs_2_уфункции вывода printf.c

```c
#include <stdio.h>
int main ( )
{
int a = 31; float b=314.14;
printf ("%d  %o  %x  ",a,a,a);
//printf ("%f  %e  ", b, b);
return 0;
}
```

```
31  37  1f
Process returned 0 (0x0)   execution time : 1.031 s
Press any key to continue.
```

Функция printf

Если после знака % стоит цифра, то она задает поле, в котором будет выполнен вывод числа. Приведем несколько функций printf, которые будут обеспечивать вывод одной и той же переменной S целого типа, имеющей значение **336**.

Функция **printf(“%2d”, S);** выдает на экран: **336**

В этом примере ширина поля (она равна двум) меньше, чем число цифр в числе 336, поэтому поле автоматически расширяется до необходимого размера.

Функция **printf(“%6d”, S);** выдаст на экран: **_ _ _336**

То есть, в результате работы функции число сдвинуто к правому краю поля, а лишние позиции перед числом заполнены пробелами.

Функция **printf(“%-6d”, S);** выдаст на экран: **336_ _ _**

Знак «минус» перед спецификацией приводит к сдвигу числа к левому краю поля

Функция printf

here | Exs_2_уфункции вывода printf.c

```c
#include <stdio.h>
int main ( )
{
int a = 341;
printf ("%2d%6d%-6d%d",a,a,a,a);
return 0;
}
```

```
341   341341   341
Process returned 0 (0x0)   execution time : 0.156 s
Press any key to continue.
```

Функция printf

Спецификаторы формата для вещественных чисел:

спецификатор	назначение
%f	для вывода вещественных чисел в обычной форме
%e	для вывода вещественных чисел в экспоненциальной форме

here | Exs_2_уфункции вывода printf.c

```c
#include <stdio.h>
int main ( )
{
int a = 31; float b = 314.189;
//printf ("%d  %o  %x  ",a,a,a);
printf ("%f  %e  ", b, b);
return 0;
}
```

```
314.188995  3.141890e+002
Process returned 0 (0x0)   execution time : 0.953 s
Press any key to continue.
```

Вывод вещественных чисел

Если перед спецификацией f ничего не указано, то выводится число с шестью знаками после запятой. При печати числа с плавающей точкой перед спецификацией f тоже могут находиться цифры. Рассмотрим на примере три возможные ситуации:

%6f – печать числа с плавающей точкой в поле из шести позиций;

%.2f – печать числа с плавающей точкой с двумя цифрами после десятичной точки;

%6.2f – печать числа с плавающей точкой в поле из шести позиций и двумя цифрами после десятичной точки.

Вывод вещественных чисел

Exs_2_уфункции вывода printf.c

```c
#include <stdio.h>
int main ( )
{ float a=3.687; float b=10.17;
printf ("%f%f\n", a, b);
printf ("%12f%12f  ", a, b);
return 0;
}
```

```
3.68700010.170000
    3.687000   10.170000
Process returned 0 (0x0)   execution time : 0.203 s
Press any key to continue.
```

Вывод вещественных чисел

Exs_2_уфункции вывода printf.c

```c
#include <stdio.h>
int main ( )
{ float a=3.687; float b=10.17;
printf ("%f%f\n", a, b);
printf ("%12.2f%12.1f  ", a, b);
return 0;
}
```

```
3.68700010.170000
        3.69        10.2
Process returned 0 (0x0)   execution time : 1.078 s
Press any key to continue.
```

Вывод вещественных чисел

Exs_2_уфункции вывода printf.c

```c
#include <stdio.h>
int main ( )
{ float a=3.687; float b=10.17;
printf ("%f%f\n", a, b);
printf ("%.2f%10.1e  ", a, b);
return 0;
}
```

```
3.68700010.170000
3.69  1.0e+001
Process returned 0 (0x0)   execution time : 0.172 s
Press any key to continue.
```

Вывод вещественных чисел

Поскольку для вывода значения переменной b применена спецификация e, то результат выдан в экспоненциальной форме. Следует отметить, что, если ширина поля меньше, чем число цифр в числе, то поле автоматически расширяется до необходимого размера.

Управляющие символьные константы

В управляющей строке могут содержаться управляющие символьные константы. Среди управляющих символьных констант наиболее часто используются следующие:

- 1) \a – для кратковременной подачи звукового сигнала;
- 2) \b – для перевода курсора влево на одну позицию;
- 3) \n – для перехода на новую строку;
- 4) \r – для перевода курсора в начало текущей строки;
- 5) \t – для горизонтальной табуляции.

Управляющие символьные константы

В управляющей строке могут содержаться управляющие символьные константы.

Exs_2_2_printf управляющие символы.c

```c
#include <stdio.h>
int main()
{
int a=1234; int b=987;
printf("%d%d",a,b);
printf("%d%d\n",a,b);
printf("%d\b%d\a\n",a,b);
printf("%d\r%d\n",a,b);

return 0;
}
```

```
12349871234987
123987
9874

Process returned 0 (0x0)   execution time : 0.185 s
Press any key to continue.
```

Управляющие символьные константы

Другие символьные константы.

```
#include <stdio.h>
int main()
{
int a=1234; int b=987;
printf("%d%d\n",a,b);
printf("\t%d\t%d\n",a,b);
printf("%d\n%d\n",a,b);
return 0;
}
```

```
1234987
        1234    987
1234
987

Process returned 0 (0x0)   execution time : 0.188 s
Press any key to continue.
```

Форматный ввод

Функция форматного ввода

Оператор вызова этой функции форматного ввода scanf ( ) имеет вид:

scanf(форматная_строка, список_ввода)

Список ввода показывает, что выводить. Он содержит перечисленные через запятую адреса вводимых переменных. В список ввода не могут входить выражения или константы, так как ввод предполагает изменение значения.

Форматный ввод

Функция форматного ввода

Оператор вызова этой функции форматного ввода scanf ( ) имеет вид: scanf(форматная_строка, список_ввода)

```
#include <stdio.h>
int main()
{ int a,b;
scanf("%d%d",&a,&b);
printf("\t%d\t%d\n",a,b);
printf("%d\n%d\n",a,b);
return 0;}
```

```
1234
987
        1234    987
1234
987

Process returned 0 (0x0)   execution time : 8.047 s
Press any key to continue.
```

Форматный ввод

Форматная строка - это строковая константа, которая, так же как при выводе, показывает, в каком виде значения переменных будут выглядеть на экране. Форматная строка при вводе содержит только спецификации формата, включать в нее какой либо пояснительный текст бессмысленно. Спецификации формата при вводе записываются так же при выводе, но ширина поля и точность обычно упускаются.

Заметим, что для функции scanf( ) после ввода числа или символа необходимо нажать клавишу <Enter>.

Форматный ввод

Управляющая строка содержит спецификации преобразования и используется для установления количества и типов аргументов. спецификации для определения типов аргументов такие же, как и для функции printf.

```
#include <stdio.h>
int main()
{ int a; float b;
scanf("%d%f",&a,&b);
printf("\t%d\t%f\n",a,b);
printf("%d\n%f\n",a,b);
return 0;}
```

```
123
9876
        123     9876.000000
123
9876.000000

Process returned 0 (0x0)   execution time : 6.032 s
Press any key to continue.
```

Посимвольное чтение и вывод

В СИ есть простой механизм ввода - чтение по одному символу из стандартного входного потока, с клавиатуры, с помощью функции getchar( ). Она имеет следующий прототип (т.е. описание заголовка):

```
int getchar(void);
```

Здесь определен тип единственного аргумента (void) и тип возвращаемого функцией значения (int). Оператор вида:

```
x = getchar( );
```

присваивает переменной x очередной вводимый символ. Переменная x должна иметь символьный или целый тип.

Заметим, что для функции getchar( ) после выбора символа необходимо нажать клавишу

Посимвольное чтение и вывод

Другая функция - putchar(x) выдает значение переменной x в стандартный выходной поток - на экран дисплея. Функция putchar( ) имеет прототип:

```
int putchar(int);
```

Объявления getchar( ) и putchar( ) сделаны в заголовочном файле stdio.h, содержащем описания заголовков библиотечных функций стандартного ввода/вывода. Чтобы библиотечные функции стали доступны программе, к ней необходимо подключить данный файл. Подключение осуществляется с помощью директивы препроцессора

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

помещаемой в начало программы.

Посимвольное чтение и вывод

char – символьный тип в СИ

Символьные константы

- Последовательность из одного или нескольких символов, заключенная в одинарные кавычки

  ‘A’ ‘a’ ‘8’ ‘+’ ‘;’

- Значение символьной константы равно числовому коду символа в таблице кодировки

Таблица кодировки символов

- ASCII (American Standard Code for Information Interchange) – это код для представления символов в виде чисел, в котором каждому символу сопоставлено число от 0 до 127
- Стандартный набор символов ASCII использует только 7 битов для каждого символа. Добавление 8-го разряда позволяет увеличить количество кодов таблицы ASCII до 255
- Коды от 128 до 255 представляют собой расширение таблицы ASCII. Эти коды используются для кодирования символов национальных алфавитов, а также символов псевдографики

http://ascii.org.ru/ascii.pdf

Таблица кодировки символов

Dec	Hex	Char	Cmd
0	00		NUL
1	01	☺	SOH
2	02	☻	STX
3	03	♥	ETX
4	04	♦	EOT
5	05	♣	ENQ
6	06	♠	ACK
7	07	•	BEL
8	08	◘	BS
9	09	○	TAB
10	0A	◙	LF
11	0B	♂	VT
12	0C	♀	FF
13	0D	♪	CR
14	0E	♫	SO
15	0F	☼	SI
16	10	►	DLE
17	11	◄	DC1
18	12	↕	DC2
19	13	‼	DC3
20	14	¶	DC4
21	15	§	NAK
22	16	▬	SYN
23	17	↨	ETB
24	18	↑	CAN
25	19	↓	EM
26	1A	→	SUB
27	1B	←	ESC
28	1C	∟	FS
29	1D	↔	GS
30	1E	▲	RS
31	1F	▼	US

Dec	Hex	Char	Cmd
32	20		(sp)
33	21	!	
34	22	"	
35	23	#	
36	24	$	
37	25	%	
38	26	&	
39	27	'	
40	28	(	
41	29	)	
42	2A	*	
43	2B	+	
44	2C	,	
45	2D	-	
46	2E	.	
47	2F	/	
48	30	0	
49	31	1	
50	32	2	
51	33	3	
52	34	4	
53	35	5	
54	36	6	
55	37	7	
56	38	8	
57	39	9	
58	3A	:	
59	3B	;	
60	3C	<	
61	3D	=	
62	3E	>	
63	3F	?	

Dec	Hex	Char	Cmd
64	40	@	
65	41	A	
66	42	B	
67	43	C	
68	44	D	
69	45	E	
70	46	F	
71	47	G	
72	48	H	
73	49	I	
74	4A	J	
75	4B	K	
76	4C	L	
77	4D	M	
78	4E	N	
79	4F	O	
80	50	P	
81	51	Q	
82	52	R	
83	53	S	
84	54	T	
85	55	U	
86	56	V	
87	57	W	
88	58	X	
89	59	Y	
90	5A	Z	
91	5B	[	
92	5C	\	
93	5D	]	
94	5E	^	
95	5F	_	

Dec	Hex	Char	Cmd
96	60	`	
97	61	a	
98	62	b	
99	63	c	
100	64	d	
101	65	e	
102	66	f	
103	67	g	
104	68	h	
105	69	i	
106	6A	j	
107	6B	k	
108	6C	l	
109	6D	m	
110	6E	n	
111	6F	o	
112	70	p	
113	71	q	
114	72	r	
115	73	s	
116	74	t	
117	75	u	
118	76	v	
119	77	w	
120	78	x	
121	79	y	
122	7A	z	
123	7B	{	
124	7C	|	
125	7D	}	
126	7E	~	
127	7F	⌂	DEL

Символьный тип

%c - прочитать символ

char ch;

scanf ("%c", &ch);

Exs_2_2_scanf.c | Exs_4_1_ ch='a' +i;.c

```
#include <stdio.h>
int main()
{ char ch;
  int i;
     for(i=0;i<10;i++)
  {   ch='a' +i;
      printf(" %c",ch);
      ch = ch - 32;
       printf(" %c\t",ch);    }
    return 0;   }
```

```
 a A     b B     c C     d D     e E     f F     g G     h H     i I     j
Process returned 0 (0x0)   execution time : 0.172 s
Press any key to continue.
```

Символьный тип

getchar ();

putchar (rsim);

```
#include <stdio.h>
int main ( )
{
char rsim ;
printf ("Enter a symbol\n");
rsim = getchar ();
printf ("Symbol:\n");
putchar (rsim);
return 0;
}
```

```
Enter a symbol
G
Symbol:
G
Process returned 0 (0x0)   execution time : 3.047 s
Press any key to continue.
```

Символьный тип

puts ();

```
#include <stdio.h>
int main ( )
{
int rsim ;
puts ("Enter a symbol");
rsim = getchar ();
printf ("Symbol: %c\n",rsim);
return 0;
}
```

```
Enter a symbol
K
Symbol: K

Process returned 0 (0x0)   execution time : 6.072 s
Press any key to continue.
```

Символьный тип

gets ();

puts ();

```
#include <stdio.h>
int main ( )
{
char rsim[80] ;
puts ("Enter a symbol");
gets (rsim);
getchar ();
puts (rsim );
return 0;
}
```

```
Enter a symbol
asdfghj

asdfghj

Process returned 0 (0x0)   execution time : 6.266 s
Press any key to continue.
```