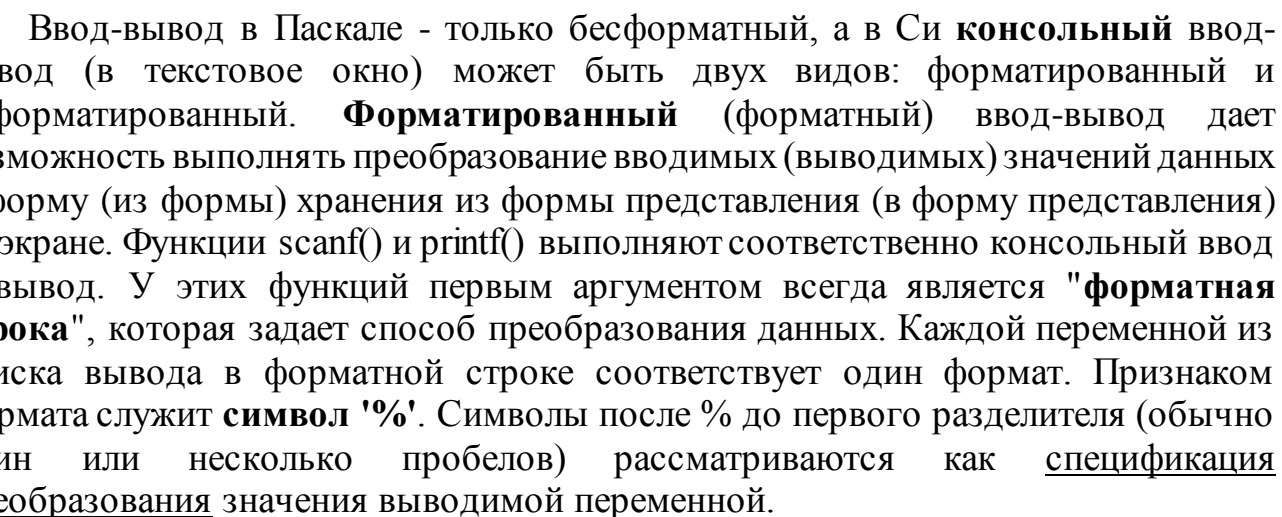


4.1. Форматный вывод значений



Функция **printf()** выдает на экран всю форматную строку, в которой знаки %xxx заменяются символами выводимой информации. Для **перехода на новую строку** (по аналогии с Writeln) надо вставить символ ‘\n’ в конец форматной строки.

Полный вид спецификации формата:

%[флаг]	[ширина]	[.точность]	[размер]	тип_формата
----	-----	-----	-----	-----
				см. ниже
	+-----+			

- прижать значение как в
влево при выводе Паскале

0 заполнить свободное
пространство нулями

выводить 0 перед
8-м числом и 0X
перед 16-м числом

h короткое целое (short)
l длинное целое (long)
L long double



В пределах форматной строки можно использовать так называемые **ESC-последовательности** (управляющие символы). ESC-последовательность состоит из наклонной черты, за которой следует буква, знаки пунктуации ' " \ ? или комбинация цифр (типа \"). Если же необходимо вывести знак %, то в форматной строке должно быть \%%.

ESC-последовательность	Наименование
\n == '\x0A'	Новая строка
\r == '\x0D'	Возврат каретки
\t	Горизонтальная табуляция
\b (blank) == '\x20'	Пробел
\'	Одиночная кавычка
\"	Двойная кавычка
\\	Наклонная черта
\ddd	ASCII символ в восьмеричном представлении
\xdd	ASCII символ в шестнадцатиричном представлении
\0	Нулевое значение байта

Примеры:

Паскаль

Си

var

```
A,B,C: Integer;  
Amt : Real;  
Name : string(20);  
AM : Char;
```

begin

```
Write('Привет');  
Writeln('Еще привет');  
Writeln("Привет," сказал я');
```

```
Name := 'Иван';  
Writeln('Вас зовут', Name);  
A:=1;  
B:=2;  
C:=A+B;  
Writeln(A,' + ',B,' = ',C);  
AM := 12.34;  
Writeln('Ваш долг $',  
    AM:6:2);  
Writeln(' A*A = ',(A*A):6);
```

```
int a,b,c;  
float amt;  
char name[21]; (или *name)  
char am;  
void main(void)
```

```
{  
    printf("Привет");  
    printf("Еще привет\n");  
    printf("\"Привет, \"сказал я\n");
```

```
--      --      --  
|        |        |
```

ESC-последовательности

```
strcpy(name, "Иван");  
    printf("Вас зовут %s\n", name);  
a=1;  
b=2;  
  
printf("%d + %d = %d\n",a,b,c=a+b);  
am = 12.34;  
printf("Ваш долг  
$%6.2f\n", am);  
printf(" a*a = %6d\n",a*a);
```

4.2. Вывод строк (форматный и бесформатный)

Неформатированный вывод строк в Си выполняется функциями вывода строк и функциями вывода символов (строка выводится посимвольно).

Следующие строки на Паскале и Си эквивалентны:

Var s: string[80];	char s[81];	не надо для puts
Writeln(Name);	puts(Name);	(но надо для printf)
Writeln('Мы здесь!);	puts("Мы здесь!");	
	printf("Мы здесь!\n");	

Функция puts() получает строку в качестве аргумента и выводит ее на экран. При этом символ '\0' заменяется парой символов \r\n (при работе в среде Windows), где \n == 0x0A служит для перехода на новую строку, а \r == 0x0D служит для возврата каретки. При вводе [т.е. в случае функции gets()] производится обратная замена: пара символов \r\n заменяется на один символ '\0':

в текстовом режиме	
+-----+ ---ввод----->	+-----+ /
\r \n (с клавиатуры)	\0
+-----+ <---вывод-----	+-----+ \
(на экран)	
в текстовом режиме	

Замечание: необходимо помнить о различии в выводе строк с помощью puts() и printf():

puts("Строка");		puts("Строка1"); puts("Строка2")	будут на разных строках экрана
или printf("%s", "Строка");		или printf("%s", "Строка1");	
		printf("%s", "Строка2");	будут на одной строке экрана

функция puts() выведет на экран слово (то, что между двойными кавычками) и переведет курсор в начало следующей строки. В отличие от этого функция printf() курсор в начало следующей строки не переведет. Для перехода на следующую строку надо использовать ESC-последовательность \n:

```
printf("%s \n", "Строка");
```

--

4.3. Вывод символов - форматный (printf) и бесформатный (_putch)

Функции

```
printf("%c", ch);    --- форматный вывод
putchar(ch);         --- бесформатный вывод
_putchar(ch);        --- бесформатный вывод
```

выводят на экран один символ. Например конструкция на Паскале вида:

```
Var ch: char;
write(ch);
```

эквивалентна на языке Си следующим конструкциям:

```
char ch;
putchar(ch); или _putch(ch);
                с подчеркиванием в VC++ (нужен <conio.h>)
```

Пример: посимвольный вывод строки на экран.

Вариант № 1. Бесформатный вывод (нужен <conio.h>).

```
int i;
char s[10] = "Строка";
void main(void)
{
    clrscr(); //нужен был #include <conio.h> в VC++ Builder
    i=0;
    while (s[i] != '\0')
    {
        putchar(s[i++]);
        Sleep(100); //задержка 100мс - нужен <windows.h>
    }
}
```

на Си это system("cls");
(нужен windows.h и stdlib.h)
ситуация продолжения цикла
переход к следующему символу строки
вместо delay(100) на Паскале
в Windows нужна большая S (в Unix - маленькая)

Вариант № 2. Форматный вывод.

```
#include <stdio.h>
void print_string(char *s)
{
    while (*s != NULL)
    {
        printf("%c", *s);
        Sleep(100);
        s++;
    }
}
void main(void)
{
    print_string("Печать строки");
}
```

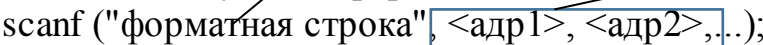
или (*s)++ ≠ *s++
операции ++ и * имеют одинаковый приоритет и выполняются **справа налево**

5. Ввод значений - форматный и бесформатный

5.1. Ввод значений (форматный)

здесь нельзя использовать \n

В Паскале имеется базовая процедура ввода Read() с вариантами Readln(), Read(f),

В Си базовая функция, используемая для ввода с клавиатуры, называется scanf() и имеет следующий формат вызова: 

где "форматная строка" - строка, содержащая опции форматирования (аналогично printf **за исключением формата g** для вещественных чисел), а каждый <адр> из списка ввода - это адрес, по которому scanf размещает вводимые данные. Это значит, что для имен из списка ввода необходимо будет использовать операцию взятия адреса (&).

Например, если a и b - целые числа, то правильно их ввести и вывести можно так:

```
printf("%d %d", a, b );
scanf( "%d %d", &a, &b).
```

Функция scanf() принимает с консоли все введенные (или оставшиеся от предыдущего ввода) символы до нажатия клавиши ENTER и помещает их во **внутренний буфер** [fflush(stdout) очищает буфер стандартного файла вывода, но fflush(stdin) не очищает буфер стандартного файла ввода stdin - надо вручную в цикле посимвольно очищать]. После нажатия кнопки ENTER функция scanf() прекращает прием символов в буфер и переходит к обработке последовательности символов в буфере в соответствии с форматной строкой. При этом по форматной строке определяется способ преобразования введенных в буфер символов из формы представления на экране (как мы их вводили с клавиатуры) в формат хранения (в памяти ЭВМ) в соответствии с заданными спецификациями формата. Полученное в результате преобразования значение помещается по адресу очередной переменной из списка ввода.

5.2. Ввод строк (форматный и бесформатный)

Строки можно вводить двумя способами.

Первый - с помощью

```
scanf("%s", адрес начала строки);
```

имя символьного массива (char s[80])

имя указателя на строку (char *s)

адрес первого символа строки (&s[0])

которая читает входную строку до первого разделителя (обычно пробел).

Второй - с помощью

```
gets(адрес начала строки),
```

которая читает входную строку до конца.

При этом строка может представляться **двумя путями**:

1) char *s; //здесь в scanf() адрес задается именем указателя (без '&');

2) char s[80]; //здесь адрес в scanf() может задаваться:

// а) именем массива - scanf("%s", s);

// б) адресом 1-го символа - scanf("%s", &s[0]) -- так лучше.

Отличия этих двух путей в следующем:

- во 2-м случае **память выделена под строку статически**.


- в 1-м случае **память выделена лишь под указатель** на начало строки. Если забыть динамически выделить память под строку и ввести значение по адресу s, то в программе будет испорчено все, что после s ;

Отличия чтения строки с помощью scanf() и gets():

- **scanf()** читает все символы до тех пор, пока во входном потоке не встретится разделитель (пробел, символ табуляции или конец строки);
- **gets()** считывает весь входной поток, т.е. любые символы (включая пробелы), до конца строки.

Пример: пусть входной поток равен

1	2	3
---	---	---

char s[81];  лучше &s[0]

scanf("%s", s);
scanf("%s", s);
scanf("%s", s); } или вместо них один gets(s);

Выводы:

1) с помощью scanf() ввести целиком строку, содержащую разделители, нельзя (введется или только часть строки до первого разделителя или то что между соседними разделителями или то что между последним разделителем и концом строки);

2) **остаток строки** после чтения с помощью scanf() останется в буфере и может быть считан следующим обращением к scanf(). Чтобы этого не произошло (то есть чтобы следующий вызов scanf() начал считывание со следующей строки), надо перед вызовом scanf() очищать буфер стандартного входного потока stdin.

5.3. Ввод символов (форматный и бесформатный)


Пусть имеется

char ch;

Возможные варианты его ввода:

1) scanf("%c", &ch)

2) ch = ; ---  _getch() в Visual Studio

3) ch = ; --- читает код символа с клавиатуры **без эхо-отображения**.
и клавишу Enter не надо нажимать после ввода символа
вызывается как функция без аргументов.

3) ch = getchar(); --- читают код символа с клавиатуры **с эхо-отображением**.

4) ch = getc(stdin); --- и клавишу Enter надо нажимать после ввода символа

для очистки буфера входного потока надо:

```
while ((getchar()) != '\n');
```

нельзя '\0', т.к. мы очищаем
именно буфер, а не строку

Пример 1

```
char ch1;  
char ch2;  
scanf("%c", &ch1);  
ch2 = getchar(); //при вводе нажимали клавишу Enter  
printf("ch1 = %c", ch1);  
printf("ch2 = %c", ch2);  
// В буфере останется #13#10
```

Пример 2

```
char s[81]; //предусмотрели место под символ конца строки \0
printf("Введите символ: ");
s[0] = _getch();
printf("\nВведите символ: ");
s[1] = _getch();
printf("\n Введите символ: ");
s[2] = _getch();
printf("\n Введите символ: ");
s[3] = _getch();
printf("\n Введите символ: ");
s[4] = _getch();
s[5] = '\0';
printf("\nS = %s \n", s);
```

Если в ответ на приглашения ввода вводить последовательно 1,2,3,4,5, то на экране будет:

Введите символ:

Введите символ:

Введите символ:

Введите символ:

Введите символ:

S = 12345

Пример 3 -- что будет, если в Примере 2 _getch() заменить на getchar()?

на экране будет другая картина

```
Введите символ: 1
пустая строка.....
Введите символ: 2
пустая строка.....
Введите символ: 3
пустая строка.....
S = 1
2
3
```

чтобы было как раньше (как в Примере 2), надо будет после каждого getchar() вставлять очистку буфера ввода:

```
while ((getchar)) != '\n';
```