

Лекция 13

Ввод-вывод

Ввод-вывод

Возможности для ввода и вывода не являются частью самого языка Си.

Рассмотрим **стандартную библиотеку `stdio.h`**, содержащую набор функций, обеспечивающих ввод-вывод.

Библиотечные функции ввода-вывода точно определяются стандартом ANSI, так что они совместимы в любых реализациях Си.

1. Стандартный ввод-вывод

Библиотечные функции реализуют простую **модель текстового ввода-вывода**.

Текстовый поток состоит из последовательности строк; каждая строка заканчивается символом новой строки.

Простейший механизм ввода — это чтение одного символа из *стандартного потока ввода* (клавиатуры) функцией **getchar**:

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

В качестве результата функция **getchar** возвращает символ из потока ввода или **EOF (-1)**, если обнаружен конец файла.

1. Стандартный ввод-вывод

Пример:

```
#include <stdio.h>
main ( ) {
    int v;
    while ((v=getchar()) != -1)
        printf("\t%c\n", v);
    return 0;
}
```

abc123^Z

a

b

c

1

2

3

1. Стандартный ввод-вывод

Во многих системах **клавиатуру можно заменить файлом**, перенаправив ввод с помощью значка **<**. Так, если программа **prog** использует **getchar**, то командная строка

prog < infile

предпишет программе читать символы из файла **infile**, а не с клавиатуры.

The screenshot displays a Windows desktop environment. In the background, a Notepad window titled 't1.txt - Блокнот' contains the text '4321x'. In the foreground, a command prompt window titled 'C:\WINDOWS\system32\cmd.exe - 1.exe' shows the execution of a program. The user enters the command 'C:\1\2>1.exe < t1.txt'. The program's output, displayed on a black background, is the characters '4', '3', '2', '1', and 'x' on separate lines. Below this, the user enters the command 'C:\1\2>1.exe' followed by the input '567qwerty^Z', which results in the characters '5', '6', '7', 'q', 'w', 'e', 'r', 't', and 'y' being printed on separate lines.

```
1.cpp x
1  #include <stdio.h>
2  main ( ) {
3      int v;
4      while ((v=getchar()) != -1)
5          printf("\t%c\n", v);
6      return 0;
7  }
8

C:\WINDOWS\system32\cmd.exe - 1.exe
C:\1\2>1.exe < t1.txt
4
3
2
1
x

C:\1\2>1.exe
567qwerty^Z
5
6
7
q
w
e
r
t
y
```

1. Стандартный ввод-вывод

Функция

`int putchar(int)`

используется для **ВЫВОДА**.

Вызов `putchar(c)` отправляет символ `c` в *стандартный поток вывода* (дисплей).

Функция `putchar` в качестве результата возвращает посланный символ или, в случае ошибки, **EOF**.

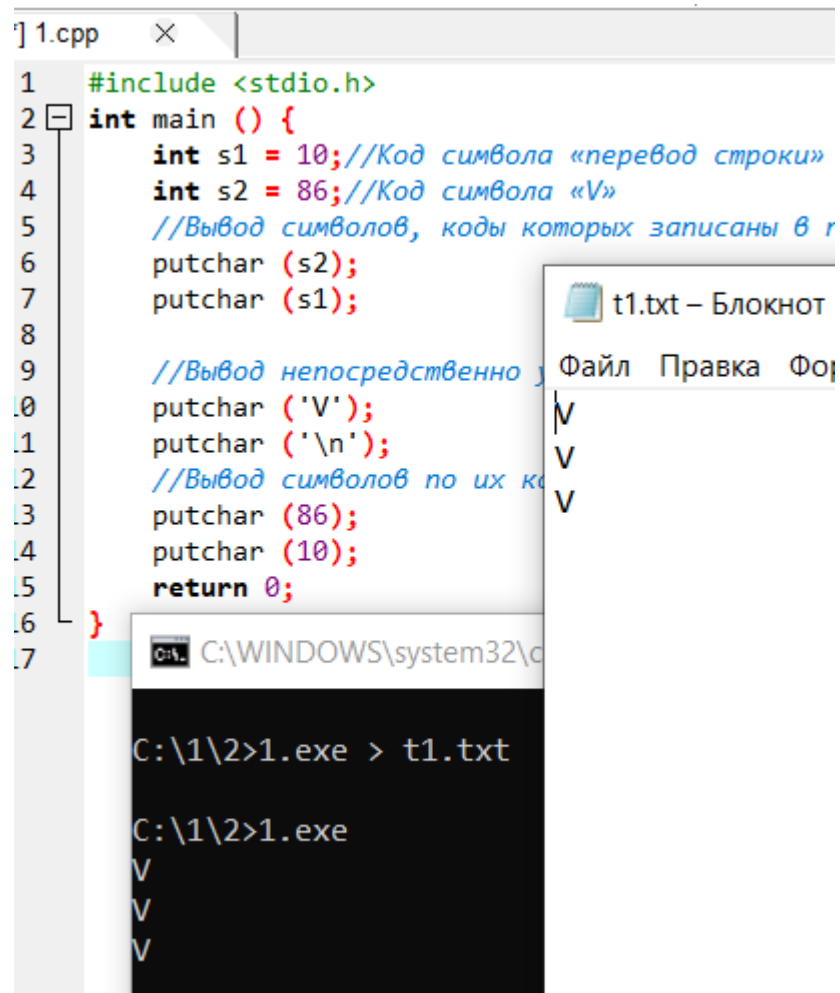
С помощью записи вида `> имя-файла` **ВЫВОД** `putchar` можно перенаправить **в файл**:

`prog > outfile`

1. Стандартный ввод-вывод

```
#include <stdio.h>

int main () {
    int s1 = 10; //код символа «перевод строки»
    int s2 = 86; //код символа «V»
    //вывод переменных
    putchar (s2);
    putchar (s1);
    //ВЫВОД СИМВОЛОВ
    putchar ('V');
    putchar ('\n');
    //ВЫВОД СИМВОЛОВ ПО ИХ КОДАМ
    putchar (86);
    putchar (10);
    return 0;
}
```



The screenshot displays a C++ program in a code editor and its execution results. The code defines two integers, `s1` (10) and `s2` (86), and uses `putchar` to output their values and corresponding characters. The output window shows the characters 'V' and a newline, followed by the characters 'V' and a newline, which is visually represented by two lines of 'V'.

```
1  #include <stdio.h>
2  int main () {
3      int s1 = 10; //Код символа «перевод строки»
4      int s2 = 86; //Код символа «V»
5      //Вывод символов, коды которых записаны в r
6      putchar (s2);
7      putchar (s1);
8
9      //Вывод непосредственно
10     putchar ('V');
11     putchar ('\n');
12     //Вывод символов по их ко
13     putchar (86);
14     putchar (10);
15     return 0;
16 }
17
```

Output window (t1.txt - Блокнот):

```
File  Edit  Format
V
V
V
```

Command Prompt (C:\WINDOWS\system32\cmd.exe):

```
C:\1\2>1.exe > t1.txt

C:\1\2>1.exe
V
V
V
```

2. Форматный вывод (printf)

Функция **printf** преобразует, форматирует и печатает свои аргументы в стандартном потоке вывода под управлением строки формата. Возвращает она количество напечатанных символов.

```
int printf(char *format, arg1, arg2, . . . );
```

Строка формата содержит два вида объектов: **обычные символы**, которые копируются в выходной поток, и **спецификации преобразования**, каждая из которых вызывает преобразование и печать очередного аргумента **printf**.

Спецификация преобразования начинается знаком % и заканчивается символом-формата:

% [флаг] [ширина] [.точность] [h|l] символ_формата

где **ширина** — минимальное количество позиций, отводимых под выводимое значение,

точность — количество позиций, отводимых под дробную часть числа.

2. Форматный вывод (printf)

Модификаторы [h|l]: h-short или unsigned short; l-long или unsigned long (для целых) или long double (для вещественных).

Флаг – если равен минусу, то выравнивание по левому краю.

Таблица – Описание значений поля **символ_формата**

символ_формата	тип выводимого объекта
c	char; единичная литера
s	char *; печатает символы до \0 или в кол-ве заданном точностью
d,i	int; десятичное целое
o	int; беззнаковое восьмеричное
u	int; беззнаковое десятичное целое
x,X	int; беззнаковая шестнадцатеричное
f	double; вещественное число с фиксированной точкой
e,E	double; вещественное число с плавающей точкой
g,G	double; вещественное число в виде %f или %e в зависимости от значения
p	void *; указатель (представление зависит от реализации)
%	знак процента %

2. Форматный вывод (printf)

Ширину и точность можно специфицировать с помощью *****; значение ширины (или точности) в этом случае берется из следующего аргумента (который должен быть типа **int**). Например, печать не более **max** символов из строки **s** :

```
printf("%.s", max, s);
```

Примеры (печать строки **hello, world** – 12 литер, “.” условно показывает границы поля):

<code>%s</code>	<code>:hello, world:</code>
<code>%10s</code>	<code>:hello, world:</code>
<code>%.10s</code>	<code>:hello, wor:</code>
<code>%-10s</code>	<code>:hello, world:</code>
<code>%.15s</code>	<code>:hello, world:</code>
<code>%-15s</code>	<code>:hello, world :</code>
<code>%15.10s</code>	<code>: hello, wor:</code>
<code>%-15.10s</code>	<code>:hello, wor :</code>

2. Форматный вывод (printf)

Функция **sprintf** выполняет те же преобразования, что и **printf**, но вывод запоминает в строке

```
int sprintf(char *string, char *format, arg1, arg2, ...)
```

Заметим, что строка **string** должна быть достаточно большой, чтобы в ней поместился результат.

3. Форматный ввод (**scanf**)

Функция **scanf**, обеспечивающая **ввод**, является обратным аналогом **printf**; она выполняет многие из упоминавшихся преобразований, но в противоположном направлении. Объявление функции:

```
int scanf(char *format, arg1, arg2, ...)
```

Функция **scanf** читает символы из стандартного входного потока, интерпретирует их согласно спецификациям строки **format** и рассылает результаты в свои остальные **аргументы, которые являются указателями.**

В качестве результата **scanf** возвращает количество успешно введенных элементов данных. По исчерпанию файла она выдает **EOF**.

3. Форматный ввод (**scanf**)

Функция **scanf** прекращает работу, когда оказывается, что исчерпан формат или вводимая величина не соответствует управляющей спецификации.

Существует также функция **sscanf**, которая читает из строки (а не из стандартного ввода):

```
int sscanf(char *string, char *format, arg1, arg2, ...)
```

Функция **sscanf** просматривает строку **string** согласно формату **format** и рассылает полученные значения в **arg1**, **arg2** и т. д. Последние должны быть указателями.

Спецификация:

%[*] [ширина] [символ-формата]

***** - поле ввода пропускается и присваивание не выполняется.

3. Форматный ввод (scanf)

Таблица – Описание значений поля **символ_формата**

символ_формата	тип поля; вводимые данные
c	char * ; единичная литера
s	char * ; строка
d, u	int * ; десятичное целое
i, o, x	int * ; целые (i – p=8 или p=16)
e, f, g	float * ; вещественное число с плавающей точкой

Перед символами-формата **d, i, o, u** и **x** может стоять буква **h**, указывающая на то, что соответствующий аргумент должен иметь тип **short *** (а не **int ***), или **l**, указывающая на тип **long ***.

Аналогично, перед символами-спецификаторами **e, f** и **g** может стоять буква **l**, указывающая, что тип аргумента – **double *** (а не **float ***).

3. Форматный ввод (scanf)

Пример:

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

```
main ( ) { /* программа-калькулятор */
```

```
    double sum, v;
```

```
    sum = 0;
```

```
    while (scanf ("%lf", &v) == 1)
```

```
        printf ("\t%.2f\n", sum += v);
```

```
    return 0; }
```

1	1.00
0.1	1.10
0.2	1.30
0.3	1.60
^Z	

Предположим, что нам нужно прочитывать из потока ввода:

26 декабря 1928

Обращение к `scanf` выглядит следующим образом:

```
int day, year; /* день, год */
```

```
char monthname[10]; /* название месяца */
```

```
scanf ("%d %s %d", &day, monthname, &year);
```

При вводе `scanf` игнорирует пробелы и табуляции.

Ввод-вывод

```
char *gets (char *s) ;
```

считывает строку из стандартного потока ввода и помещает ее в массив указанный аргументом **s**. Чтение строки производится пока не будет встречен символ '**\n**', или не будет достигнут конец файла.

Если чтение строки завершилось по считыванию символа '**\n**', то символ '**\n**' не записывается в массив, а заменяется символом '**\0**' .

Если при чтении данные произошла ошибка, то возвращается **NULL**.

Применение этой функции не рекомендуется, так как размер считываемой строки *не ограничивается* размером массива, в который должна быть записана считываемая строка. В результате может возникнуть переполнение массива и несанкционированная запись в память, что приведет к ошибкам в работе программы или ее зависанию.

```
int puts (const char *s) ;
```

Функция **puts** выводит строку в стандартный поток вывода. После вывода строки производится переход на новую строку. Символ конца строки (нулевой символ) не выводится. Возвращает **EOF** - в случае ошибки или не отрицательное число, если вывод прошел успешно.

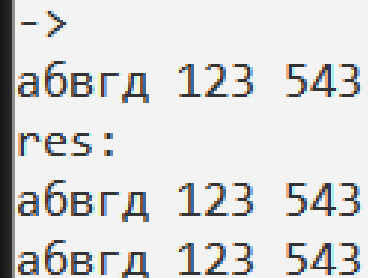
Ввод-вывод

Пример:

```
#include <stdio.h>
int main () {
    char masstr[100]=" ";
    char *rstr;
    printf ("->\n") ; //Запрос ввода строки

    //Чтение строки из стандартного потока ввода
    rstr = gets (masstr) ;

    //Вывод результата работы
    printf ("res:\n%s\n",masstr) ;
    puts (masstr) ;
    return 0 ;
}
```



```
->
абвгд 123 543
res:
абвгд 123 543
абвгд 123 543
```