Язык программирования Си

Строковые литералы, ввод-вывод
Ветвления

Югов Иван Олегович
МОУ Гимназия №10, г. Тверь

Строковые литералы

Указываются в кавычках:

`"Test #2"`; `"Press any key..."`.

Пустая строка: `""`.

В тексте литерала единичные кавычки недопустимы, а двойные игнорируются:

`"2°19'59" в. д."` — недопустимо;

`"Роман ""Тихий Дон"""` → `Роман Тихий Дон`.

Кавычки нужно *экранировать* служебным символом «\»:

`"Роман \"Тихий Дон\""` → `Роман "Тихий Дон"`.

Строковые литералы

Экранирование — оформление служебных символов так, что они получают буквальное значение.

Символ «\» экранируется самим собой:

`"C:\\bootlog.txt"` → `C:\bootlog.txt`.

Символ «\» также экранирует конец строки:

```
"Программирование \
на языке Си"
```

обозначает строку

`Программирование на языке Си`.

Строковые литералы

Escape-последовательность (управляющая последовательность, УП) — последовательность символов, имеющая особое значение.

Используется для вставки служебных символов.

Начинаются с «\»: `\"`, `\\`, `\'`.

Даёт возможность указывать символы по их восьмеричным (oct) и шестнадцатеричным (hex) кодам:

- `\000`, где `000` — трёхзначный oct-код символа;
- `\x0`, где `0` — hex-код символа (возможно несколько цифр).

`"\065"` → `5`; `"\x35"` → `5`; `"\x74\145\x73\x74"` → `test`.

В коде символов можно указывать меньше цифр, но нежелательно.

Строковые литералы

Примеры:

- После символа `\7` можно написать цифру 4: `"\0074"`.
- При записи `"\74"` получится символ с кодом 64: `>`.
- Нех-запись не органичивается каким-то количеством цифр:

`"\x445"` — предупреждение о переполнении строковой константы. Результат не определён.

Строковые литералы

Как записать символ `5`
сразу после символа `\x44`?

```
"\x44""5"
```

Строковые литералы

Стандартом также определены:

УП	Символ	Код
`\a`	Сигнал (alarm)	7
`\b`	Забой (backspace)	8
`\f`	Новая страница (form feed)	13
`\n`	Новая строка (line feed)	10
`\r`	Возврат каретки (carriage return)	12
`\t`	Горизонтальная табуляция	9
`\v`	Вертикальная табуляция	11

Строковые литералы

Примеры результата на экране:

`"Information\b\bcs"` → `Informatics`;

`"A\tB\tC"` → `A  B  C`.

Использование символа `\n` в литерале

```
"Л. Н. Толстой\n\"Война и мир\""
```

даёт на экране переход на следующую строку:

```
Л. Н. Толстой
"Война и мир".
```

Перед прочими символами «\» игнорируется:

`\h` → `h`, `\N` → `N`.

Строковые литералы

Каждый символ занимает в памяти 1 байт.

Любая строковая константа заканчивается символом с кодом 0 (*«нуль-терминатором»*).

Все функции работы со строками это знают, поэтому не обрабатывают символы после первого же «нуль-терминатора» в строке.

Явно указывать «нуль-терминатор» не нужно.

Строковые литералы

Символьная константа — в апострофах:

```
'A'; '$'; '\''; '\077'; '\n'.
```

Занимает 1 байт:

- не может быть пустым;
- «нуль-терминатором» не заканчивается.

Может храниться в переменной типа **char**:

```
char P = 'A';
```

Строковые литералы

Возможна поддержка *«широких» (wide) символов и строк* — по 2 или 4 байта на символ.

«Широкие» литералы — с префиксом `L`:

```
L'Ы'

L"Lorem ipsum dolor sit amet..."
```

«Широкий» символ может храниться в переменной **`int`** (**`short`**, **`long`**):

```
short P = 'Ф';
```

Ввод-вывод

Используется заголовочный файл stdio.h.

Для его использования пишем в начале кода:

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

Основные операции:

- ввод: `scanf`, `fscanf`, `wscanf` и т. п.;
- вывод: `printf`, `fprintf`, `wprintf` и т. п.

Вывод (printf)

Функция `printf` выводит на стандартный поток вывода строковое выражение. Возвращает количество выведенных символов.

```
printf("Hello world!\n");
```

Полный формат вызова `printf`:

```
printf(Форматная строка, параметры);
```

Форматная строка — особая строковая константа, задающая формат вывода остальных параметров.

Вывод (printf)

Пример вывода с форматной строкой:

```
printf("Время: %d часов %d минут", H, M);
```

В форматной строке символ «%» является служебным. Экранируется сам собой:

```
printf("Загрузка завершена на 100%%");
```

Со знака «%» начинаются УП, задающие формат вывода для каждого из остальных параметров:

%ФлагиШиринаТочностьДлинаТип

Вывод (printf)

В форматных УП обязателен только *Тип*:

%c	для символов	K
%d или **%i**	для целых знаковых чисел	240
%u	для целых беззнаковых чисел	75
%o	для целых беззнаковых чисел (oct)	176
%x или **%X**	для целых беззнаковых чисел (hex)	4d3, 4D3
%f	для вещественных чисел (обычная форма записи)	451.68
%e или **%E**	для вещественных чисел (экспоненциальная форма записи)	4.5168e+2, 4.5168E+2
%g или **%G**	для вещественных чисел (более краткая форма записи из двух)	451.68
%s	для строк	Yes

Вывод (printf)

Флаги определяют параметры вывода:

-	выравнивать значение по левому краю в пределах предоставленного места (по умолчанию — по правому)
+	выводить знак у положительных чисел
пробел	выводить пробел перед положительными числами
#	• для **%o**, **%x** и **%X** выводить соответственно с **0**, **0x** и **0X** впереди для ненулевых значений; • для **%f**, **%e**, **%E**, **%g** и **%G** выводить десятичную точку, даже если нет дробной части; • для **%g** и **%G** также выводить лидирующие нули
0	дополнять числа нулями слева

Пример: `printf("Value %+i at %#X", Val, Cell);`

Флаги можно комбинировать: `printf("%#+X", U);`

Вывод (printf)

Ширина — число: какой минимум знаков отвести под значение:

```
printf("%6i", Value);
```

Либо символ «*» — количество знаков указано в дополнительном параметре перед основным:

```
printf("%*i", Length, Value);
```

Если число короче, то остаток заполняется пробелами (или нулями, если есть флаг `0`):

```
printf("[%12i]", 125); → [          125]
```

Если число длиннее, то оно выводится целиком:

```
printf("[%3i]", 142857); → [142857]
```

Вывод (printf)

Точность — точка и следующее за ним число:

- для целых (`%i`, `%d`, `%u`, `%o`, `%x`, `%X`) — минимум знаков под значение, остаток заполняется нулями; *точность* `0` означает вывод пустой строки для нулевого значения;
- для `%f`, `%e`, `%E` — число десятичных знаков;
- для `%g`, `%G` — максимальное число значащих цифр;
- для `%s` — только указанное количество первых символов.

Точность по умолчанию — `.1`. Символы «.*» означают задание точности отдельным параметром.

Примеры:

```
printf("[%3.2d] [%.2f]", 3, 5.376); → [ 03] [5.38]
printf("[%.0d] [%.*f]", 0, 1, 2.39); → [] [2.4]
```

Вывод (printf)

Длина — уточняет длину типов:

- `h` — для целых (`%i`, `%d`, `%u`, `%o`, `%x`, `%X`), тип «короткий» (**`short`**);
- `l` — для целых (`%i`, `%d`, `%u`, `%o`, `%x`, `%X`), тип «длинный» (**`long`**);
- `ll` — для целых (`%i`, `%d`, `%u`, `%o`, `%x`, `%X`), тип «очень длинный» (**`long long`**);
- `L` — для вещественных (`%f`, `%e`, `%E`, `%g`, `%G`), тип «длинный» (**`long double`**).

Вывод (printf)

Пример:

```
char A = -1; unsigned char B = 1;
short C = -2; unsigned short D = 2;
long E = -3; unsigned long F = 3;
long long G = -4; unsigned long long H = 4;
double I = 5.0; long double J = 6.0;
char K = 'a';
printf("%d %u %hd %hu %li %lu %lli %llu %f %Lf \
%c %s", A, B, C, D, E, F, G, H, I, J, K, "!!!");
```

Вывод на экран:

```
-1 1 -2 2 -3 3 -4 4 5.000000 6.000000 a !!!
```

Вывод (printf)

Пример:

```
printf("%+07.*lF\n", 2, 3.14159265359);
printf("%.4s\n", "Computer");
printf("%c%c%c\n", 65, 66, 67);
printf("%f %f\n", 0.0 / 0.0, 5.0 / 0.0);
```

Вывод на экран:

```
+003.14
Comp
ABC
-nan inf
```

Ввод (scanf)

Полный формат вызова `scanf`:

```
scanf(Форматная строка, параметры);
```

Возвращает количество успешно считанных величин.

Форматная строка — как у `printf`.

В УП указывается *тип*. Можно указать *ширину* (максимальное количество знаков для считывания) и *длину* значения.

Остальные параметры — соответствующее количество адресов (переменных), в которые будут считаны данные.

Ввод (scanf)

Унарная операция `&` — *взятие адреса*.

Получает адрес, начиная с которого, в памяти хранится переменная: `&A`, `&Result`.

```
scanf("%Lf %Lf %Lf", &x, &y, &z);
```

Форматная строка может содержать другие символы кроме УП. Тогда необходимо водить и их:

```
scanf("R%i", &N);
```

(ожидается ввод, например, `R12`, `R-46`...)

Исключение — пробелы, переводы строк, табуляции.

Ветвления

Синтаксис ветвления:

```
if(условие)
   команда1;
else
   команда2;
```

Вычисляет значение условия.

Если условие истинно (ненулевое), тогда выполняется команда 1, иначе — команда 2.

Ветвления

Раздел **`else`** может отсутствовать.

Можно указать несколько команд — в `{}`:

```
if(условие)
{
   командаA1; командаA2; командаA3;...
}
else
{ командаB1; командаB2; командаB3;...}
```

Точка с запятой после команды обязательна, после *операторных скобок* — нет.

Ветвления

Пример:

```
if(A == B || B == C || A == C)
   if(A == B && B == C)
      printf("Треугольник равносторонний\n");
   else
      printf("Треугольник равнобедренный\n");
else
   printf("Треугольник общего вида\n");
```