7. Ввод-вывод

Владимир Верстов

Б. Керниган, Д. Ритчи. Язык программирования C

- Средства ввода и вывода не являются частью языка

- Средства ввода и вывода — часть стандартной библиотеки

- Стандарт ANSI регламентирует стандартную библиотеку

Модель ввода и вывода

- Текстовый поток — последовательность строк, каждая строка завершается символом перехода на новую строку

- Стандартная библиотека делает все возможное, чтобы для программы это всегда было так, даже если система работает по-другому

Стандартные потоки

- Стандартный поток ввода — Standart Input, `stdin`. Чаще всего, это ввод с клавиатуры в консоль

- Стандартный поток вывода — Standart Output, `stdout`. Чаще всего, это вывод в консоль

Перенаправление потоков

- `prog < file` — вместо `stdin` будет использовано содержимое файла `file`

- `prog > file` — вместо `stdout` результат будет записан в файл `file`

- `prog1 | prog2` — вывод `prog1` заменит `stdin` для `prog2`. Это конвейер, `pipe`.

- `getchar` — считывание символа из `stdin`
- `putchar` — печать символа в `stdout`

Форматированный ВЫВОД

Функция `printf`

- Сигнатура: `int printf(char *format, ...)`
- `...` — список аргументов переменной длины
- Функция возвращает количество выведенных символов
- `format` — строка формата или формат вывода

Спецификация формата

- Спецификация начинается с `%` и заканчивается символом типа
- Между `%` и типом могут быть:
  - ➡ `-` — выравнивание по левому краю
  - ➡ `число`, задающее минимальную ширину поля. При необходимости поле дополняется пробелами слева или справа
  - ➡ `.` — отделяет ширину поля от точности представления
  - ➡ `число`, задающее максимальное количество символов при выводе строки или количество цифр после запятой
  - ➡ буквы `h` или `l` — если число нужно вывести как `short` или `long`

Символ	Тип аргумента и способ вывода	Пример
d, i	int, десятичное целое число	-10
o	int, восьмеричное число без знака	17
x, X	int, шестандцатиричное число	AF1
u	int, десятичное целое число без знака	124
c	int или char, символ	a
s	char *, строка	Abc
f	float или double	2222
e, E	float или double	1,23E+23
g, G	float или double	1,23E+23 или 2.2
p	void *, любой указатель	AAFFEECC12
%	символ %	%

Вывод в строку

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

Форматированный ВВОД

Функция `scanf`

- Сигнатура: `int scanf(char *format, ...)`
- Аналог `printf` для ввода
- Возвращает количество считанных переменных:
  - ➡ `0` означает, что ввод не соответствует формату
  - ➡ `EOF` означает конец ввода
- Все **аргументы должны быть указателями**!!!
- **Пропускает** все **символы пустого пространства**!!!

Формат состоит из:

- Пробелы или табуляции, которые игнорируются при вводе
- Обыкновенные символы (не `%`), которые соответствуют ожидаемым негустым символам
- Спецификация формата состоит из **`%`**, запрета присваивания `*`, максимальной ширины поля, символов `h` или `l` и символа **`типа`**

Символ	Тип аргумента и способ вывода	Пример
d	десятичное целое число	-10
i	целое число	11 / 012 / AF
u	десятичное целое число без знака	124
o	восьмеричное число без знака	17 или 017
x	шестандцатиричное число	AF1 или 0xFF
c	символ, в том числе пустого пространства. Если `%lc` — первый не пустой символ	a
s	строка, указатель char *	Abc
e, f, g	вещественное число	2.22 или 1.2E+23
%	символ %	%

Ввод из строки

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

Доступ к файлам

Файловый указатель

- `FILE *` — указатель на структуру, которая описывает файл или файловый указатель.

- Структура `FILE` объявлна в `stdio.h`

- Структура `FILE` содержит: местонахождение буфера, текущую позицию в буфере, характер операций (чтение или запись), наличие ошибок и состояние конца файла

Работа с файлом

```c
FILE *fp;

char *name = "test.txt";
char *mode = "r";

fp = fopen(name, mode);

// что-то делаем с файлом

fclose(fp);
```

- `name` — строка, которая содержит имя или путь до файла

- `mode` — режим работы с файлом:

  ➡ `r` — чтение

  ➡ `w` — запись

  ➡ `a` — добавление в конец

  ➡ `b` — бинарный файл

- `stdin` и `stdout` — константы типа `FILE *`

Функция	Описание
`int getc(FILE *fp)`	• Возвращает очередной символ из файла • Функция `getchar` эквивалентна `getc(stdin)`
`int putc(int c, FILE *fp)`	• Печатает очередной символ в файл • Функция `putchar` эквивалентна `putc(c, stdin)` • Возвращает записанный символ или `EOF` в случае ошибки
`int fscanf(FILE *fp, char * format, …)`	• Аналог `scanf` и `sscanf`
`int fprinf(FILE *fp, char * format, …)`	• Аналог `printf` и `sprintf`
`char *fgets(char *line, int n, FILE *fp)`	• Чтение `n` символов из файла `fp` в строку `line` • В конце файла или в случае ошибки возвращает `NULL`
`int fputs(char *line, FILE *fp)`	• Запись строки `line` в файл `fp` • Возвращает `EOF` в случае ошибки или `0` в противном случае