Лекция VI

22 ноября 2017

Ввод - вывод: как работает

Ввод/вывод: что происходит

?

Данные в программе:
* Текст: **char**
* Числа: **int, double, size_t**

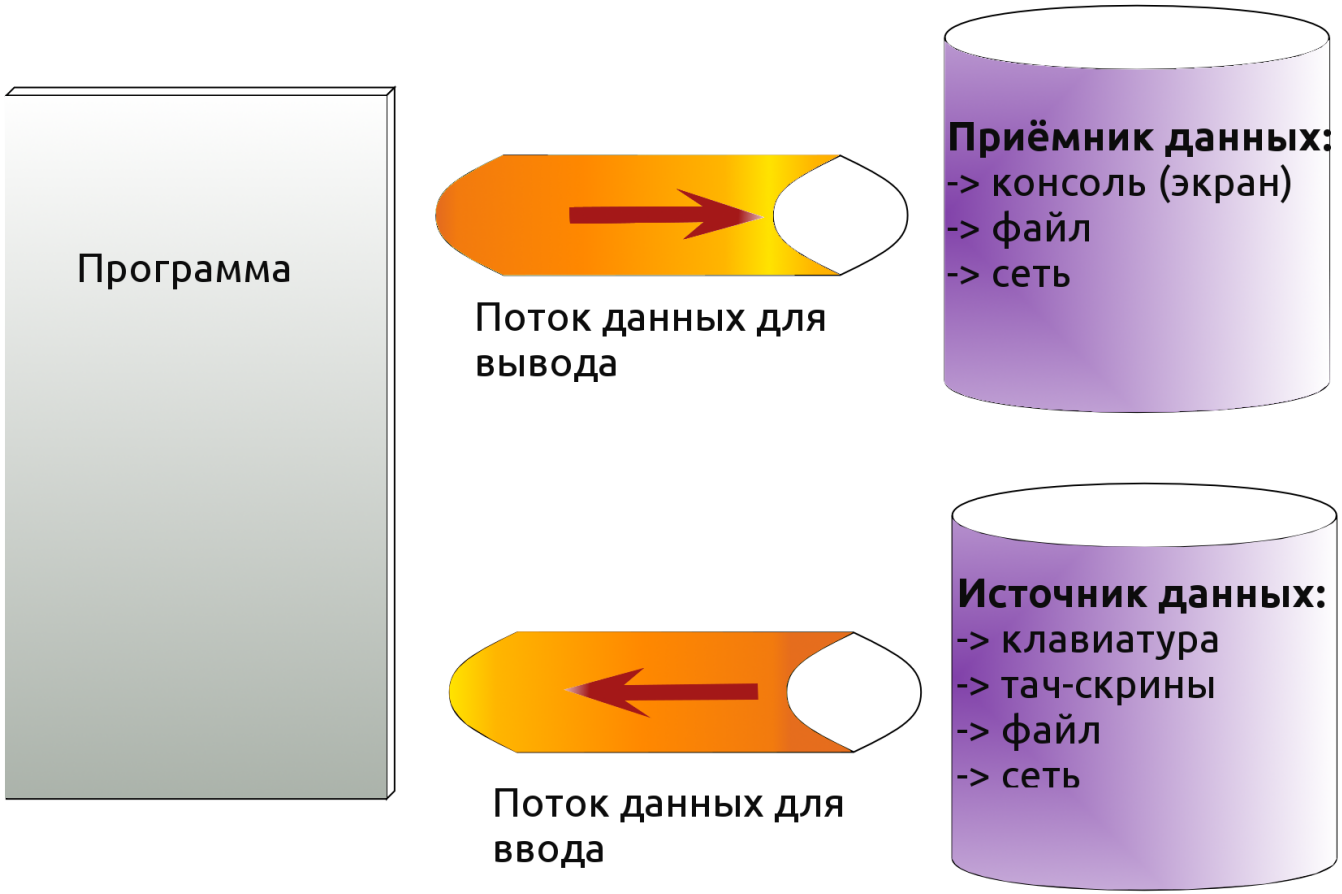
Данные снаружи:
* Байты
  -> текст (ASCII, UTF8, ...)
  -> бинарные данные (raw bytes)

Ввод/вывод: кто работает

Концепция **потока данных (stream)**

- Логическая сущность, связывающаяся с одним приёмником или источником данных
- Берёт на себя обязанности по приёму/отправке байт извне
- Преобразует байты в значения требуемых типов при вводе; преобразует значения различных типов в байтовое представление при выводе
- Скрывает взаимодействие с реальным устройством ввода-вывода

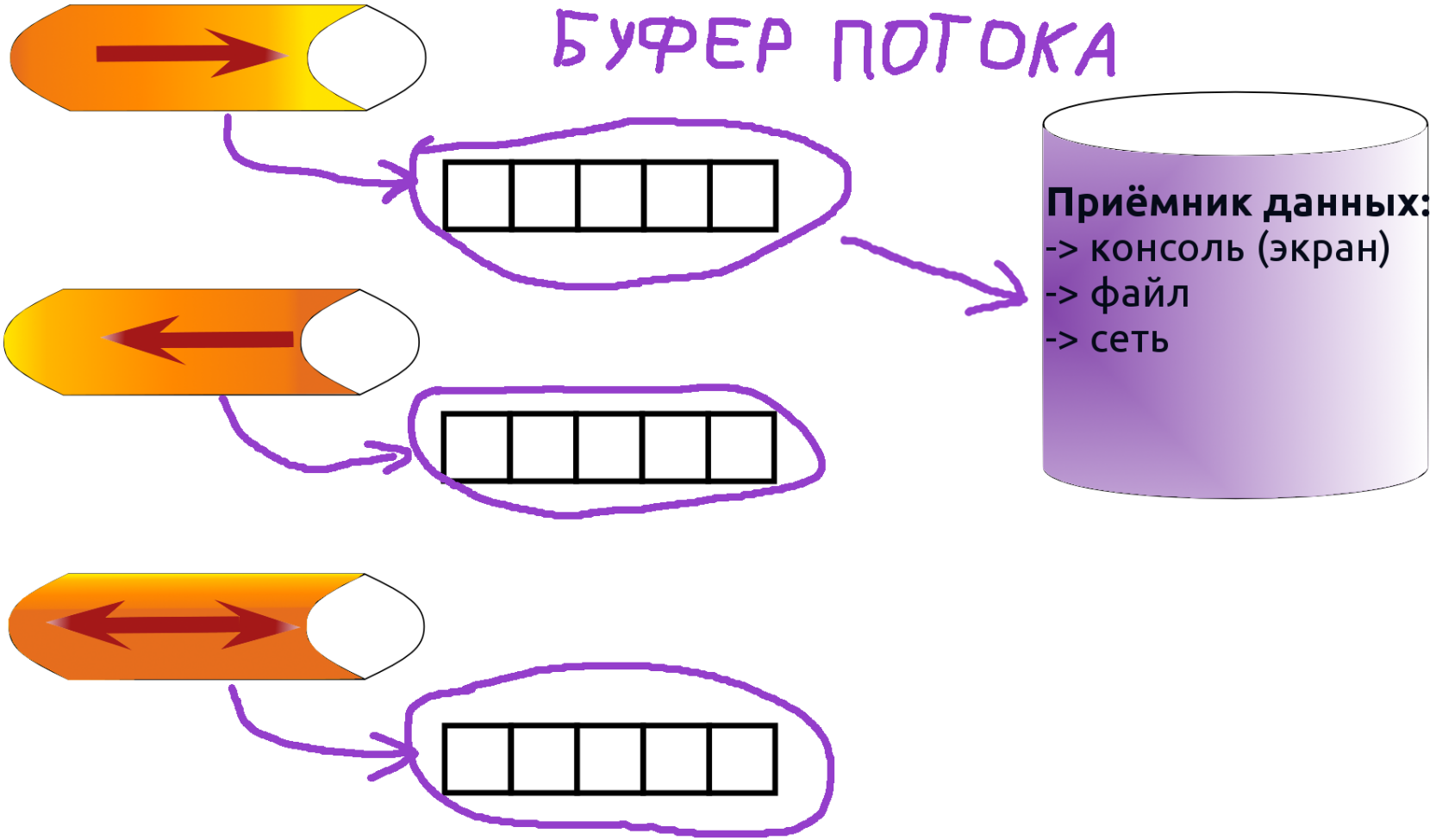
Потоковый ввод/вывод



Данные в программе:
* Текст: **char**
* Числа: **int, double, size_t**

Данные снаружи:
* Байты
  -> текст (ASCII, UTF8, ...)
  -> бинарные данные (raw bytes)

Потоковый ввод/вывод



Потоковый ввод/вывод

Как правило, для каждой исполняемой программы на языке C при запуске создаются три потока

stdout: printf, putc, puts

stdin: scanf, getc, fgets

stderr: perror

Потоковый ввод/вывод: файлы

Функции для создания файловых потоков ввода-вывода определены в библиотеке **<stdio.h>**.

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

Тип, с помощью которого происходят все манипуляции с потоком, это структура **FILE**

Полезная константа, объявленная внутри библиотеки - **EOF** (конец файла, как правило целое значение, равное -1)

Потоковый ввод/вывод: файлы

Функция **fopen** - создание потока для работы с конкретным файлом

```
FILE* fopen(const char *file_name,
            const char *mode);
```

- 1-ая строка **file_name** - название файла (полный путь)
- 2-ая строка **mode** - режим работы потока, состоящий из **типа операции** (ввод, вывод, всё вместе) и **дополнительных модификаторов** (читать информацию как текст или двоичные данные)
- Возращаемое значение: указатель на структуру **FILE** (который и является потоком), если создание потока прошло успешно, и **NULL** - в противном случае

Потоковый ввод/вывод: файлы

Функция **fopen** - режимы открытия (1):

mode	Что означает
"r", "rb"	**read**: создание потока для ввода (чтения) данных. Файл **должен** существовать
"w", "wb"	**write**: создание потока для вывода (записи) данных. Если файл уже существует, то всё его содержимое **удаляется**. Если не существует - создаётся новый файл
"a", "ab"	**append**: создание потока для вывода данных в конец файла (дозапись). Если файл не существует - создаётся новый. Невозможно записать в произвольное место в файле

Потоковый ввод/вывод: файлы

Функция **fopen** - режимы открытия (2):

mode	Что означает
"r+", "rb+"	**update**: создание потока для одновременного ввода/вывода. Файл **должен** существовать, выводить данные можно в любое место файла
"w+", "wb+"	**update**: одновременный ввод/вывод. Создаётся новый файл, либо удаляется содержимое существующего
"a+", "ab+"	**update**: одновременный ввод/вывод, но операции вывода осуществляются **только** в конец файла

Потоковый ввод/вывод: файлы

Функция **fclose** - закрытие существующего файлового потока (прекращение связи с файлом)

```
int fclose(FILE *stream);
```

- Аргумент **stream** - указатель на поток, подлежащий закрытию. **Не пытайтесь** передать в функцию нулевой указатель.
- Возвращаемое значение: **нуль**, если закрытие прошло успешно; значение **EOF** - в противном случае

Потоковый ввод/вывод: файлы

Создание/закрытие файловых потоков: общий шаблон для работы

```
1  FILE *f_in_data = NULL;
2
3  f_in_data = fopen("my_data_file.txt", "r");
4  if (f_in_data != NULL) {
5    // Получаем данные из файла
6    // в какие-нибудь переменные
7    fclose(f_in_data);
8  } else {
9    perror("Ошибка открытия файла");
10 }
```

Ввод/вывод данных

↙ ↘

Форматированный:
Для чтения: каждая группа символов (разделяемых пробелами) преобразуется в значение требуемого типа.
Для записи: значения конкретного типа преобразуется в текстовый вид самим потоком, и отправляется в файл.

Неформатированный:
читается/записывается строго определённое количество байт (указываемое в программе) и никаких форматных преобразований не происходит

Потоковый вывод в файлы

Функция **fprintf** - форматирование и вывод значений всех фундаментальных типов (и строк) в заданный файл

```
int fprintf(FILE *stream,
            const char *format_str, ...);
```

- Аргумент **stream** - указатель на поток, в который осуществляется запись. Должен быть открыть в соответствующем режиме
- Аргумент **format_str** - форматная строка, содержащая произвольные символы и спецификаторы выводимых значений (**%**-последовательности)
- **...** - переменное количество аргументов, равное количеству указанных в **format_str** спецификаторов, и соответствующие значения для вывода
- Возращаемое значение: в случае успеха - количество записанных байт; иначе - некоторое отрицательное значение. **Традиционно, *очень важный параметр***

Потоковый вывод в файлы

Пример **fprintf**

```
1  FILE *f_out = NULL;
2
3  f_out = fopen("some_results.txt", "w");
4  if (f_out != NULL) {
5    int i_num = 567;
6    double r_num = 14.8326372364277;
7    char str[] = "Как всё просто";
8
9    fprintf(f_out, "Целое число: %+08d\n", i_num);
10   fprintf(f_out, "  Десятичное: %+.5f\n", r_num);
11   fprintf(f_out, "Строка: {{%s}}\n", str);
12   fprintf(f_out, "%d * %d = %d", 4, 5, 20);
13
14   fclose(f_out);
15 } else {
16   perror("Ошибка открытия файла");
17 }
```

Потоковый вывод в файлы

Пример **fprintf**: в файле "some_results.txt"окажется текст:

```
Целое число: +0000567
  Десятичное: +14.83264
Строка: {{Как всё просто}}
4 * 5 = 20
```

Для справки

Функция **fopen** при открытии файла на запись, создаёт его в той же директории, откуда запускается программа. Создавать поддиректории, увы, невозможно стандартной библиотекой ввода-вывода.

Более подробно про вывод значений - https://github.com/posgen/OmsuMaterials/wiki/Format-output-in-C

Потоковый вывод в файлы

Функция **fputc** - запись одного символа (значения типа **char**) в файл

```
int fputc(int character, FILE *stream);
```

- Аргумент **character** - символ для записи. Принимается целочисленный код символа, но рекомендуется ограничиваться символами из таблицы ASCII
- Аргумент **stream** - указатель на поток, в который происходит запись
- Возвращаемое значение: в случае успеха - записанный символ; иначе - константа **EOF**

Потоковый вывод в файлы

Пример **fputc** - запись цифр в файл

```
1  FILE *f_digits = NULL;
2
3  f_digits = fopen("all_digits.dat", "w");
4  if (f_digits != NULL) {
5    for (char sym = '0'; sym <= '9'; ++sym) {
6      fputc(sym, f_digits);
7    }
8
9    fclose(f_digits);
10 } else {
11   perror("Ошибка открытия файла");
12 }
```

Потоковый вывод в файлы

Функция **fputs** - запись строк в файл

```
int fputs(const char *str, FILE *stream);
```

- **str** - строка для записи. Функция записывает всю переданную строку (до символа окончания строки). В отличии от **puts**, *не добавляет* символа переноса '\n'
- **stream** - указатель на поток, в который происходит запись
- Возращаемое значение: в случае успеха - некоторое неотрицательное число; иначе - константа **EOF**

Потоковый вывод в файлы

Пример **fputs**

```
1  FILE *f_log = NULL;
2
3  f_log = fopen("logfile.dat", "w");
4  if (f_log != NULL) {
5    fputs("1-ые вычисления - успешно\n", f_log);
6
7    fputs("2-ые вычисления - успешно\n", f_log);
8
9    fputs("3-ые вычисления - неудача\n", f_log);
10
11   fclose(f_log);
12 } else {
13   perror("Ошибка открытия файла");
14 }
```

Потоковый вывод в файлы

Пример **fputs**: в файле "logfile.dat"окажутся три строки:

```
1-ые вычисления - успешно
2-ые вычисления - успешно
3-ые вычисления - неудача
```

Потоковый ввод из файлов

Функция **fscanf** - форматированный ввод значений различных типов из файлового потока

```
int fscanf(FILE *stream,
           const char *format_str, ...);
```

- **stream** - указатель на поток, из которого осуществляется чтение
- **format_str** - форматная строка, содержащая произвольные символы и спецификаторы вводимых значений (**%**-последовательности)
- **...** - переменное количество аргументов, равное количеству указанных в **format_str** спецификаторов, и соответствующие адреса переменных для записи вводимых значений
- Возращаемое значение: в случае успеха - количество записанных значений, равное количеству дополнительных аргументов; иначе - либо константа **EOF**, либо число меньшее количества доп. аргументов

Потоковый ввод из файлов

Пример **fscanf**: дан файл my_data.txt

```
45    678.905
1.2387E-3
Строка - просто строка
```

```
1  FILE *f_data = NULL;
2
3  f_data = fopen("my_data.txt", "r");
4  if (f_data == NULL) {
5    perror("Ошибка открытия файла"); exit(1);
6  }
7
8  int num1; double real1, real2;
9  fscanf(f_data, "%d %lf", &num1, &real1);
10 fscanf(f_data, "%le", &real2);
11
12 printf("Целое: %d, вещественные: %f, %f\n",
13        num1, real1, real2);
14 //... продолжение на следующем слайде
```

Потоковый ввод из файлов

Пример **fscanf**: дан файл my_data.txt

```
45    678.905
1.2387E-3
Строка - просто строка
```

```
1 //... начало на предыдущем слайде
2
3 char word[50];
4 fscanf(f_data, "%50s", word);
5
6 puts("Первое слово из файла:");
7 puts(word);
8
9 fclose(f_data);
```

Более подробно про ввод значений - https://github.com/posgen/OmsuMaterials/wiki/Format-output-in-C

Потоковый ввод из файлов

Функция **fgetc** - ввод одного символа из файла

```
int fgetc(FILE *stream);
```

- **stream** - указатель на поток, из которого осуществляется чтение
- Возращаемое значение: в случае успеха - код текущего символа в файле; иначе - константа **EOF**

Потоковый ввод из файлов

Пример **fgetc**: есть файл text.txt:

```
Файл для посимвольного
чтения информации; #1 2# #3 #4 5
#...#
```

```
1  FILE *f_text = NULL;
2  f_text = fopen("text.txt", "r");
3  if (f_text == NULL) { ... }
4
5  int symb, sharp_count = 0;
6
7  do {
8    symb = fgetc(f_text);
9    putc(symb);
10
11   if (symb == '#') { ++sharp_count; }
12 } while (symb != EOF);
13
14 fclose(f_text);
15 printf("Найдено %d решёток\n", sharp_count);
```

Потоковый ввод из файлов

Функция **fgets** - ввод строки из файла

```
char* fgets(char *str, int max_count, FILE *stream)
```

- **str** - указатель на массив типа **char**, в который записываются извлекаемые символы
- **max_count** - максимально возможное количество символов, для записи в **str** с учётом символа окончания строки. Символ переноса строки '\n' - также может быть записан в **str**
- **stream** - указатель на поток, из которого осуществляется чтение
- Возращаемое значение: в случае успеха - указатель на **str**; иначе - **NULL**

Потоковый ввод из файлов

Пример **fgets**: дан файл vip_text.txt с текстом:

```
В теоретической механике широко применяются методы

векторного исчисления и дифференциальной геометрии,
математического анализа,
дифференциальных уравнений,
вариационного исчисления.
```

Потоковый ввод из файлов

Пример **fgets**: дан файл vip_text.txt

```
1  FILE *f_text = NULL;
2  f_text = fopen("vip_text.txt", "r");
3  if (f_text == NULL) { ... }
4
5  char buf[30];
6
7  while (fgets(buf, 30, f_text) != NULL) {
8    printf("<<%s>>\n", buf);
9  }
10
11 fclose(f_text);
```

Потоковый ввод из файлов

Пример **fgets**: считывание всего текстового файла в строку

```
1  char* fread_all_lines(const char *f_name) {
2    FILE *f_text = NULL;
3    f_text = fopen(f_name, "r");
4    if (f_text == NULL) { return NULL; }
5
6    const size_t SZ = 256; size_t cur_sz = SZ;
7    char buf[SZ], *str = NULL;
8
9    while (fgets(buf, SZ, f_text) != NULL) {
10     str = (char *) realloc(str, cur_sz * sizeof(↩
           char));
11     strcat(str, buf); cur_sz += SZ;
12   }
13   fclose(f_text);
14
15   str = (char*) realloc(str, strlen(str) + 1);
16   return str;
17 }
```

Потоковый ввод из файлов

Пример **fgets**: считывание всего текстового файла в строку

```
1 char file_name[] = "my_text.txt";
2 char *str = fread_all_lines(file_name);
3
4 /*
5   Что-нибудь делаем с текстом
6 */
7
8 // Не забываем удалить динамическую память
9 free(str);
```