Рекомендуемый порядок выполнения Л.Р. 5 по этапам.

0. Пустая программа.
   Каждой лабе -- отдельный каталог.
    Называть или по номеру лабы (lab5, lab_05 ...)
     или по названию программы (fputofs, fgetofs_my_name).
     Не рекомендуется использовать пробелы, специальные или локальные символы.
     Пример:
       mkdir fpostofs
       cd fputofs
   Скачать с http://metal/docs.html "Простой Makefile для C и C++",
     сохранить в созданном каталоге под именем "Makefile".
     В этом файле заменить (vim Makefile) название главной цели
     (помечено "^----- change here")
     на имя программы без суффиксов, например
     PGM=fputofs
   Создать файл программы с суффиксом 'c'
     с выводом 'Hellow world';
     vim fputofs.c

     #include <stdio.h>

     int main(int argc, char **argv)
     {
       printf("Hellow world\n");
       return 0;
     }

     Сохранить (:w), собрать (:make), запустить (:! ./fputofs)
     Если надо, исправить ошибки.

1. Обработка аргументов командной строки.
   Прочитать условие лабы (скачать там же, где и Makefile).
   Подсчитать, сколько должно быть аргументов (включая argv[0] - имя программы).
   Проверить правильное значение argc.
    Если неправильное --
     выругаться (fprintf(stderr,"...",...))
     и уйти (return 1;).
   Проверить первый аргумент на совпадение со строкой "-h" (strcmp).
    Совпало -- выдать синтаксис запуска программы и уйти (return 0;)
   Можно для отладки выводить argc, argv[..]
   Первый аргумент (argv[1]) -- имя файла -- уже строка,
     просто переписываем указатель в локальную переменную.
   Второй и третий -- преобразуем из строки в число (strtol),
     сохраняем, выводим.
   Читать маны: strcmp(3), strtol(3), fprintf(3)

2. Открытие и закрытие файла.
   Из условия понять, как нам надо открыть файл.
   Открыть файл (open), передав правильный набор параметров,
    сохранив возвращаемое значение (прочитать про него).
   Проверить на ошибки. Если есть --
     выругаться (printf(stderr, ...)), сообщив имя файла

```
     и причину ошибки ( strerror(errno) )
     и выйти (return 2;)
   Закрыть файл (close), проверив на ошибки (если только надо).
   Проверить программу, задавая различные имена файлов
   и проверяя на ошибки ( /etc/shadow, /xxx/zzz )
   Читать маны: open(2) close(2), strerror(3).

3. Чтение или запись простой строки.
   Если по условию из файла читаем -- создать файл
    с текстом для проверки. Если пишем -- тоже не помешает.
   Описать размер буфера в 16 (#define или const int).
   Определить буфер из char заданного размера.
   Если по условию пишем в файл -- скопировать в буфер простую строку
     и записываем в файл (write).
   Если по условию читаем из файла --
     читаем чуть-чуть байт (read)  и выводим.
   Не забывать обработку ошибок и конца файла.
   Проверить работу.
   Читать маны: read(2) write(2), strcpy(3).

4. Продумать и реализовывать циклический алгоритм
   чтения и записи. Помнить, что read и write
   могут вернуть значение меньшее, чем запрошено байт.
   Использовать специальные файловые дескрипторы (0, 1 ).
   Проверить на размерах меньше буфера, чуть больше буфера,
   и больше размера файла.

5. Перед циклом чтение/записи смещаемся в файле (lseek).
   Проверяем.
```