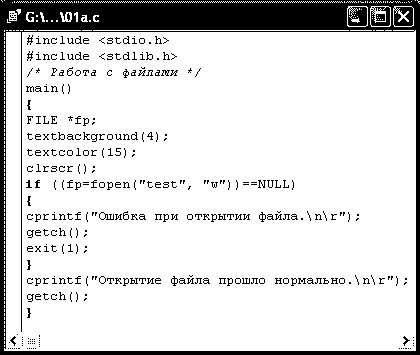
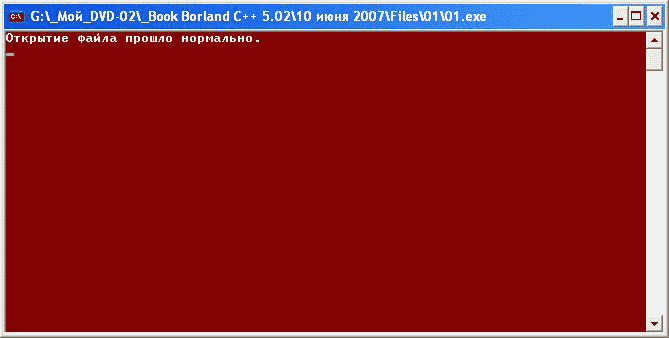
Известно, что в языке Си система ввода/вывода информации реализуется не с помощью ключевых слов, а с помощью библиотечных функций. Благодаря этому система ввода/вывода является очень мощной и гибкой. Во время работы с файлами данные могут передаваться или в своем внутреннем двоичном представлении или в текстовом формате, то есть в более удобочитаемом виде.   
  
    Язык Си является фундаментом С++. При этом С++ поддерживает всю файловую систему Си. Поэтому при использовании С-кода в С++ нет необходимости менять процедуры ввода/вывода. Хотя при написании программ на С++ обычно более удобно использовать именно систему С++. Это касается, в частности, и использования "iostream.h" взамен "stdio.h", реализующим ввод/вывод. Однако, не будем забегать вперед и пообещаем, что в свое время в соответствующем месте упомянутые вопросы будут рассмотрены, то есть это будет сделано на соответствующих уроках. Сейчас же наберемся терпения, и изучим файловый ввод/вывод в языке Си. Тем более, что это само по себе очень интересно и очень важно для понимания "потоков" и "файлов" как в Си, так и в С++.   
  
    На этом уроке и в дальнейшем при изучении языка Си мы будем придерживаться именно файловой системы языка Си.   
  
    В системе ввода/вывода в Си для программ поддерживается единый интерфейс, не зависящий от того, к какому конкретному устройству осуществляется доступ. То есть в Си между программой и устройством находится нечто более общее, чем само устройство. Такое обобщенное устройство ввода или вывода (устройство более высокого уровня абстракции) называется **потоком**. В то же время конкретное устройство называется **файлом**. Наша задача - понять, каким обрзом происходит взаимодействие **потоков** и **файлов.**   
  
    **3. Потоки в Си.**   
  
    Файловая система Си предназначена для работы с разными устройствами, в том числе с терминалами, дисководами и накопителями на магнитной ленте. Даже, если какое-то устройство очень сильно отличается от других устройств, буферизованная файловая система все равно представит его в виде логического устройства, которое называется **потоком**. Все потоки ведут себя похожим образом. И так как они в основном не зависят от физических устройств, то та же функция, которая выполняет запись в дисковый файл, может ту же операцию выполнить и на другом устройстве. Например, на консоли. **Потоки** бывают двух видов: **текстовые** и **двоичные**.   
  
    **4. Файлы в Си.**   
  
    В языке Си файлом может быть все, что угодно, начиная в дискового файла и заканчивая терминалом или принтером. Поток связывают с определенным файлом, выполняя обязательную операцию **открытия**. Как только файл открыт, можно проводить обмен информацией между ним и программой.   
  
    Но не у всех файлов одинаковые возможности. Например, к дисковому файлу прямой доступ возможен, в то время как к некоторым принтерам - он не возможен. Таким образом, вы видите, что напрашивается определенный вывод, являющийся принципом системы ввода/вывода языка Си: все потоки одинаковы, а файлы - нет!   
  
    Если файл может поддерживать запросы на местоположение (указатель текущей позиции), то при открытии такого файла указатель текущей позиции в файле устанавливается в начало файла. При чтении каждого символа из файла (или записи в файл) указатель текущей позиции увеличивается. Тем самым обеспечивается продвижение по файлу.   
  
    Файл отсоединяется от определенного потока (то есть разрывается связь между файлом и потоком) с помощью операции **закрытия файла**. При закрытии файла, открытого с целью вывода, содержимое (если оно, конечно, есть) связанного с ним потока записывается на внешнее устройство. Этот процесс обычно называют **дозаписью** потока. При этом гарантируется, что никакая информация случайно не останется в буфере диска.   
  
    Если программа завершает работу нормально, то есть либо **main()** возвращает управление операционной системе, либо выход происходит через **exit()**, то все файлы закрываются автоматически.   
  
    В случае же аварийного завершения работы программы, например, в случае краха или завершения путем вызова **abort()**, файлы не закрываются.   
  
    У каждого потока, связанного с файлом, имеется управляющая структура, содержащая информацию о файле. Она имеет тип **FILE**. Блок управления файлом - это небольшой блок памяти, временно выделенный операционной системой для хранения информации о файле, который был открыт для использования. Блок управления файлом обычно содержит информацию об идентификаторе файла, его расположении на диске и указателе текущей позициив файле.   
  
    Для тех, кто начинает изучать язык Си потоки и файлы кажутся одним и тем же, и их разграничение может показаться излишним или даже подчас черезчур "заумным". Но не спешите с выводами. Помните, что основная цель такого разграничения - обеспечить единый интерфейс операций ввода/вывода в Си. Для выполнения всех операций ввода/вывода следует использовать только понятия потоков и применять всего лишь одну файловую систему. Ввод или вывод от каждого устройства автоматически преобразуется системой в легко управлемый поток. И это является достижением автора языка Си.   
  
    Таковы основополагающие замечания относительно существования различных потоков информации и связанных с ними файлов.   
  
    **5. Основы файловой системы в Си.**   
  
    Файловая система языка Си состоит из нескольких взаимосвязанных между собой функций. Для их работы в Си требуется заголовочный файл **<stdio.h>** и такой же аналогичный ему заголовочный файл **<iostream.h>** требуется для работы в С++.   
  
    Ниже приведена таблица основных (часто используемых) функций файловой системы языка Си.

|  |  |
| --- | --- |
| Имя | Что делает эта функция |
| fopen() | Открывает файл |
| fclose() | Закрывает файл |
| putc() | Записывает символ в файл |
| fputc() | То же, что и putc() |
| getc() | Читает символ из файла |
| fgetc() | То же, что и getc() |
| fgets() | Читает строку из файла |
| fputs() | Записывает строку в файл |
| fseek() | Устанавливает указатель текущей позиции на определенный байт файла |
| ftell() | Возвращает текущее значение указателя текущей позиции в файле |
| fprintf() | Для файла то же, что printf() для консоли |
| fscanf() | Для файла то же, что scanf() для консоли |
| feof() | Возвращает значение true (истина), если достигнут конец файла |
| ferror() | Возвращает значение true (истина), если произошла ошибка |
| rewind() | Устанавливает указатель текущей позиции в начало файла |
| remove() | Стирает файл |
| fflush() | Дозапись потока в файл |

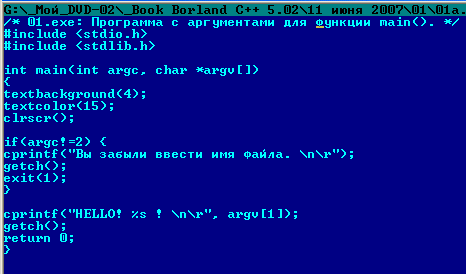
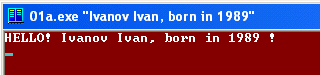
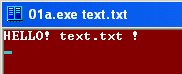
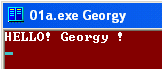
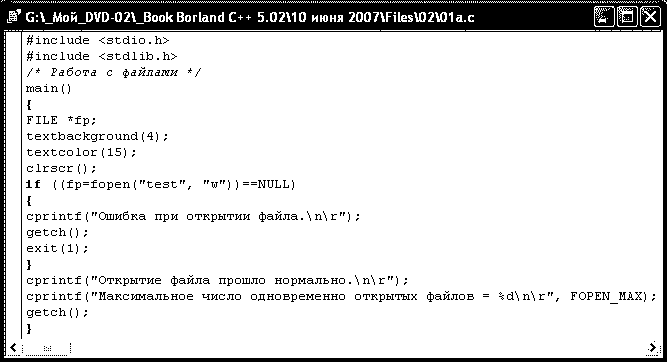
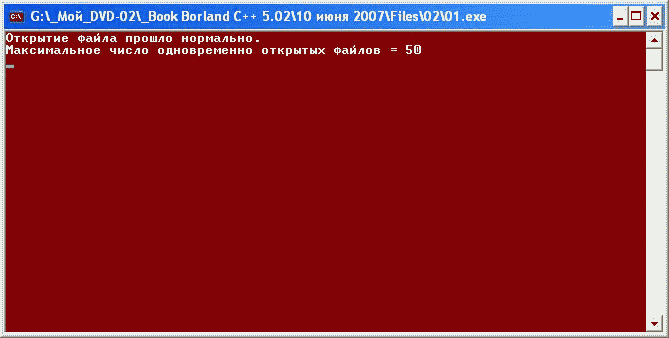
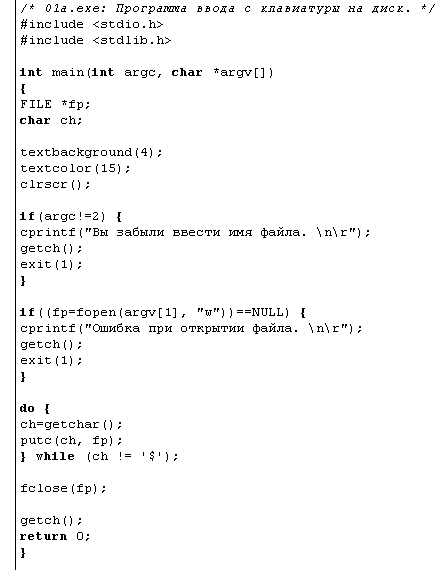
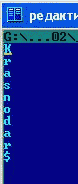
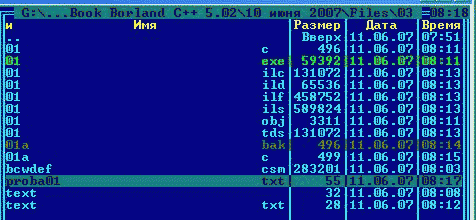
    Заголовок **<stdio.h>** представляет прототипы функций ввода/вывода в Си и определяет следующие три типа: **size\_t**, **fpos\_t** и **FILE**. Первые два: **size\_t**, **fpos\_t** представляют собой разновидности такого типа, как целое без знака. Отдельно рассмотрим третий тип: **FILE**.   
  
    **6. Указатель файла FILE.**   
  
    Указатель файла - это то, что соединяет в единое целое всю систему ввода/вывода языка Си. **Указатель файла** - это указатель на структуру типа **FILE**. Он указывает на структуру, содержащую различные сведения о файле, например, его имя, статус, и указатель текущей позиции в начало файла. В сущности указатель файла определяет конкретный файл и используется соответствующим потокомпри выполнениифункции ввода/вывода.   
  
    Чтобы выполнять в файлах операции чтения и записи, программы должны использовать указатели соответствующих файлов. Чтобы объвить переменную-указатель файла необходимо использовать следующий оператор:   
  
    **FILE \*fp;   
  
    7. Открытие файла.   
  
    Функция fopen() открывает поток и всязывает с этим потоком файл. Затем она возвращает указатель этого файла. Прототип функции имеет вид:  
  
   FILE \*fopen(const char \*имя\_файла, const char \*режим);   
  
    Здесь имя\_файла - это указатель на строку символов, представляющую собой допустимое имя файла, в которое может входить спецификация файла (включает обозначение логического устройства, путь к файлу и собственно имя файла).   
  
    Режим - определяет, каким образом файл будет открыт. Ниже в таблице показаны допустимые значения режимов.**

|  |  |
| --- | --- |
| Режим | Что обозначает данный режим |
| r | Открыть текстовый файл для чтения |
| w | Создать текстовый файл для записи |
| a | Добавить в конец текстового файла |
| rb | Открыть двоичный файл для чтения |
| wb | Создать двоичный файл для записи |
| ab | Добавить в конец двоичного файла |
| r+ | Открыть текстовый файл для чтения/записи |
| w+ | Создать текстовый файл для чтения/записи |
| a+ | Добавить в конец текстового файла или создать текстовый файл для чтения/записи |
| r+b | Открыть двоичный файл для чтения/записи |
| w+b | Создать двоичный файл для чтения/записи |
| a+b | Добавить в конец двоичного файла или создать двоичный файл для чтения/записи |

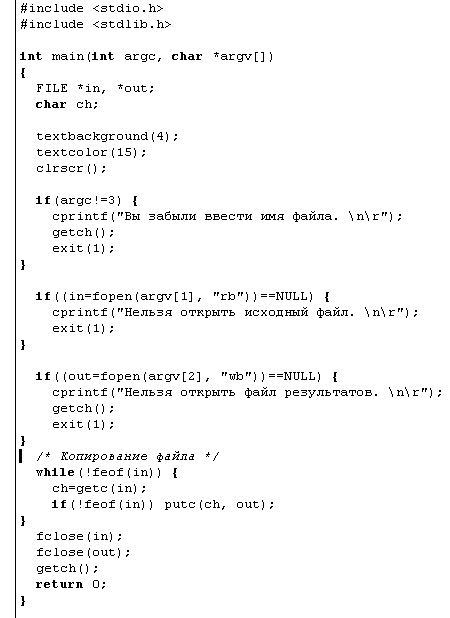
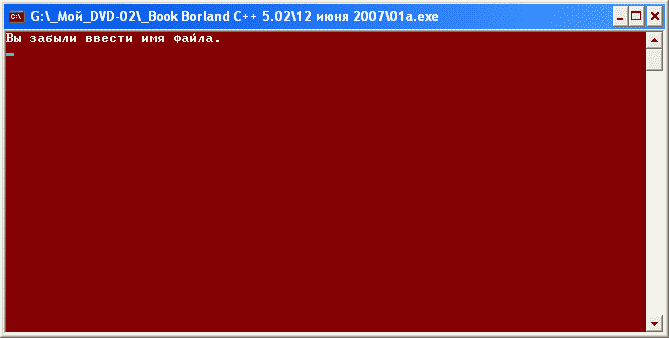
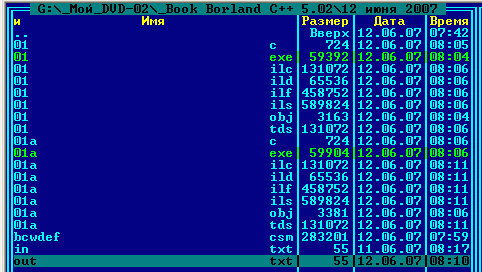
**Приведем фрагмент программы, в котором используется функция fopen() для открытия файла по имени TEST.   
  
   FILE \*fp;  
   fp = fopen("test", "w");  
  
    Следует сразу же указать на недостаточность такого кода в программе. Хотя приведенный код технически правильный, но его обычно пишут немного по-другому.   
  
   FILE \*fp;  
   if ((fp = fopen("test", "w")==NUL)  
   {  
   printf("Ошибка при открытии файла.\n\r")"  
   exit(1);  
   }  
  
    Этот метод помогает при открытии файла обнаружить любую ошибку. Например, защиту от записи или полный диск. Причем, обнаружить еще до того, как программа попытается в этот файл что-то записать. Поэтому всегда нужно вначале получить подтверждение, что функция fopen() выполнилась успешно, и лишь затем выполнять c файлом другие операции. Ниже Вы видите небольшую часть программы, которая подтверждает или не подтверждает открытие файла.   
  
  
  
**

**Урок 76:** [**Запись, чтение символа. Закрытие файла.**](http://ipg.h1.ru/lessons/ci/les76.html)   
[(автор урока Климант Юрий Викторович.)](http://ipg.h1.ru/aboutme/aboutme.html)

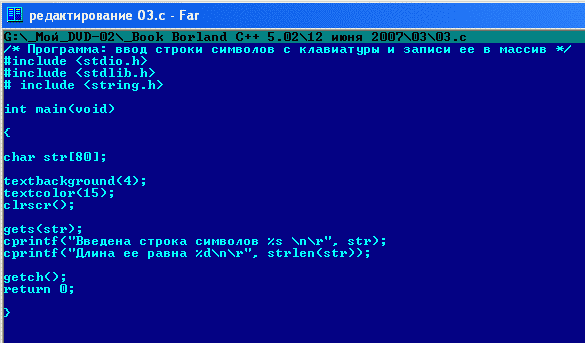
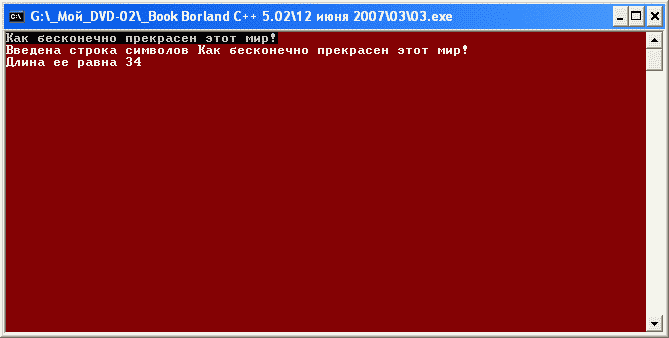
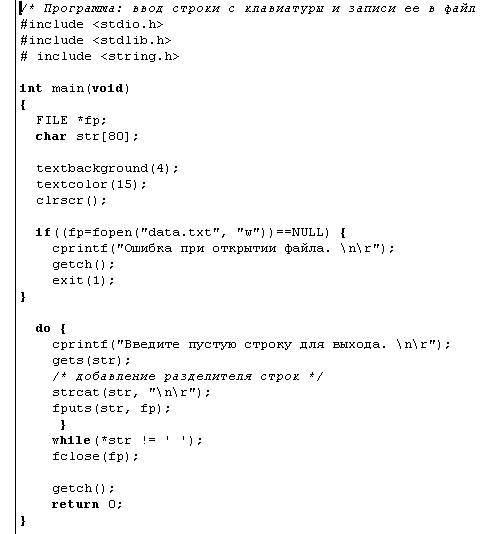
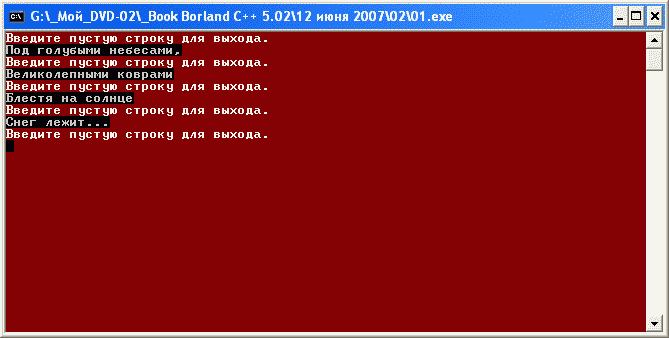
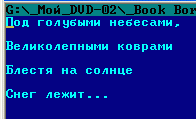
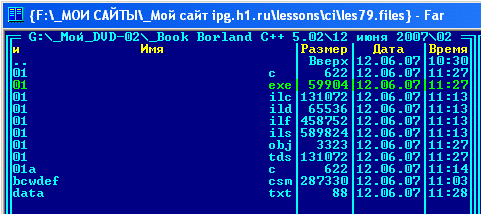
    **1. Цель урока.**   
  
    Наша цель - продолжить изучение функций ввода/вывода в языке Си. На уроке будут рассмотрены функции:   
  
   Запись символа в файл.  
   Чтение символа из файла.  
   Закрытие файла.   
  
   Приступим...   
  
    **2. Максимальное число одновременно отУрок 77:** [**Аргументы функции main() в языке Си.**](http://ipg.h1.ru/lessons/ci/les77.html)   
[(автор урока Климант Юрий Викторович.)](http://ipg.h1.ru/aboutme/aboutme.html)

    **1. Цель урока.**   
  
    Наша цель - изучить аргументы функции main(). Предыдущий урок показал Вам, что можно создать программу, которая при вызове будет вводить имя создаваемого файла для записи информации на диск. Это очень интересный материал. Его нужно хорошо понять, усвоить и применять в своей работе при написании программ, которые должны одновременно с вызовом воспринимать дополнительную информацию, такую, например, как имя файла, в который будет записываться вся вводимая с клавиатуры текстовая информация.   
  
    **2. Аргументы функции main() в языке Си.**   
  
    На уроке 76 Вы могли заметить, что можно, обращаясь с командной строки к программе, передать ей некоторые важные сведения для дальнейшей работы. Такими сведениями было, в частности, имя файла, который использовался в программе для записи символов, вводимых с клавиатуры. Обращение к программе было таким: 01.exe proba01.txt для ввода текста "Краснодарский край" и 01.exe text.txt для ввода текста "Krasnodar". Метод включал передачу информации в программу при ее вызове через использование аргументов функции main().   
  
В языке Си можно использовать следующие два встроенные аргумента функции main(): это argc и argv. В результате получается следующее обращение к функции main() для вызова этой функции с аргументами: int main(int argc, char \*argv[]). Существует еще и третий аргумент функции main(). Это аргумент char \*env[]. Два первых аргумента используются для передачи аргументов командной строки. Третий аргумент позволяет организовать доступ к параметрам среды операционной системы. Эти три аргумента доступны только для функции main().   
  
    Параметр argc содержит количество аргументов командной строки и является параметром типа int. Он всегда не меньше 1, так как имя программы, вызываемой для выполнения трактуется как первый параметр командной строки.   
  
    Параметр argv является указателем на массив строк. Каждый элемент массива указывает на аргументы командной строки. Один параметр отделяется от другого пробелом. Поэтому, если в Вашем сообщении, которое Вы хотите передать программе, предполагается иметь пробелы, то Ваше сообщение должно быть заключено в кавычки. Например, если Вы хотите передать Вашей программе следующий текст: "Ivanov Ivan, born in 1989", то этот текст нужно сопровождать кавычками, так как в самом тексте уже есть пробелы. Покажем это, используя специально написанную небольшую программу.   
  
  
  
    В этой программе предусмотрено обращение к функции main(): int main(int argc, char \*argv[]). Если Вы не указали никакой дополнительной информации при обращении к программе 01.exe, то Вам будет выдано следубщее сообщение об этом: Вы забыли ввести имя.   
  
http://ipg.h1.ru/lessons/ci/les77.files/01a.png  
  
    В противном случае, программа поприветствует Вас следующим образом: HELLO! Ivanov Ivan, born in 1989. Вы можете наблюдать это на приложенных рисунках с результатами работы программы.   
  
  
  
    Аналогично, если вызвать программу 01.exe с указанием имени файла, например, так: 01.exe text.txt, то Вы получите на выходе как результат работы программы следующее сообщение, показанное ниже на рисунке:   
  
  
  
    Если же попытаться вызвать программу 01.exe и дать ей дополнительную информацию, например, Georgy, то Вы получите на выходе как результат работы программы следующее сообщение, показанное на рисунке:   
  
  
  
    Таким образом, Вы познакомились с существующей в языке Си возможностью задавать дополнительную информацию программе непосредственно при вызове программы прямо из командной строки. Это можно использовать для информирования программы также об имени файла, который будет использован Вами для сохранекния некоторой информации. Так было сделано Вами в программе, рассмотренной на уроке 76, и снова этот факт рассмотрен более детально и подтвержден на данном уроке. Что ж, очень хорошо. Вы, надеюсь, научились пользоваться такой прекрасной возможностью, предусмотренной в языке Си.   
  
    На этом урок закончен **открытых файлов.**   
  
    В языке Си существует ограничение на максимальное число одновременно открытых файлов. Это значение не меньше 8. Но более точно Вы узнаете об этом из документации, сопровождающей компьютер. Максимальное число одновременно открытых файлов определяется **FOPENMAX**. Давайте попытаемся извлечь значение **FOPENMAX** и выдадим его на экран дисплея. Для этой цели немного подправим маленькую программку предыдущего урока. Добавим в нее команды распечатки интересующего нас показателя. Посмотрим, что у нас получится из этой затеи.   
  
  
  
  
  
    Компьютер нам выдал, что максимальное число одновременно открытых файлов = 50 ! Итак, получилось. Теперь, совершенно осмелев, можно проводить эксперименты и дальше. Последуем этому. Введем в ту же программку необходимые дополнения для записи и чтения отдельного символа, а затем и большого текста. Но продолжим, как и прежде, с объяснения новых для Вас функций записи и чтения символа.   
  
    **3. Функции языка Си: запись символа в файл.**   
  
    Судя по таблице часто используемых функций из предыдущего урока, в файловой системе ввода/вывода языка Си предусмотрены две эквивалентные функции для ввода символа: **putc()** и **fputc()**. Две идентичные функции существуют для того, чтобы сохранить преемственность со старыми версиями Си.   
  
    Функция **putc()** записывает символы в файл, который уже был заранее открыт функцией **fopen()**. Прототип этой функции имеет вид:  
  
   **int putc(int ch, FILE \*указатель файла);**   
  
    Здесь указатель файла - это возвращенный функцией **fopen()** указатель,   
    **ch** - выводимый в файл символ. Указанный тип **int** позволяет записывать в файл только младший байт слова.   
  
    Если запись прошла удачно, то возвращается записанный символ, в противном случае возвращается **EOF**.   
  
    **4. Функции языка Си: чтение символа из файла.**   
  
    Существуют, как и в предыдущем случае, две функции для чтения символа из файла. Это **getc()** и **fgetc()**. Объясняется наличие двух разновидностей функции чтения из файла необходимостью обеспечить совместимость со старыми версиями Си.   
  
    Функция **getc()** читает символ из файла, который уже был открыт заранее функцией **fopen()**. Прототип этой функции имеет вид:  
  
   **int getc(FILE \*указатель файла);**   
  
    Здесь указатель файла - это указатель, имеющий тип **FILE**, возвращенный функцией **fopen()**.   
  
    Функция getc() возвращает целое значение, но читаемый из файла символ находится в младшем байте. Если не произошло ошибки чтения, то старший байт слова обнуляется. Если обнаружен конец файла, то функция getc() возвращает EOF. Поэтому, чтобы читать символы файла до конца файла, можно использовать следующий программный код:   
  
   do {  
   ch=getc();  
   } while(ch!=EOF);  
  
    Функция getc() возвращает EOF и в случае ошибки. Поэтому для определения причины можно воспользоваться функцией **ferror()**.   
  
    **5. Использование функций языка Си: fopen(), getc(), putc() и fclose().**   
  
    Следующие функции: fopen(), getc(), putc() и fclose() представляют минимальный набор функций, достаточный для работы с файлами. Покажем простой пример программы, работающей с названными функциями. После компиляции получается программа **01.exe**. Остановимся на том, что она может делать?   
  
1. Первое - что необходимо помнить, это следующее: при обращении к программе нужно указывать имя файла для записи информации.  
Например, обращение к программе может иметь вид: **01.exe proba01.txt** или **01.exe text.txt**. Если Вы не укажите имя файла, то программа напомнит Вам об этом сообщением: Вы забыли ввести имя файла.   
  
2. Если открытие файла произошло неудачно, то программы выдаст другое сообщение: Ошибка при открытии файла.   
  
3. Если файл открыт, то программа переходит в ожидание ввода с клавиатуры символа, который Вы хотите записать на диск. Ввод символа с клавиатуры выполняется с помощью функции **getchar()**. Введенный символ сразу же записывается в файл с помощью функции **putc(ch, fp)**. Программе не просто вводит один символ. В программе выполняется цикл "пока":**while**. Если выполняется условие работы цикла, то есть введенный с клавиатуры символ не равен доллару **$**, то программа снова обращается к вводу очередного символа и записи его в файл. Как только Вы введете символ доллара **$** указанный символ также записывается в файл, и дальнейший ввод символов с клавиатуры прекращается. Файл закрывается и программа завершает свою работу.   
  
4. Функция **getch()**, присутствующая а разных местах программы, позволяет задерживать экран до нажатия на любую клавишу. Этот прием позволяет видеть то, что на экране, и свободно читать выдаваемое сообщение. Тем самым ликвидируется поспешное мелькание экрана, при котором пользователь не успевает читать сообщения и вникать в то, что делается в программе и что отражается на экране.   
  
  
  
    Программа опробована на примере создания двух файлов: proba01.txt и text.txt.  
Обращение к программе для создания первого файла имеет вид: **01.exe proba01.txt**.  
Обращение к программе для создания второго файла имеет вид: **01.exe text.txt**.  
Ниже на рисунках показано содержание 1-ого и 2-ого файлов. Это текст: Краснодарский край и Krasnodar.  
После каждого символа, набранного на клавиатуре необходимо нажимать клавишу Enter. Файлы записаны на внешний носитель информации. Это подтверждается распечаткой фрагмента справки из файлового менеждера FAR, которая приведена тут же на рисунке.   
  
http://ipg.h1.ru/lessons/ci/les76.files/03.png                                   
  
    **6. Закрытие файла. Функция fclose().**   
  
    Функция fclose() закрывает поток, который был открыт функцией fopen(). Прототип функции имеет вид:  
  
   int fclose(FILE \*указатель файла);  
  
    Возвращение нуля означает успешно выполенную операцию закрытия файла. В случае ошибки возвращается EOF. Более точно о причине ошибки можно узнать, воспользовавшись функцией ferror(). Обычно отказ при выполнении функции fclose() возникает только тогда, когда диск был преждевременно удален с дисковода или когда на диске нет свободного места для записи файла.   
  
    Цель урока достигнута. Обращаю Ваше внимание на тот факт, что в программе было использовано неизвестное еще Вам обращение к главному модулю программы int main(int argc, char \*argv[]) через аргументы функции main(). Материал об аргументах функции main() будет отдельно рассмотрен на специальном уроке.   
  
    На этом урок закончен.

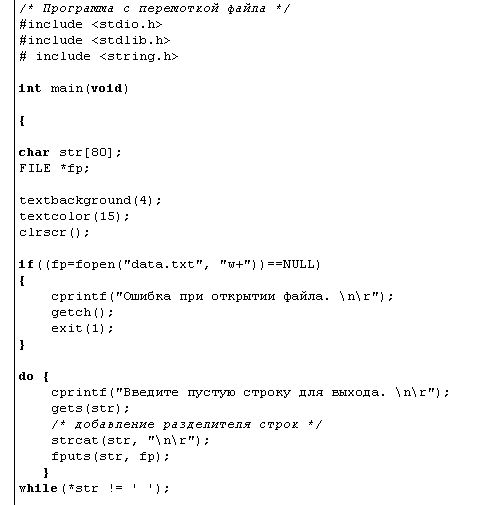
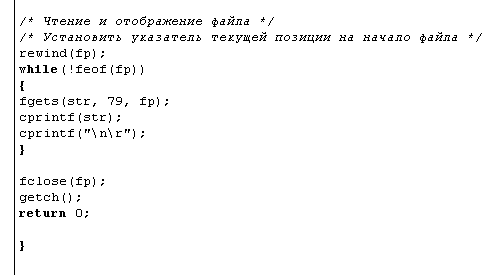
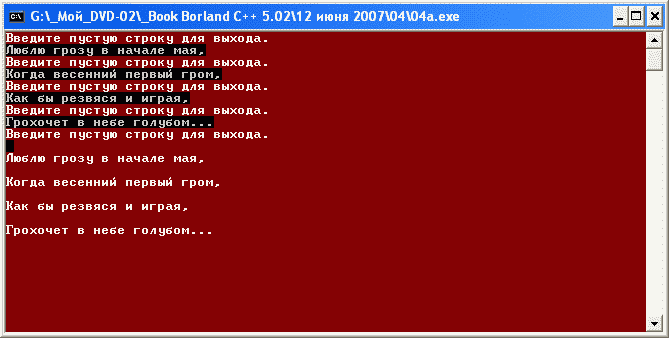
**Урок 78:** [**Функция feof() в языке Си.**](http://ipg.h1.ru/lessons/ci/les78.html)   
[(автор урока Климант Юрий Викторович.)](http://ipg.h1.ru/aboutme/aboutme.html)

    **1. Цель урока.**   
  
    Изучить функцию feof().   
  
    **2. Функция feof().**   
  
    Определяет достигнут ли конец файла. В первом случае, то есть при конце файла возвращается значение true, во втором случае - возвращается значение ноль.   
  
   Прототип функции имеет вид: int feof(FILE \*указатель файла);   
  
    Приведем пример однострочного фрагмента программы, который читает двоичный файл до тех пор, пока не будет достигнут конец файла.   
  
   while(!feof(fp)) ch=getc(fp);   
  
    **2. Программа копирования файла в другой файл.**   
  
    Программа копирования файла in.txt в файл out.txt приведена ниже на рисунке.   
  
  
  
    Обращение к программе для копирования файла in.txt в файл out.txt должно иметь следующий вид:   
  
   01.exe in.txt out.txt  
  
    Если Вы случайно забыли указать имена файлов в обращении к программе копирования или хотя бы одного из них, то программа выдаст Вам об этом сообщение для размышления, какой же на самом деле файл Вы упустили. Это может быть как входной (исходный), так и выходной файл (копия).   
  
  
  
    Если копирование файла пройдет успешно, то об этом Вы сможете узнать по справке, которую можно получить в файловом менеджере FAR. На рисунке ниже показан исходный файл, файл (копия) - результат и приведена справка об изменении содержания директории, в которой до выполнения копирования находился только один исходный файл in.txt. После завершения копирования в этой директории находятся уже два файла - исходный файл in.txt и файл-копия out.txt.   
  
http://ipg.h1.ru/lessons/ci/les78.files/01c.png             http://ipg.h1.ru/lessons/ci/les78.files/01c.png               
  
    На этом урок закончен.

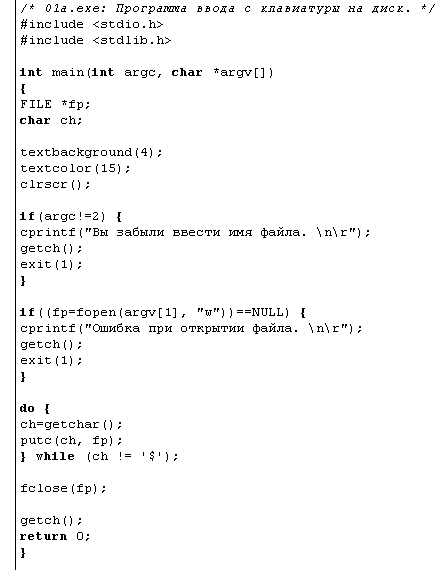
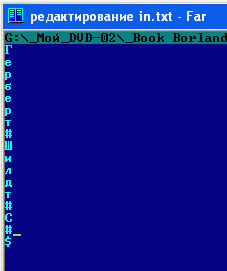
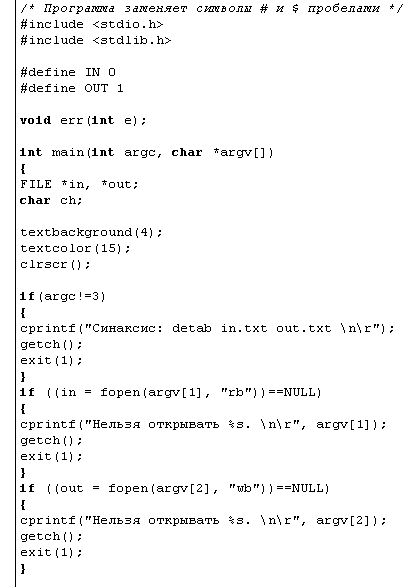
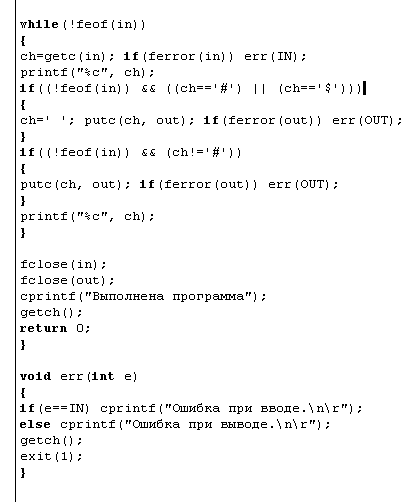
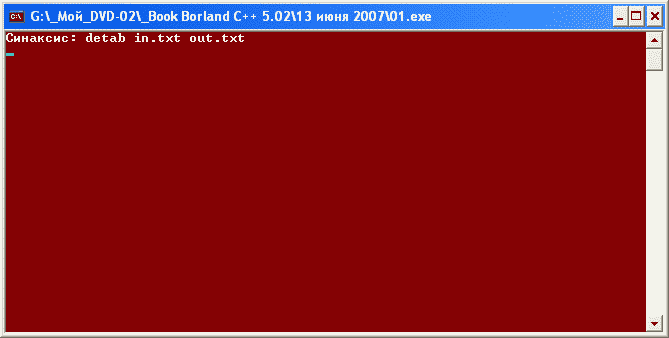
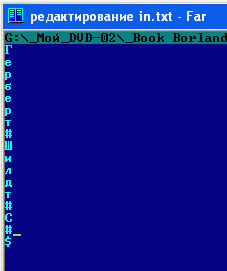
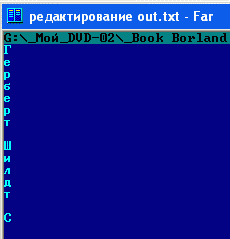
**Урок 79:** [**Ввод/вывод строк. Функции fputs(), fgets() и gets() в языке Си.**](http://ipg.h1.ru/lessons/ci/les79.html)   
[(автор урока Климант Юрий Викторович.)](http://ipg.h1.ru/aboutme/aboutme.html)

    **1. Цель урока.**   
  
    Изучить функции ввода/вывода строк в файл: fputs(), fgets() и функцию gets() в языке Си.   
  
    **2. Функция gets().**   
  
    Функция gets() позволяет читать строку символов при вводе с клавиатуры и записывает ее в память по адресу, на который указывает аргумент. Символы вводят до тех пор, пока не будет введен символ "возврат каретки" или, иными словами, в конце строки должна быть нажата клавиша Enter. Символ "возврат каретки" не станет частью строки. Вместо него в конец строки будет введен символ конца строки - '0'. После этого произойдет возврат из функции gets().   
    Можно до нажатия клавиши Enter исправлять неправильно введенные символы. Для этого пользуются клавишей backspace - возврат каретки на одину позицию назад.   
  
   Прототип функции имеет вледующий вид: char \*gets(char \*str);   
  
    Здесь \*str - это указатель на массив символов, в который записываются символы, вводимые пользователем с клавиатуры. Функция gets() возвращает str. Ниже приведена программа, которая читает с клавиатуры строку символов и записывает ее в массив str - обыкновенную строке Си, состоящую из 80 символов, то есть str[80]. Кроме того, программа выводит на экран еще и длину введенной с клавиатуры строки символов. Смотрите на рисунке результат работы этой программы.   
  
  
  
  
  
    **3. Функции fputs() и fgets().**   
  
    Прототипы функций таковы:   
  
   int fputs(const char \*str, FILE \*указатель файла);   
   char \*fgets(char \*str, int длина, FILE \*увазатель файла);   
  
    Функция fputs() помещает (записывает) в определенный поток строку, на которую указывает str. При наличии ошибки эта функция возвращает EOF.   
  
    Функция fgets() извлекает (читает) из определенного потока строку. Делается это до тех пор, пока не будет прочитан символ новой строки или количество прочитанных (извлеченных) символов не станет равным длина - 1. При успешном завершении работы функция возвращает str. В случае ошибки возвращается пустой указатель (null).   
  
    В приведенной программе показано использование рассмотренных функций fputs() и fgets(). В программе читается строка склавиатуры. Полученная строка записывается в файл data.txt. Завершается программа вводом пустой строки. Далее полученный файл строк закрывается.   
  
    В программе вставляется перед каждой строкой разделитель строк, так как функция gets() не записывает разделитель строк. Это делается для того, чтобы файл легче было читать.   
  
    Программа представлена на рисунке ниже. На последующих рисунках показаны результаты работы программы. В частности, показано распечатанное содержимое записанного на диск файла data.txt и справка из той директории, в которую записан созданный в программе текстовый файл строк data.txt.   
  
  
  
    На этом рисунке показан процесс ввода нескольких стихотворных строк с клавиатуры.   
  
  
  
    На следующих двух рисунках Вы видите текст, который находится в файле data.txt после ввода текста и записи его в файл и полученную в FAR справку о файлах, находящихся в директории, с которой работала программа.   
  
               
  
    На этом урок закончен.

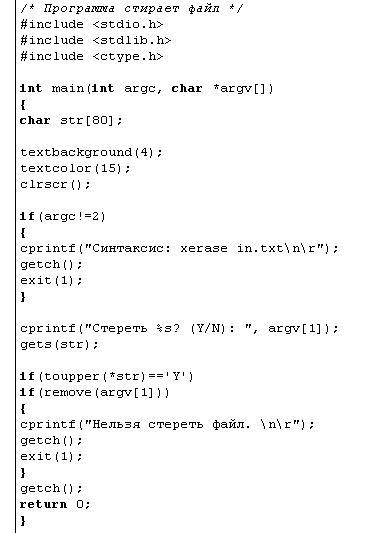
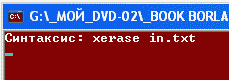
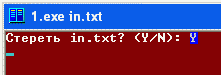
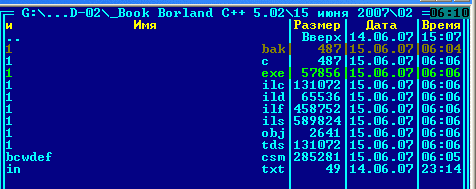
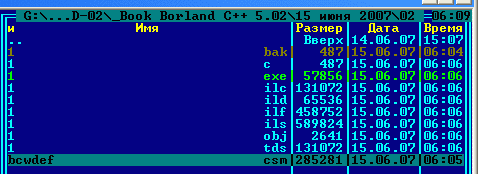
**Урок 80:** [**Функция rewind(). Перемотка файла к началу в языке Си.**](http://ipg.h1.ru/lessons/ci/les80.html)   
[(автор урока Климант Юрий Викторович.)](http://ipg.h1.ru/aboutme/aboutme.html)

    **1. Цель урока.**   
  
    Изучить функцию rewind() - перемотка файла к началу в языке Си.   
  
    **2. Функция rewind().**   
  
    Функция rewind() устанавливает указатель текущей позиции в файле на начало файла, названного в качестве аргумента данной функции. Можно сказать, что функция rewind() выполняет "перемотку" (rewind) файла. Прототип функции имеет вид:   
  
   void rewind(FILE \*указатель файла);   
  
    **3. Программа, использующая функцию "перемотки" файла к началу.**   
  
    Рассмотрим теперь работу этой функции на примере программы, тест которой Вы видите ниже на рисунке. Дадим файлу название data.txt. Файл имеет допустимы режим работы w+. В силу этого он вначале записывается на диск, а затем после "перемотки" (rewind) он читается из той же директории, в которой находится текст программы.   
  
  
  
  
  
    Вначале вводим небольшой стихотворный текст. Далее применим функцию "перемотки" файла к началу и выполним чтение файла и распечатку его строк на экране компьютера. Результат этой работы Вы можете видеть ниже на рисунке.   
  
  
    На этом урок закончен.

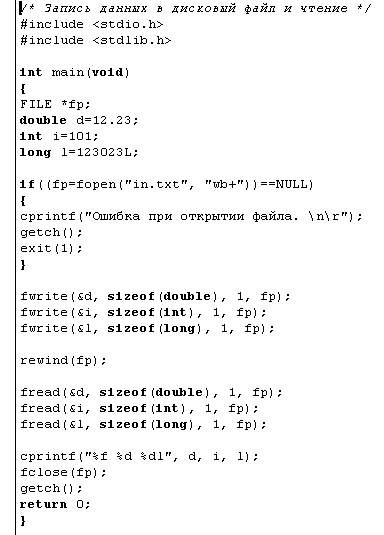
**Урок 81:** [**Функции ferror() в языке Си.**](http://ipg.h1.ru/lessons/ci/les81.html)   
[(автор урока Климант Юрий Викторович.)](http://ipg.h1.ru/aboutme/aboutme.html)

    **1. Цель урока.**   
  
    Изучить функции ferror() в языке Си.   
  
    **2. Функция ferror().**   
  
    Функция ferror() позволяет определить, произошла ли ошибка во время выполнения операции с файлом. Прототип этой функции имеет вид:   
  
   int ferror(FILE \*указатель файла);   
  
    Функция возвращает значение true (истина), если при выполнении операции с файлом произошла ошибка. В противном случае - она возвращает значение false (ложь). После каждой операции с файлом устанавливается свое условие ошибки. Поэтому после каждой операции с файлом следует вызывать функцию ferror() и проверять сведения об ошибке.   
  
    Программа, приведенная ниже, удаляет знаки табуляции из файла и заменяет их соответствующим количеством пробелов. Размер табуляции определяется макросом TAB\_SIZE. В программе функция ferror(0 вызывается после каждой операции с файлом.   
  
    При вызове программы необходимо указывать в командной строке имена входного и выходного файлов.   
  
    **3. Программа, использующая функцию ferror() и функцию err().**   
  
    Задача.   
    В исходном файле in.txt записан текст на русском языке. В этом тексте слова разделены знаком (#). Написать программу, которая заменит знака (#) между словами на пробел и знак доллара ($) в конце текстового файла in.txt также заменит на пробел. Результат запишите в новый файл out.txt.   
  
    Файл in.txt должен быть заранее подготовлен. В этом файле должна уже находиться входная информация c кодами табуляции для программы данного урока. Как ее получить? Вы должны воспользовайться программой предыдущего урока 76 и на ее основе создать входной файл in.txt. На рисунках ниже показаны необходимая для этих целей вспомогательная "Программа ввода с клавиатуры на диск" из урока 76 и весь процесс работы этой программы для получения файла in.txt.   
  
                            
  
    Программа данного урока, использующая функции ferror() и err(), приведена ниже на рисунке. При обращении к ней нужно указывать в командной строке имена входного и выходного файлов. Например, так: 01.exe in/txt out.txt.   
  
    Эта программа должна анализировать исходный файл in.txt на наличие кодов табуляции. При отыскании такого кода каждый код табуляции будут заменяться на соответствующее ему число пробелов.   
  
    Кроме того, обратите внимание: в самом конце программы помещена фунеция err(), которая сообщает о наличии ошибки, обнаруженной при вводе или при выводе данных.   
  
  
  
  
    Если Вы забыли указать имена входного или выходного файлов при обращении к программе 01.exe, то Вам будет выдано соответствующее сообщение, указанное на рисунке ниже.   
  
  
  
    Еще раз напомним, что обращение к программе данного урока должно быть таким: 01.exe in.txt out.txt.   
  
    Процесс замены доллара на пробел и получения из файла in.txt файла out.txt Вы видите на первом рисунке. Здесь же рядом на двух остальных рисунках показаны файлы in.txt и out.txt после работы программы данного урока. Действительно, каждый знак доллара из файла in.txt заменен на пробел и записан в новый файл out.txt, в чем Вы наглядно убеждаетесь сами глядя на рисунки. Все остальные символы файлов не изменились.   
  
                            
  
    На этом урок закончен.

**Урок 82** [**Функции remove() - стереть файл.**](http://ipg.h1.ru/lessons/ci/les82.html)   
[(автор урока Климант Юрий Викторович.)](http://ipg.h1.ru/aboutme/aboutme.html)

    **1. Цель урока.**   
  
    Изучить функции remove() и стереть конкретный файл in.txt.   
  
    **2. Функция remove().**   
  
    Функция стирает указанный в команде файл. Прототип этой функции такой:   
  
    int remove(const char \*имя файла);   
  
    Программа, стирающая файл, указанный в команде remove(const char \*имя файла), показана на рисунке.   
  
  
  
    Обращение к программе имеет вид:   
  
    1.exe in.txt   
  
    Если при обращении к программе не указано имя файла в командной строке, то последует сообщение, приведенное на рисунке.   
  
  
  
    При соблюдении всех условий прежде, чем стереть файл, будет задан вопрос, указанный на рисунке ниже: "Стереть in.txt? (Y/N): "   
  
  
  
    В результате при ответе "Y" файл in.txt будет уничтожен и от него не останется даже следов: одно воспоминание... Это подтверждают два следующих рисунка. На первом из них Вы еще видите файл in.txt.   
  
  
  
    На втором - файл in.txt отсутствует, так как он полностью стерт командой remove(argv[1]), и от него не осталось никаких видимых следов. Здесь в скобках Вы видите argv[1] - имя файла in.txt, принятое через командную строку при обращении к программе.   
  


**Урок 83** [**Функции fread() и fwrite() в языке Си.**](http://ipg.h1.ru/lessons/ci/les83.html)   
[(автор урока Климант Юрий Викторович.)](http://ipg.h1.ru/aboutme/aboutme.html)

    **1. Цель урока.**   
  
    Изучить функции fread() и fwrite() в языке Си.   
  
    **2. Функции fread() и fwrite() в языке Си.**   
  
    В языке Си существует еще две функции, предназначенные для чтения (записи) данных, занимающих более 1 байта. Это могут быть числа с плавающей запятой в формате double, целые числа типа int, а так же целые числа в удвоенном формате. Названные функции позволяют читать и записывать блоки данных любого типа.   
  
    Прототипы этих функций следующие:   
  
    size\_t fread(void \*буфер, size\_t колич\_байт, size\_t счетчик, FILE \*указатель файла);   
    size\_t fwrite(const void \*буфер, size\_t колич\_байт, size\_t счетчик, FILE \*указатель файла);   
  
    Для функции fread() буфер - это указатель на область памяти, в которую будут прочтены данные из файла. Для функции fwrite() буфер - это указатель на данные, которые будут записаны в файл.  
    Значение счетчик - определяет, сколько считывается или записывается элементов данных, причем длина каждого элемента в байтахравна указанному количеству колич\_байт. (Вспомните, что тип size\_t определяется как одна из разновидностей целого типабез знака). Указатель файла - это указатель на уже открытый поток.   
  
    Функция fread() возвращает количество прочитанных элементов. Если достигнут конец файла или произошла ошибка, то возвращаемое значение может быть меньше, чем счетчик. Аналогично, функция fwrite() возвращает количество записанных элементов. Если ошибка не произошла, то возвращаемый результат будет равен значению счетчик.   
  
    Программа для записи данных разных фроматов на диск и чтения их с диска помещена ниже на рисунке.   
  
  
  
    Вначале данные были записаны на диск (но файл не был еще закрыт), затем произошла перемотка файла, и данные читались с диска. Окончательно Вы видите распечатку прочитанных с диска данных. Результат полностью совпадает с данными, которые имелись первоначально. После распечатки файл закрывается.   
  
  
  
    Таким образом, показано применение функций fread() и fwrite() в задаче записи, перемотки и чтения данных разного формата с диска.   
  
    На этом урок закончен.