Элементы ЯПВУ. ФАЙЛЫ

C / C++

C / C++

Язык С не имеет ключевых слов для организации ввода / вывода. Вместо них используются библиотечные функции описанные в заголовочном файле <stdio.h>.
Язык С++ поддерживает две системы ввода / вывода: унаследованную от С и новую, объектно-ориентированную систему, включающую как новые функции, так и операторы ввода/вывода (заголовочный файл <iostream.h>).
Обе системы абсолютно совместимы (в одном исходном файле С++).

Система ввода / вывода языка С.

В С имеются консольные и файловые функции ввода / вывода.
Консольные функции работают с клавиатурой и экраном (и, на самом деле, со стандартными потоками, которые можно переназначать).

Основные функции консольного ввода/вывода:

Функция	Действие	Примечания
getchar ();	Читает символ с клавиатуры и отображает на экране	Обычно ожидает Enter
getche ();	Читает символ с клавиатуры и отображает на экране	Не ожидает Enter. В стандарте С не определена.
getch ():	Читает символ с клавиатуры, но не отображает на экране	Не ожидает Enter. В стандарте С не определена.
putchar (ch);	Отображает символ на экране	
gets (str);	Читает строку с клавиатуры	
puts (str);	Отображает строку на экране	
printf (“[text]%fmt, …,a,b,…);	Осуществляет форматный вывод списка переменных на экран	
scanf (“%fmt, …”, &a”,&b, …);	Осуществляет форматный ввод списка переменных с клавиатуры	

Элементы ЯПВУ. ФАЙЛЫ

Pascal

В Pascal для организации обмена с внешними хранилищами данных используется файловые типы. Элементы могут быть любого типа, но не "файл" или "объект". Любой файл может содержать неограниченное количество элементов.

ТИПЫ файлов:

- типизированные файлы – файлы с объявленным типом элементов,

```
Var F3 : file of integer;
```

- текстовые файлы – файлы содержащие символьные строки переменной длины,

```
Var FT : text;
```

- нетипизированные (бестиповые) файлы – файлы содержащие последовательности элементов произвольного типа (но с оговоренным размером элементов).

```
Var F1 : file;
```

C / C++

Потоки и файлы

Поток – это логическое устройство ввода/вывода, являющееся *универсальным интерфейсом* между программой и физическими устройствами. Они называются **файлами**.

В языке С отсутствуют типы файлов, а, следовательно, отсутствует и записе-ориентированый обмен.

В С / С++ существует только потоко-ориентированный обмен данными.

Потоки бывают двух видов: **текстовые** и **бинарные (двоичные)**.

Текстовый поток – это последовательность символов. Стандарт С позволяет (но не требует!) организовывать потоки в виде строк с символом EOL в конце. При обмене с физическим устройством в текстовом потоке могут происходить определенные преобразования (например, EOL заменяться на CR, LF).

Бинарный поток – это последовательность байтов, однозначно соответствующая байтам на внешнем физическом устройстве. Никакого преобразования при обмене не происходит. В конце бинарного потока могут добавляться нулевые байты, например, для полного заполнения сектора на физическом диске.

Элементы ЯПВУ. ФАЙЛЫ

Pascal

Переменные файлового типа называются логическими файлами, а реальные устройства ввода/вывода и файлы на дисках – физическими файлами.

Доступ к файлам может быть последовательным (очередной элемент можно прочитать/записать только после выполнения аналогичной операции над предыдущим элементом) или прямым (можно выполнить чтение/запись произвольного элемента файла по заданному адресу).

Последовательный доступ возможен к файловым переменным *всех* типов: текстовым (text), типизированным (file of) и нетипизированным (file).

Прямой доступ возможен только для переменных файлового типа file и file of, для этого с ними связано понятие текущей позиции. Она указывает на конкретный элемент файла, с которым в данный момент можно выполнять поэлементные действия. В результате выполнения операции текущая позиция может перемещаться настраиваясь на тот или иной элемент файла. Все элементы считаются пронумерованными (начиная с 0). Имена физических файлов описываются строковыми переменными: 'e:\BP\history.pas' – имя и путь к дисковому файлу, и т.д.

Если имя файла задается в виде пустой строки, то файловая переменная связывается, в зависимость от направления обмена информацией, со стандартными файлами Input (для чтения данных с клавиатуры) или Output (для вывода данных на экран), эти файлы в Borland Pascal считаются открытыми по умолчанию.

C / C++

В языке C / C++ файлами могут быть файлы на дисках и реальные устройства ввода / вывода.

Последовательный доступ возможен для всех файлов, а прямой доступ – не для всех; для дисковых файлов – возможен, а для большинства принтеров – нет.

Для прямого доступа файл должен поддерживать запросы не местоположение текущей позиции. При открытии такого файла указатель текущей позиции в файле устанавливается в начало файла. При чтении/записи символа текущая позиция смещается (к концу файла).

Имена файлов представляются в виде литералов (строковых констант): "e:\\BC\\history.cpp", включающих имя файла и, при необходимости, путь к нему (в кавычках, обратная косая черта –удвоенная).

По умолчанию есть пять стандартных потоков, открывающихся при начале работы программы: stdin – ввод данных (клавиатура), stdout – вывод данных (экран), stderr – вывод сообщений об ошибках (экран), stdaux – дополнительный поток, stdprn – печать (принтер).

Элементы ЯПВУ. ФАЙЛЫ

Pascal

C / C++

Организация ввода/вывода в файл

1. Объявить файловую переменную,
2. Связать её с физическим файлом,
3. Открыть файл для чтения и/или записи,
4. Выполнить операции ввода/вывода,
5. Закрыть файл.

Обмен между логическими и физическими файлами происходит через **буфер** в системной области оперативной памяти, который выделяется для каждого открытого файла.

При записи в файл все данные поступают в буфер. Передача данных на внешнее хранилище (дисковый файл или устройство) происходит после заполнения буфера или после специальной команды.

При чтении из файла данные считываются в буфер, причем считано будет не столько, сколько запрашивается, а сколько поместиться в буфер.

Подпрограммы можно разделить на две подгруппы: **универсальные**, пригодные для любого типа файлов и **специализированные**, применимые только для определенных типов.

Специализированные подпрограммы будут в дальнейшем отмечены перечнем типов, к которым они применимы, заключенным в круглые скобки: (text, file, file of).

1. Определить указатель файла.
2. Открыть поток (связать его с файлом и задать режим обмена).
3. Выполнить операции ввода/вывода.
4. Закрыть файл.

Указатель файла – это указатель на структуру типа FILE, содержащую сведения о файле: имя, статус, указатель текущей позиции в начало файла и др. В этой структуре – блоке управления файлом не следует ничего изменять.

Объявление переменной-указателя файла:

FILE *fp; Указатель файла - fp используется в дальнейших операциях с потоком. Его передают функциям ввода \ вывода в качестве параметра, определяющего поток.

При открытии потока с ним связывается область памяти – буфер. При выводе данные накапливаются в буфере до его заполнения (а потом записываются в файл - выгружаются) или до закрытия потока. Чтение из файла производиться блоками, размер их равен буферу. При аварийном завершении программы данные в буфере могут пропасть.

Элементы ЯПВУ. ФАЙЛЫ

Pascal | **C**

Подпрограммы для работы с файлами

Процедура assign (f[, 'f_name']); – связывает логический файл f с физическим файлом f_name.
– f – имя файловой переменной (ифп),
– f_name – литеральное имя файла (иф)
Напр.: assign (F1, 'd:\BP\a.txt');
Если путь не задан, файл – в текущей папке. Связь файлов существует пока для переменной f не будет выполнена другая процедура assign (или close).

Процедура reset (f [, size]); – открывает логический файл f для чтения данных с первого элемента.
Здесь f – ифп, а size – размер записи в файле, используется только для нетипизированных файлов (file), по умолчанию 128 байт (это же размер буфера).
Напр.: reset (F1); Если файл не существует – выдается ошибка. Если уже открыт – открывается снова (текущая позиция в начале файла). Файл типа (text) открывается только на чтение.

Процедура append (f); – открывает текстовый файл для дополнения данных в конец файла.
Здесь f – ифп текстового типа (text).
Напр.: append (T1); Если файл не существует – выдается ошибка. Если уже открыт, то сначала закрывается и потом открывается. Текущая позиция устанавливается перед концом файла.

Процедура rewrite (f [, size]); – создает и открывает физический файл, имя которого присвоено логическому файлу f для записи данных с первого элемента.
Здесь f – ифп, а size – размер записи (используется только для (file), по умолчанию 128 байт).
Напр.: rewrite (F1); Если файл не существует он создается, если существует – он очищается и записывается с начала. Файл типа (text) открывается только на запись.

Функция fopen (fname, mode); – открывает поток и связывает его с файлом fname. Возвращает указатель файла или NULL.
Здесь fname – имя файла – указатель на строку символов (литерал), может включать путь к файлу, а mode – режим, который определяет назначение файла.
Напр.: FILE *fp; fp = fopen ("d:\\BC\\test.txt", "w");

Значения параметра режима (mode):
r (rt) – открыть текстовый файл для чтения,
w (wt) – создать текстовый файл для записи,
a (wa) – добавить записи в конец текстового файла,
rb – открыть бинарный файл для чтения,
wb – создать бинарный файл для записи,
ab – добавить записи в конец бинарного файла.
r+ – открыть текстовый файл для чтения и записи,
w+ – создать текстовый файл для чтения и записи,
a+ – добавить записи в конец текстового файла или создать текстовый файл для чтения и записи,
r+b (rb+) – открыть бинарный файл для чтения и записи,
w+b (wb+) – создать бинарный файл для чтения и записи,
a+b (ab+) – добавить записи в конец бинарного файла или создать бинарный файл для чтения и записи.

Файл не существует: r (rb, r+, rb+) – работа fopen() завершается отказом (NULL), w (…) – файл создается, a (…) – файл создается.
Файл существует: r (rb, r+, rb+) – файл открывается, w (…) – файл удаляется и открывается снова, a (…) – запись в конец файла.

Напр. – правильнее открывать файл с проверкой:
FILE *fp; if ((fp=fopen (" d:\\BC\\test.txt", "w"))==NULL)
{ printf("Ошибка при открытии файла\n"); exit(1); }

Элементы ЯПВУ. ФАЙЛЫ

Pascal

C

Подпрограммы для работы с файлами

Процедура close (f); – закрывает открытый логический файл f. Здесь f – ифп.
Напр.: close (F1); Обязательно надо использовать close для завершения работы с открытым для записи (*выходным*) файлом, т.к. при её выполнении происходит выгрузка буфера. Если не выполнить close содержимое буфера может пропасть. Для *входных* файлов close можно не выполнять.

Процедура erase (f); – стирает физический файл связанный с файловой переменной (логическим файлом) f. Здесь f – ифп.
Напр.: erase (F1); Файл к моменту вызова erase должен быть закрыт.

Процедура flush (f); – завершает обмен с файлом f без его закрытия (очищает буфер обмена). Здесь f – ифп.
Напр.: flush (F1); Для открытых файлов типа (text).

Функция fclose (fp); – явно закрывает поток ввода/вывода fp. Возвращает 0 при успешной операции закрытия или EOF в случае ошибки. Файл закрывается автоматически при завершении программы.
Здесь fp – указатель файла.
Напр.: fclose (fp); Функция записывает в файл все данные из буфера, а также освобождает блок управления файлом.

Функция remove (fname); – удаляет существующий файл. Возвращает 0 при успешной операции закрытия иначе - ненулевое.
Здесь fname – имя файла (литерал), может включать путь к файлу.
Напр.: remove (fp); Очищает буфер путем немедленной засылки записываемых данных на физическое устройство (файл).

Функция fflush (fp); – записывает данные из буфера, при этом файл остается открытым. Возвращает 0 при успешной операции закрытия или EOF в случае ошибки.
Здесь fp – указатель файла.
Напр.: fflush (fp); Очищает буфер путем немедленной засылки записываемых данных на физическое устройство (файл). При вызове с пустым (NULL) указателем файла – производит запись буферов во все открытые файлы.

Элементы ЯПВУ. ФАЙЛЫ

Pascal | **C**

Подпрограммы для работы с файлами

Процедура rename (f, nm); – переименовывает на диске физический файл связанный с логическим файлом f.
f – ифп, nm – новое имя файла (литерал).
Напр.: rename (F1, new_name); Файл к моменту переименования должен быть закрыт.

Процедура chdir (path); – изменение (смена) текущей папки.

Процедура getdir (drv, path); – для заданного диска drv помещает имя текущей папки в строковую переменную path.

Процедура mkdir (path); – создает новую папку с путём заданным переменной path.

Процедура rmdir (path); – удаляет пустую папку с путём заданным переменной path.

Процедуры и функции FindFirst, FindNext, GetFTime, SetFTime, GetFAttr, SetFAttr, FSplit, FSearch – изучить самостоятельно.

Функция filesize (f); – возвращает общее число элементов файла f.

Функция diskfree (drv); – возвращает число свободных байтов на диске drv.

Функция disksize (drv); – возвращает общее число байтов на диске drv.

Функция rename(oldfname, newfname); – переименовывает существующий файл или папку. Возвращает 0 при успешной операции закрытия, иначе – ненулевое значение.
Здесь oldfname – старое имя файла (литерал), может включать путь к файлу, а newfname – новое имя файла.
Напр.: rename ("test", "text.txt"); Новое имя файла не должно совпадать ни с одним существующим в данном каталоге.

Функция ferror (fp); – возвращает код ошибки при работе с потоком. Возвращает 0 при отсутствии ошибок, иначе – код ошибки (целое число).
Здесь fp – указатель файла.
Напр.: ferror (fp); чтобы определить природу ошибки надо исп. функцию perror().

Функция perror (s); – печатает строку вида "[s]: Сообщение об ошибке" на stderr.
Где s – текстовая строка до Сообщения об ошибке.
Напр.: perror ("Ошибка при работе с файлом");
Сообщение об ошибке – значение глобальной переменной errno, преобразованное в строку.

Функция clearerr (fp); – обнуляет флаг ошибки для потока fp.
Где fp – указатель файла.
Напр.: clearerr (fp);

Элементы ЯПВУ. ФАЙЛЫ

Pascal

C

Подпрограммы для работы с файлами

Процедура read ([f,] v1 [, v2, …, vN]);

для текстовых файлов (text)

– считывает N значений из текстового файла в переменные.

Здесь f – ифп, а v1, …, vN – список переменных, в которые считываются значения из f.

Если параметр f опущен, то используется стандартная файловая переменная Input.

Каждый параметр v является переменной символь-ного, строкового, целого и вещественного типа.

Напр.: read (x, y); После считывания строки не делается пропуск до следующей строки (надо использовать ReadLn).

для типизированных файлов (file of)

– считывает в переменную элемент файла f. Типы переменной и элемента файла – одинаковы.

Напр.: read (F1, x, y); При каждом считывании указатель позиции сдвигается к следующему элементу, и так до конца файла. Файл должен быть открыт.

Процедура readln ([f,] v1 [, v2, …, vN]);

– выполняет процедуру read и переходит к следующей строке. ReadLn (f); - перемещает текущую позицию к следующей строке. Тип f -(text).

Процедура blockread (f, buf, count[, result]); – считывает count элементов из файла f в переменную buf. (file, file of) Здесь f – ифп; buf – переменная любого типа, в которую происходит считывание; count – выражение, определяющее количество считываемых элементов; result – число фактически считанных элементов.

Напр.: blockread (F1, k, x, c);

Функция fgetc (fp); – читает символ из потока как значение unsigned char, преобразованный в целое (int). Здесь fp – указатель файла.

Напр.: fgetc (fp); Берется символ находящийся за текущей позицией, а затем указатель текущей позиции увеличивается на 1. Если символ не прочитался, возвращается EOF. Т.к. EOF – действительное целое, то при работе с бинарными файлами надо исп. функцию feof().

Функция fgets (s, n, fp); – читает из входного потока fp не более n-1 символов в строку s. Возвращает s при отсутствии ошибок, иначе – NULL. Здесь s – указатель строки, n – количество читаемых символов (начиная с 0), fp – указатель файла.

Напр.: fgets (str, num, fp); Чтение завершается при достижении num-1, конца строки (EOL) или конца файла (EOF). Прочитанная строка автоматически завершается '\0'. EOL не отбрасывается, а помещается в конец строки str.

Функция fscanf (fp, fmt…, list….); – читает из потока форматированные данные, как scanf. Возвращает число записанных переменных или EOF. Где fp – указатель файла, fmt - форматы вводимых данных, list – список переменных.

Напр.: fscanf (fp,"%f",&g);

Функция fread (buf, size, count, fp); – читает из потока fp count объектов размером size и помещает их в массив buf. Указатель сдвигается. Возвращает число прочитанных элементов.

Напр.: fread (b, sizeof(int), n. fp);

Элементы ЯПВУ. ФАЙЛЫ

Pascal

C

Подпрограммы для работы с файлами

Процедура write ([f,] v1 [, v2, …, vN]);

для текстовых файлов (text)

– записывает N значений параметров v в файл f. Здесь f – ифп, а v1, …, vN – список параметров, из которые записываются значения в f.

Если параметр f опущен, то используется стандартная файловая переменная Output.

Каждый параметр v является выражением символьного, целого, вещественного, строкового, упакованного строкового или булевого типа, значение которого записывается в файл f. Параметр v имеет вид: expr [: size [:dec]] где expr – вводимое в файл выражение, size – минимальная ширина поля dec – число десятичных знаков в вещественном числе с плавающей точкой.

Напр.: write (F1, $y+z^3$, x);

для типизированных файлов (file of)

– записывает N значений переменных v в файл f. Типы переменной и элемента файла – одинаковы.

Напр.: write (F1, x, y); При каждой записи текущая позиция сдвигается к следующему элементу, когда достигнут EOF – файл расширяется.

Процедура writeln ([f,] v1 [, v2, …, vN]);

– выполняет процедуру write и записывает в файл EOL. Writeln (f); - записывает в файл EOL. (text).

Процедура blockwrite (f, buf, count[, result]); – записывает count элементов в файл f из переменной buf. (file, file of) Здесь f – ифп; buf – переменная любого типа, из которой происходит запись; count – выражение, определяющее количество записываемых элементов; result–число фактически записанных элементов.

Напр.: blockwrite (F1, k, x, c);

Функция fputc (ch, fp); – записывает символ ch в поток fp. Возвращает значение записанного символа или EOF. Здесь ch – символ, fp – указатель файла.

Напр.: fputc ('a', fp);

Функция fputs (s, fp); – записывает в заданный поток fp строку s. Возвращает неотрицательное при отсутствии ошибок, иначе – EOF.

Здесь s – указатель строки, fp – указатель файла.

Напр.: fputs (str, fp); Символ '\0' не записывается.

Функция fprintf (fp, fmt…, list….); – записывает в поток форматированные данные, как printf. Возвращает число записанных переменных или EOF.

Здесь fp – указатель файла, fmt - форматы выводимых данных, list – список переменных.

Напр.: fprintf (fp,"%f", g);

Функция fwrite (buf, size, count, fp); – записывает в поток fp count объектов размером size из массива символов адресуемого указателем buf. Указатель сдвигается. Возвращает число записанных элементов.

Напр.: fwrite (b, sizeof(int), n, fp);

Элементы ЯПВУ. ФАЙЛЫ

Pascal | **C**

Подпрограммы для работы с файлами

Функция eof [(f)]; – возвращает True, если при чтении текущая позиция находится за последним элементом файла f или файл пуст, иначе False.
Здесь f – ифп.
Напр.: eof (F1); Если имя файла f опущено, используется файл Input.
Функция filepos (f); – возвращает текущую позицию в файле f.
Здесь f – ифп. (file, file of)
Напр.: filepos (F1); Если текущей позицией является начало файла, функция возвращает значение 0, а если – конец файла, то размер файла.
Функция eoln [(f)]; – возвращает для файла f True, если при чтении текущая позиция находится за последним элементом стоки или строка пуста, иначе False.
Здесь f – ифп. . (text)
Текущая позиция – на EOL – True, иначе False.
Напр.: eoln (F1); Если имя файла f опущено, используется файл Input.
Процедура truncate (f); – усекает размер файла f до текущей позиции.
Здесь f – ифп. (file, file of)
Напр.: trancate (F1); Все элементы после текущей позиции в файле F1 удаляются и текущая позиция становится концом файла.

Функция feof (fp); – проверяет достигнут ли конец файла (EOF).
Возвращает ненулевое значение когда достигнут EOF или 0, если не достигнут.
Здесь fp – указатель файла.
Напр.: feof (fp); Полезен при работе с при работе с бинарными файлами, т.к. маркер конца файла – полноценное двоичное целое.
Функция fgetpos (fp, pos); – сохраняет в объекте pos (типа fpos_t) текущую позицию в файле, связанном с потоком fp. Возвращает 0 при отсутствии ошибок, иначе – ненулевое значение.
Здесь fp – указатель файла, pos – текущая позиция.
Напр.: fgetpos (fp, fpos);
Функция fsetpos (fp, pos); – перемещает текущую позицию в файле, связанном с потоком fp, на позицию, заданную указателем pos. Возвращает 0 при успешном завершении и ненулевое значение при ошибке.
Здесь fp – указатель файла, pos – указатель позиции.
Напр.: fscanf (fp, &pos); Значение текущей позиции pos предварительно запрашивается функцией fgetpos().
Функция rewind (fp); – перемещает указатель текущей позиции в начало заданного потока fp и очищает флаги ошибок.
Напр.: rewind (fp);

Элементы ЯПВУ. ФАЙЛЫ

Pascal | **C**

Подпрограммы для работы с файлами

Процедура seek (f, n); – перемещает текущую позицию в файле f к элементу n.
Здесь f – ифп, n – порядковый номер элемента, целое число. (file, file of)
Напр.: seek (F1, 10); Номер первого элемента файла – 0. Чтобы расширить файл, можно переместить текущую позицию в конец файла: seek (F1, filesize(f));,- а затем добавить элементы.

Функция seekeof [(f)]; – возвращает для файла f True, если при чтении текущая позиция находится за последним элементом файла f или файл пуст, иначе False.
Здесь f – ифп. (text)
Аналогична функции eof(f); – но пропускает все пробелы, знаки табуляции и EOL, как лидирующие, так и от последнего значащего символа до конца файла.
Напр.: seekeof (F1); Можно исп. при считывании числовых значений из текстового файла.

Функция seekeoln [(f)]; – возвращает для файла f True, если при чтении текущая позиция находится за последним элементом стоки или строка пуста, иначе False.
Здесь f – ифп. . (text)
Аналогична функции eoln(f); - но пропускает все пробелы и знаки табуляции, как лидирующие, так и от последнего значащего символа до конца строки.
Напр.: seekeoln (F1); Можно исп. при считывании числовых значений из текстового файла. Если имя файла f опущено, используется файл Input.

Функция fseek (fp, offset, origin); – устанавливает текущую позицию в файле связанном с потоком fp, на позицию offset, отсчитываемую от origin.
Возвращает 0 при успешной операции и ненулевое значение при ошибке.
Здесь fp – указатель файла; origin – начало отсчета, определяется макросами SEEK_SET (начало файла), SEEK_CUR (текущая позиция), SEEK_END (конец файла); offset – смещение текущей позиции от начала отчета (в байтах).
Напр.: fseek (fp, sizeof(gh), SEEK_CUR); Эта функция исп. преимущественно при работе с бинарными файлами. Для текстовых файлов origin должен быть всегда SEEK_SET, а offset – брать значение из функции ftell() для потока fp или нуль.

Функция ftell (fp); – возвращает текущую позицию в файле, связанном с потоком fp. При возникновении ошибки функция возвращает -1.
Здесь fp – указатель файла.
Напр.: ftell (fp); Для бинарных потоков значение ftell() равно количеству байтов от начала файла до текущей позиции. Для текстовых файлов возвращаемое значение используется исключительно как аргумент функции fseek(). (Суть дела в возможных преобразованиях в тестовых файлах.)

Элементы ЯПВУ. ФАЙЛЫ

Pascal

C

Подпрограммы для работы с файлами

Процедура settextbuf (f, buf[, size]); – переназначает буфер для обмена с текстовым файлом. Здесь f – ифп., buf – переменная для размещения буфера (любого типа – нетипизированная), size – выражение – размер буфера в байтах. Settextbuf действует до нового assign(f). (text)
Напр.: SetTextBuf(F1, bf, 512); По умолчанию size равна sizeoff(buf). Рекомендуется задавать settextbuf до открытия файла (или сразу после открытия), чтобы не потерять данные буфера. Оптимальный размер буфера равен размеру сектора диска.
Функция ioresult; – возвращает – код ошибки последней операции ввода/вывода.
При нормальном завершении код – 0.
Напр.: IOResult; Работает только при включенном режиме проверки ошибок ввода/вывода, ключ компиляции {$I-} (по умолчанию ключ {$I+} – выключен).

Стандартные текстовые файлы

input – стандартный файл ввода (по умолчанию – клавиатура).
output – стандартный файл вывода (по умолчанию – экран дисплея).

Разрешено переназначение стандартных файлов ввода/вывода:

```
assign (output, 'd:\BP\outfile.dat');
```

Функция setbuf (fp, buf); – устанавливает буфер ввода/вывода для файла связанного с потоком fp, или отключает буферизацию, если buf = 0.
Здесь fp – указатель файла; buf – указатель на буфер.
Напр.: setbuf (fp, buffer); Размер буфера, задаваемого программистом, равен константе BUFSIZ.
Функция setvbuf (fp, buf, mode, size); – устанавливает буфер обмена, заданный указателем buf, для потока fp. При возникновении ошибки функция возвращает -1.
Здесь fp – указатель файла, buf – указатель на массив символов, size – размер буфера в байтах, mode – режим буферизации (определяется константами как: _IOFBF, _IOLBF, _IONBF.
Напр.: setvbuf(fp, buffer, _IOFDF, 128);
Функция tmpfile (); – открывает временный двоичный файл для ввода/вывода и возвращает указатель на связанный с файлом поток. При возникновении ошибки функция возвращает 0.
Напр.: FILE *tmpfile (void); Автоматически исп. уникальное имя файла. Временный файл автоматически удаляется при закрытии файла или завершении программы. Количество временных файлов определяется константой TMP_MAX (предел устанавливается константой FOPEN_MAX).
Функция tmpnam (name); – создает уникальное имя для временного файла. При возникновении ошибки функция возвращает 0, иначе указатель на массив имен файлов..
Здесь fp – указатель файла,
Напр.: tmpnam (fnam);

Элементы ЯПВУ. ФАЙЛЫ

C

C

Подпрограммы для работы с файлами

Функция freopen (fname, mode, fp); - связывает существующий поток с другим файлом - fname. Возвращает указатель файла или NULL (при ошибке). Если поток fp был открыт, то он закрывается и создается новый поток.
Здесь fname – новое имя файла, mode – режим, который определяет назначение файла, fp– указатель закрываемого файла.
Напр.: freopen ('d', w+, p); Эту функцию можно использовать, в частности, для перенаправления потоков.

Функция ungetc (ch, fp); – возвращает символ ch в поток fp.
Здесь fp – указатель файла; ch – возвращаемый символ.
При возникновении ошибки функция возвращает значение EOF, при отсутствии ошибок - ch.
Напр.: ungetc ('d', fp); Вызов функций fflush(), fseek(), rewind() аннулирует действие ungetc() и сбрасывает символ (ch). Попытка вернуть в поток ввода EOF игнорируется. Обращение к функции ungetc() очищает признак конца файла, связанного с данным потоком. Значение указателя текущей позиции файла для текстового потока до тех пор, пока не прочитаны все возвращаемые символы, остается таким, как до первого вызова ungetc(). При работе с бинарными файлами каждый вызов функции уменьшает указатель текущей позиции файла.

Функции vprintf, vfprintf – их действие аналогично функциям printf и fprintf, но список аргументов заменяется указателем на список аргументов.

Функции vscanf, vfscanf – их действие аналогично функциям scanf и fscanf, но список аргументов заменяется указателем на список аргументов.

Стандартные потоки

Открываются автоматически в начале выполнения программы на C / C++ стандартные потоки можно перенаправить:

```
freopen ("output.txt', "w+", stdout);
```

Элементы ЯПВУ. ФАЙЛЫ

C

C

Подпрограммы для работы с файлами

Функции ввода/вывода языка С собраны в библиотеке (заголовке) <stdio.h>

Некоторые типы данных, макросы и константы заголовка <stdio.h>

Тип данных FILE – содержат информацию для работы с файлом.

Структура этого типа зависит от конкретней реализации, приведем один из вариантов:

```
typedef struct {
  int level;                  // флаг состояния буфера
  unsigned flag;              // флаг состояния файла
  char fd;                    // префикс файла
  unsigned char hold;               // непереданный символ
  int  bsize;                 // размер внутреннего буфера
  unsigned char_FAR *buffer;/* значение указателя для доступа внутрь буфера, задает (в зависимости от режима буферизации) начало буфера, начало строки или текущее значение указателя внутри буфера */
  unsigned char_FAR *curp;  /* текущее значение указателя для доступа внутрь буфера, задает текущую позицию в буфере для обмена с программой */
  unsigned  istemp;                 // флаг временного файла
  short token;                // маркер действительности файла
 }  FILE;
```

Тип данных fpos_t – используется функциями fgetpos() и fsetpos() для хранения текущей позиции в файле (целое без знака).

Тип данных size_t – используется для представления результат операции sizeof (целое без знака. не длиннее unsigned long).

Константа BUFSIZ – используется для задания в байтах размера буфера потока данных функциями fopen(), freopen(), setbuf(). При открытии потока данных к нему автоматически присоединяется буфер длиной BUFSIZ. Минимальный размер BUFSIZ – 255 байт.

Константа EOF – используется для сообщения об окончании файла (отрицательное целое число).

Элементы ЯПВУ. ФАЙЛЫ

C

C

Подпрограммы для работы с файлами

Функции ввода/вывода языка С собраны в библиотеке (заголовке) <stdio.h>

Некоторые типы данных, макросы и константы заголовка <stdio.h>

Константа FOPEN_MAX – задает максимальное число открытых потоков, оно должно быть <=8, при этом stdin, stdout и stderr – открывающиеся автоматически входят в это число. Программы, открывающие более 5 потоков одновременно, должны проверить доступно ли это количество потоков.

Константы _IOFBF, _IOLBF, _IONBF – применяются для определения режима буферизации в функции setvbuf(). Где _IOFBF – 0 – полная буферизация, _IOLBF – 1 – построчная буферизация текстового файла, _IONBF – 2 – отмена буферизации.

Константа L_tmpnam – задает минимальную длину строки имени файла, генерируемого функцией tmpnam().

Константы SEEK_SET, SEEK_CUR, SEEK_END – это аргументы функции fseek(). Здесь SEEK_SET– 0 – сдвиг выполняется от начала файла, SEEK_CUR – 1 – сдвиг выполняется от текущей позиции указателя записи/чтения файла, SEEK_END – 2 – сдвиг выполняется от конца файла.

Константа TMP_MAX – задает максимальное количество различных временных имен файла, генерируемых функцией tmpnam() в течение одного сеанса работы программы. Значение этой константы не может быть меньше 25.

ФАЙЛЫ

Pascal | Практические занятия | **C**

Задание 1. Создать на диске Z:\ файл numbers.txt, записать в него 5 введенных с клавиатуры целых чисел.

```
Program file_5num;
(*Создает на диске Z:\ файл numbers.txt и записывает в него
5 целых чисел введенных пользователем с клавиатуры*)
var
 f : text;          (* текстовый файл *)
 n : integer;       (* вводимое число *)
 i : integer;       (* счетчик чисел *)
begin
  writeln ('Создание и заполнение файла numbers.txt');
  assign (f, 'z:\numbers.txt');   (* Связь с файлом *)
  rewrite (f); (*Созд. и откр. файл в режиме перезаписи*)
   writeln ('Введите 5 целых чисел, нажимая Enter после
каждого числа');
    for i:=1   to  5  do
     begin
       write ('->');
       readln (n); (* чтение числа из буфера клавиатуры*)
       writeln (f, n); (* запись считанного числа в файл *)
     end;
     close (f);      (* закрыть файл *)
       writeln  ('Введенные числа записаны в файл ',
'z:\numbers.txt');
   readln;
end.
```

```
/* Создает на диске Z:\ файл numbers.txt и записывает в него 5
целых чисел введенных  пользователем с клавиатуры */
#include <stdio.h>
#include <conio.h>
#define      FNAME   "e:\\edu2\\numbers.txt\0"   //   имя
создаваемого файл
#define N 5          // количество вводимых в файл чисел
void main()
{    char fname[20] = FNAME;
      FILE *out;   // файл чисел
       int n;          // вводимое число
       int i;          // счетчик цикла for
 puts ("\nСоздание файла");
  // Открыть файлы режиме записи текста - wt
 if (( out = fopen(fname, "wt")) == NULL)
   {   printf ("Ошибка при открытии файла для записи");
       getch();
        return;      }
 printf ("Вводимые числа записываются в файл");
 printf ("%s\n", fname);
 puts ("Введите 5 целых чисел, нажимая Enter после
каждого числа");
   for  (i = 0; i < N; i++)
    { printf ("->");
      scanf ("%i", &n);       // читать числа с клавиатуры
      fprintf (out, "%i\n", n);  // записать числа в файл   }
  fclose (out);     // закрыть файл
  printf ("Введенные числа записаны в файл %s\n", fname);
  puts ("\nДля завершения нажмите Enter");
  getch();  }
```

ФАЙЛЫ

Pascal | Практические занятия | C

Задание 2. Вывести на экран содержимое файла numbers.txt.

```
Program file5displ;
(* Выводит на экран файл z:\numbers.txt *)
var
 f : text;        (* текстовый файл *)
 n : integer;  (* вводимое число *)
begin
  writeln ('Содержимое файла numbers.txt');
  writeln ('----------------------------------------');
  assign (f, 'z:\numbers.txt');  (*Связь с файлом*)
  reset (f);                    (* Открыть файл для чтения *)
    while not EOF(f) do  (*Выполнять до конца файла*)
     begin
       readln (f, n);  (* Читать число из файла *)
      writeln (n); (*Вывести  прочитанное числа на экран *)
     end;
   close (f);     (* закрыть файл *)
   writeln ('----------------------------------------');
   readln;
end.
```

```
// Выводит на экран файл z:\numbers.txt
#include <stdio.h>
#include <conio.h>
// имя создаваемого файл
#define  FNAME "e:\\edu2\\numbers.txt\0"
 void main()
{  char fname[20] = FNAME;
    FILE *in;        // файл чисел
    char st[80];     // строка из файла
  printf ("\nСодержимое файла %s\n", fname);
  puts ("-----------------------------------");
// Открыть файл в режиме чтения текста - rt
  if (( in = fopen(fname, "rt")) == NULL)
    { printf ("Ошибка при открытии файла для чтения");
       getch();
       return;    }
 while (!feof(in))
   {
    fscanf (in, "%s", &st);   // читать число из файла
    printf ("%s\n", st);      // вывести число на экран
   }
   fclose (in);  // закрыть файл
 puts ("-----------------------------------");
  puts ("\nДля завершения нажмите Enter");
  getch();
}
```