DCL_Встроенные библиотеки

Встроенные библиотеки

- Общесистемная библиотека - dcl_std_lib
  - Константы
  - Переменные
  - version – идентификатор версии
  - sizeof – размер объекта данных
  - buffof – размер буфера объекта данных
  - setsize – установить размер объекта данных
  - constant – установить переменную как неизменяемую
  - get_errtext – выдача расширенного сообщения об ошибке
  - stod – преобразование строки в данные
  - dtos – преобразование данных в символьную строку
  - atos – преобразование данных в символьную строку
  - dtod – преобразование данных разных форматов
  - time – выдача текущей даты/времени
  - elapsed_time – выдача временных меток
  - trim – усечение начальных и конечных символов
  - replace – замена одного символьного фрагмента на другой
  - replace_char – замена символов из заданного набора
  - transpose – структурное преобразование текста
  - upper – перевод символов в верхний регистр
  - cp_convert – преобразование кодовой страницы текста
  - system – исполнение shell-команды
  - sleep – отработка заданной паузы
  - log_file – запись сообщения в лог-файл
  - log_event – запись сообщения в лог системных событий
  - signal – отправка/подъем сигнала
- Библиотека отладочных средств - dcl_debug_lib
  - message – вывод сообщения на экран
  - variable, variable_ – вывод состояния переменной на экран
  - show, show_ – вывод значения переменной на экран
  - timestamp – выдача на экран данных по дате/времени
- Библиотека работы с файлами - dcl_file_lib
  - Константы
  - f_with – смена активного рабочего файла
  - f_open – открытие рабочего файла
  - f_close – закрытие рабочего файла
  - f_seek – позиционирование рабочего файла
  - f_tell – запрос текущей позиции в рабочем файле
  - f_read – считывание данных из рабочего файла
  - f_readl – считывание строки данных из рабочего файла
  - f_readonce – одномоментное открытие, считывание и закрытие файла
  - f_write – запись данных в рабочий файл
  - f_writeonce – одно-моментное открытие, запись и закрытие файла
  - f_unlink – удаление файла
  - f_exists – проверка существования файла или раздела
  - f_cp_convert – преобразование кодовой страницы текста
  - f_check – поиск ключевых слов в файле
- Библиотека работы с SQL-запросами - dcl_sql_lib
  - Константы
  - sql_codepages – задание преобразования кодовых страниц
  - sql_connect – установление соединение с базой данных
  - sql_disconnect – закрытие соединения с базой данных
  - sql_commit – операция COMMIT
  - sql_rollback – операция ROLLBACK
  - sql_execute – выполнение модифицирующего SQL-оператора или SQL-блока
  - sql_writeLOB – занесение LOB_значения
  - sql_fetch2csv – выборка данных в CSV-файл
  - sql_fetch2dbf – выборка данных в DBF-файл
  - sql_open – открытие курсора SELECT-оператора
  - sql_close – закрытие курсора SELECT-оператора
  - sql_error – запрос текста ошибки
- Библиотека работы с MS Office - dcl_office_lib
  - Константы
  - excel_template – загрузка данных в Excel-шаблон

Общесистемная библиотека - dcl_std_lib

Константы

SE_BADFORMAT

SE_STRSIZE

SE_BADSTRING

SE_DATABUFF

Переменные

errno - код ошибки

version – идентификатор версии

Параметров нет.

Возвращает строку с кодом версии.

sizeof – размер объекта данных

Параметры:

- объект данных произвольного типа или ссылка на него

Функция выдает размер (в байтах) заданного объекта данных.

Пример

```
sizeof(a);
sizeof("TEST");
```

buffof – размер буфера объекта данных

Параметры:

- объект данных произвольного типа или ссылка на него

Функция выдает размер (в байтах) буфера заданного объекта данных.

Пример

```
buffof(a);
```

setsize – установить размер объекта данных

Параметры:

- объект данных - символьная строка или ссылка на нее
- размер (в байтах)

Функция устанавливает размер заданного объекта данных. Ничего не возвращает.

Пример

```
setsize(a, 8);
```

constant – установить переменную как неизменяемую

Параметры:

- переменная, устанавливаемая как неизменяемая

Функция устанавливает у переменой признак неизменяемости, после чего ее значение не может быть изменено.
При попытке изменения такой переменной выдается ошибка 45 "Попытка изменения READONLY-переменной"

Пример

```
a<=="Test" ;
constant(a) ;

a<=="AAAA" ;  //
```

get_errtext – выдача расширенного сообщения об ошибке

Параметров нет.

Возвращает строку с текстом расширенного описания ошибки.

stod – преобразование строки в данные

Параметры:

- формат преобразования – символьная строка
- преобразуемая символьная строка
- принимающая переменная

Преобразует строку или ее начальный фрагмент в данные в соответствии с указанным форматом.

Структура формата: [pref_chr]%[u][pref]type[.point][>]

type - базовый тип: d - int/short/long, f - float/double

pref - префикс типа: h – short, l - long/double

u - спецификатор unsigned

point - положение запятой, если она опущена

pref_chr - префикс-символ,

так для строк "............444444" и "-............444444" префикс-символ будет '.'

> - признак авто-детектирования окончания дешифровки - при отключенном авто-детектировании может возникнуть ошибка SE_BADSTRING

Возвращает указатель на первый не дешифрованный символ.

При возникновении ошибок они прописываются в переменную errno:

SE_BADFORMAT - некорректный формат

SE_STRSIZE - слишком длинная дешифруемая строка

SE_BADSTRING - строка не полностью дешифрована

SE_DATABUFF - слишком короткий буфер данных

Пример

```
stod("%d",  buff[13...15], $BankAcc1) ;

                stod(".%lf.2", buff[36...51], $BankSpr) ;
```

dtos – преобразование данных в символьную строку

Параметры:

- формат преобразования – символьная строка
- принимающая символьная строка
- преобразуемые данные

Преобразует данные в символьную строку в соответствии с указанным форматом.

Структура формата: [+][pref_chr]%[u][pref]type<area>[.point]

<area> - длина «поля вывода»

type - базовый тип: d - int/short/long, f - float/double

pref - префикс типа: h – short, l - long/double

u - спецификатор unsigned

point - положение запятой, если она опущена

pref_chr - префикс-символ,

так для строк "............444444" и "-............444444" префикс-символ будет '.'

\+ - признак вынесения знака перед префикс-символом

Возвращает указатель на внутренний буфер результата.

При возникновении ошибок они прописываются в переменную errno:

SE_BADFORMAT - некорректный формат

SE_STRSIZE - слишком короткий буфер для строки

SE_BADSTRING - строка не умещается в поле вывода

SE_DATABUFF - размер буфера данных не соответствует типу

Пример

```
dtos("0%ld6", temp, $BankAcc2) ;
                    dtos("+.%lf16.-2", temp, $BankSpr) ;
```

atos – преобразование данных в символьную строку

Параметры:

- преобразуемые данные

Преобразует данные в символьную строку в соответствии с типом данных.

Возвращает указатель на внутренний буфер результата.

dtod – преобразование данных разных форматов

Параметры:

- формат преобразования – символьная строка
- преобразуемые данные
- результата преобразования

Преобразует данные разных типов в соответствии с указанным форматом.

Структура формата: in%out, где in и out – описания типов входных и выходных данных в следующем формате: [u][pref]type

type - базовый тип: d - int/short/long, f - float/double

pref - префикс типа: h – short, l - long/double

u - спецификатор unsigned

Ничего не возвращает.

При возникновении ошибок они прописываются в переменную errno:

SE_BADFORMAT - некорректный формат

SE_DATABUFF - размер буфера данных не соответствует типу

time – выдача текущей даты/времени

Параметры:

- буфер данных

Выдает в буфер текущее время и дату в формате HH24:MI:SS DD.MM.YYYY.

Ничего не возвращает.

elapsed_time – выдача временных меток

Параметры:

- признак сброса счетчика времени (1-отсчитывать от данного момента, 0-продолжать отсчет)

Функция возвращает время в секундах, прошедших от последнего опорного момента времени.
Опорный момент времени устанавливается при первом вызове функции и может быть изменен при передаче признака сброса счетчика времени, установленного в 1.

Пример

```
elapsed_time(1) ;
   sleep(10) ;
seconds=elapsed_time(0) ;
```

trim – усечение начальных и конечных символов

Параметры:

- буфер данных
- список усекаемых символов (необязательный)

Если список усекаемых символов не указан, то в качестве такового используется пробел.

Возвращает указатель на результат.

Пример

```
trim(buff, " \t\r\n") ;

trim(buff) ;
```

replace – замена одного символьного фрагмента на другой

Параметры:

- буфер данных
- заменяемый фрагмент
- заменяющий фрагмент

Ничего не возвращает.

Пример

```
replace(buff, "\t", " ") ;
```

replace_char – замена символов из заданного набора

Параметры:

- буфер данных
- набор заменяемых символов
- заменяющий фрагмент

Ничего не возвращает.

Пример

```
replace_char(buff, ".: ", "_") ;
```

transpose – структурное преобразование текста

Параметры:

- исходный текст
- преобразованный текст
- шаблон исходного текста
- шаблон преобразованного текста
- настройки преобразования

Возвращает: 1 - преобразование проведено

0 - текст не соответствует шаблону

-1 - ошибка обработки.

Шаблон исходного текста представляет собой последовательность текстовых маркеров с размещенными между ними метками целевых полей. При разборе исходного текста текстовые маркеры используются для контроля формата текста и в преобразованный текст не попадают. Метки целевых полей имеют следующую структуру: {<Name>[:<Size>]}, где <Name> - имя поля, <Size> - длина выбираемых данных . Если спецификатор <Size> не указан, то выборка данных для поля производится по ограничивающим его текстовым маркерам. Вместо символов {} могут использоваться любые другие символы, задаваемые в параметре Настройки преобразования.

Шаблон преобразованного текста представляет собой текстовых фрагментов с размещенными между ними метками вывода целевых полей. Метки целевых полей имеют следующую структуру: {<Name>}, где <Name> - имя поля. Вместо символов {} могут использоваться любые другие символы, задаваемые в параметре Настройки преобразования, но они должны совпадать с таковыми для Шаблона исходного текста. Каждому из целевых полей, используемых в Шаблоне должно соответствовать поле в Шаблона исходного текста.

Настройки преобразования содержат набор обязательный и необязательных спецификаторов:

FIELD: - указывает два символа – начала и конца описателя целевого поля (обязательный спецификатор)

ERROR_RAISE - задает эскалацию ошибок преобразования на уровень интерпретатора, то есть в случае их возникновения дальнейшее исполнение скрипта прекращается (необязательный спецификатор)

В качестве иллюстрации рассмотрим следующий пример.

Задаем шаблон исходного текста:

КОМИССИЯ: ДАТА [dd:2].[mm:2].[yy:2], КАРТА [card:16], ЧЕК [order]

и шаблон преобразованного текста:

CPS Fee C:[card] O:[order]

Тогда текст («вырезаемые» значения полей выделены):

КОМИССИЯ: ДАТА 24.08.12, КАРТА 1234567890123456, ЧЕК AA-33-768

будет преобразован в текст (вставляемые значения полей выделены):

CPS Fee C:1234567890123456 O:AA-33-768

Для следующих текстов преобразование не будет производиться и будет возвращено значение 0 (красным выделены «сработавшие» маркеры):

ОПЛАТА: ДАТА 24.08.12, КАРТА 1234567890123456, ЧЕК AA-33-768

КОМИССИЯ: ДАТА 24-08-12, КАРТА 1234567890123456, ЧЕК AA-33-768

Пример

```
transpose(buff_in, buff_out, "20{Y:2}-{M:2}-{D:2} ", "{D:2}.{M:2}.{Y:2}", …
… "FIELD:{}") ;
```

upper – перевод символов в верхний регистр

Параметры:

- буфер данных
- кодировка

Поддерживаются следующие кодировки:

- BASIC - преобразуются только символы латинского алфавита
- WINDOWS - перекодировка по CP-1251
- CP1251 - перекодировка по CP-1251

Если указанная кодировка не поддерживается, то никаких действий не производится.

Ничего не возвращает.

Пример

```
upper(buff, "Windows") ;
```

cp_convert – преобразование кодовой страницы текста

Параметры:

- буфер данных
- исходная кодировка
- результирующая кодировка

Поддерживаются следующие перекодировки:

- CP866 CP1251
- UTF-8 CP1251 (только для Windows)
- UTF-8 CP866 (только для Windows)

Если указанная перекодировка не поддерживается, то выдается ошибка.

Ничего не возвращает.

Пример

```
cp_convert(buff, "CP1251", "CP866") ;
```

system – исполнение shell-команды

Параметры:

- текст исполняемой команды

Исполняет заданную shell-команду. Действие аналогично вызову:

```
system(command)
```

Пример

```
system("rm test.dat") ;
```

sleep – отработка заданной паузы

Параметры:

- время ожидания в секундах

Приостанавливает исполнение на заданное время.

Ничего не возвращает.

Пример

```
sleep(300) ;
```

log_file – запись сообщения в лог-файл

Параметры:

- путь к файлу лога
- записываемая строка данных

Открывает файл лога, записывает строку данных и закрывает файл лога. Перед строкой данных вставляется метка времени.

Возвращает 0, если файл открыт нормально и -1 при наличии ошибок. Код ошибки передается через переменную errno б иблиотеки dcl_std_lib: EACCES, EEXIST, EMFILE, ENOENT (расшифровка – согласно C ANSI).

Пример

```
log_file("C:\\global.log","Fatal error");
```

log_event – запись сообщения в лог системных событий

Параметры:

- имя журнала
- имя источника
- категория события: INFO, ERROR, WARNING
- записываемая строка данных

Реализовано только для MS Windows.

Записывает событие в системный лог. В случае отсутствия журнала или источника в системе системных логов - пытается их создать.

Возвращает 0, если событие записано и -1 при наличии ошибок. Код ошибки передается через переменную errno библио теки dcl_std_lib (расшифровка – согласно C ANSI или таблице ошибок функции GetLastError).

Пример

```
status=log_event("Application", "WSH", "INFO", "RISKS report: File inactive.csv transfer SUCCESS") ;
```

signal – отправка/подъем сигнала

Параметры:

- тип сигнального интерфейса: ABTP
- адрес получателя сигнала
- имя/идентификатор сигнала
- значение сигнала
- дополнительная информация по сигналу

Отправляет (поднимает) сигнал по заданному интерфейсу.

Возвращает 0, если сигнал отправлен нормально и -1 при наличии ошибок..

Пример

```
signal("ABTP", "abs4:22008", "SUCCESS”, "");
```

Библиотека отладочных средств - dcl_debug_lib

message – вывод сообщения на экран

Параметры:

- символьная строка

Функция выдает строку на экран и добавляет перевод каретки. Ничего не возвращает.

Пример

```
message("TEST");
```

variable, variable_ – вывод состояния переменной на экран

Параметры:

- переменная произвольного типа

Выводит на экран параметры переменной в формате:

для числовой:

NAME TYPE ADDR (SIZE_B/BUFF_B) (SIZE/BUFF) VALUE1 VALUE2 ...

для бинарной (строчной):

NAME TYPE ADDR (SIZE_B/BUFF_B) "STRING"

где NAME - имя переменной

TYPE - тип переменной

ADDR - адрес буфера

SIZE - размер объекта в единицах объекта

BUFF - размер буфера объекта в единицах объекта

SIZE_B - размер объекта в байтах

BUFF_B - размер буфера объекта в байтах

VALUE - числовое значение (значения - для массива)

STRING - содержимое буфера

Вариант variable_ после вывода на экран ожидает нажатия клавиши. Ничего не возвращает.

show, show_ – вывод значения переменной на экран

Параметры:

- переменная произвольного типа

Выводит на экран значение переменной в формате.

Вариант show_ после вывода на экран ожидает нажатия клавиши. Ничего не возвращает.

timestamp – выдача на экран данных по дате/времени

Параметров нет.

Выдает на экран текущее время и дату, а также время прошедшее после предыдущего вызова.

Ничего не возвращает.

Библиотека работы с файлами - dcl_file_lib

Использует объекты системной библиотеки dcl_std_lib.

Константы

O_CREAT

O_EXCL

O_TRUNC

O_APPEND

SEEK_SET

SEEK_CUR

SEEK_END

EACCES

EEXIST

EMFILE

ENOENT

ENOMEM

EINVAL

ENOSPC

EBADF

f_with – смена активного рабочего файла

Параметры:

- дескриптор файла

Устанавливает новое значение для дескриптора активного рабочего файла. Дескриптор рабочего файла должен быть получен с помощью функции f_open.

Возвращает дескриптор предыдущего активного файла.

Пример

```
file_prv=f_with(file_next);
```

f_open – открытие рабочего файла

Параметры:

- путь к файлу
- атрибуты открытия (необязательный, по-умолчанию - 0): O_CREAT, O_EXCL, O_TRUNC, O_APPEND, O_RDONLY (расшифровка – согласно C ANSI)

Открывает рабочий файл. Действие аналогично вызову:

```
open(path, attr|O_BINARY|O_RDWR, S_IREAD|S_IWRITE)
```

Если на момент вызова существует ранее открытый рабочий файл, в который производилась запись – он будет закрыт.

Возвращает дескриптор файла, если файл открыт нормально и -1 при наличии ошибок. Код ошибки передается через переменную errno библиотеки dcl_std_lib: EACCES, EEXIST, EMFILE, ENOENT (расшифровка – согласно C ANSI).

Пример

```
status=f_open("/my_folder/Test.dat", O_TRUNC);
```

f_close – закрытие рабочего файла

Параметры:

- дескриптор файла (необязательный, по-умолчанию - последний активный)

Закрывает рабочий файл, открытый с помощью f_open. Если такового нет – никаких действий не производит.

Возвращает 0 при успешном закрытии файла или -1 – при неудаче. Код ошибки передается через переменную errno биб лиотеки dcl_std_lib: EBADF (расшифровка – согласно C ANSI).

Пример

```
f_close();

f_close(in_file);
```

f_seek – позиционирование рабочего файла

Параметры:

- смещение позиции
- код базы: SEEK_SET, SEEK_CUR, SEEK_END (расшифровка – согласно C ANSI)

Устанавливает новую текущую позицию для рабочего файла, открытого с помощью f_open. Действие аналогично вызову: lseek(..., offset, base).

Возвращает текущую позицию или -1 – при неудаче. Код ошибки передается через переменную errno библиотеки dcl_std_lib: EBADF, EINVAL (расшифровка – согласно C ANSI). При отсутствии открытого рабочего файла возвращает -1 и errno= EBADF.

f_tell – запрос текущей позиции в рабочем файле

Параметров нет.

Выдает текущую позицию для рабочего файла, открытого с помощью f_open. Действие аналогично вызову: tell(...).

Возвращает текущую позицию или -1 – при неудаче. Код ошибки передается через переменную errno библиотеки dcl_std_lib: EBADF (расшифровка – согласно C ANSI). При отсутствии открытого рабочего файла возвращает -1 и errno= EBADF.

f_read – считывание данных из рабочего файла

Параметры:

- буфер для считывания
- счетчик считывания (необязательный, по-умолчанию buffof(buff))

Считывает заданное число единиц данных из рабочего файла, открытого с помощью f_open. Размер единицы данных определяется в соответствии с типом переменной-буфера.

Возвращает число считанных байт или -1 – при ошибке. Код ошибки передается через переменную errno библиотеки dcl_std_lib: EBADF (расшифровка – согласно C ANSI). При отсутствии открытого рабочего файла возвращает -1 и errno= EBADF.

f_readl – считывание строки данных из рабочего файла

Параметры:

- буфер для считывания
- метка конца строки (необязательный)

Считывает данные из рабочего файла, открытого с помощью f_open, либо до обнаружения метки конца строки, либо до конца файла, либо до конца буфера. Если метка конца строки опущена, то работает аналогично f_read со счетчиком считывания, заданным по-умолчанию.

Возвращает число считанных байт или -1 – при ошибке. Код ошибки передается через переменную errno библиотеки dcl_std_lib: EBADF (расшифровка – согласно C ANSI). При завершении файла возвращает 0 и errno= EOF. При отсутствии открытого рабочего файла возвращает -1 и errno= EBADF.

Пример

```
cnt=f_readl(buff, "\r\n");
```

f_readonce – одномоментное открытие, считывание и закрытие файла

Параметры:

- путь к файлу
- буфер для считывания
- счетчик считывания (необязательный, по-умолчанию buffof(buff))

Открывает заданный файл, считывает заданное число единиц данных из файла, после чего закрывает файл. Размер единицы данных определяется в соответствии с типом переменной-буфера.

Возвращает число считанных байт или -1 – при ошибке. Код ошибки передается через переменную errno библиотеки dcl_std_lib: EACCES, EMFILE, ENOENT (расшифровка – согласно C ANSI).

Пример

```
cnt=f_readonce("Test.dat", buff);

cnt=f_readonce("Test.dat", buff, 90);
```

f_write – запись данных в рабочий файл

Параметры:

- данные для записи

Записывает данные в рабочий файл, открытый с помощью f_open. Число записываемы байт определяется как sizeof(buff) .

Возвращает число считанных байт или -1 – при неудаче. Код ошибки передается через переменную errno библиотеки dcl _std_lib: EBADF, ENOSPC (расшифровка – согласно C ANSI). При отсутствии открытого рабочего файла возвращает -1 и errno= EBADF.

Пример

```
cnt=f_write("Test\r\n");
```

f_writeonce – одно-моментное открытие, запись и закрытие файла

Параметры:

- путь к файлу
- буфер данных для записи
- счетчик записи (необязательный, по-умолчанию sizeof(buff))

Открывает заданный файл, записывает заданное число единиц данных в файл, после чего закрывает файл. Размер единицы данных определяется в соответствии с типом переменной-буфера.

Возвращает число записанных байт или -1 – при ошибке. Код ошибки передается через переменную errno библиотеки dcl _std_lib: EACCES, EMFILE, ENOENT, ENOSPC (расшифровка – согласно C ANSI).

Пример

```
cnt=f_writeonce("Test.dat", buff[0…12]);

cnt=f_writeonce("Test.dat", buff, 90);
```

f_unlink – удаление файла

Параметры:

- путь к файлу

Удаляет файл. Действие аналогично вызову:

```
unlink(path)
```

Возвращает 0, если файл удален нормально и -1 при наличии ошибок. Код ошибки передается через переменную errno б иблиотеки dcl_std_lib: EACCES, ENOENT (расшифровка – согласно C ANSI).

Пример

```
status=f_unlink("/my_folder/Test.dat");
```

f_exists – проверка существования файла или раздела

Параметры:

- путь к файлу/разделу

Проверяет, существует ли задаваемый путем файл/раздел. Действие аналогично вызову:

```
access(path, 0x00)
```

Возвращает 0, если файл/раздел существуют или EACCES, ENOENT (расшифровка – согласно C ANSI).

Пример

```
status=f_exists("/my_folder/Test.dat");
```

f_cp_convert – преобразование кодовой страницы текста

Параметры:

- путь к исходному файлу
- путь к преобразованному файлу
- исходная кодировка
- результирующая кодировка

Поддерживаются следующие кодировки:

- CP866
- CP1251

Если указанная кодировка не поддерживается, то выдается ошибка.

Возвращает 0 при успешном преобразовании и -1 при наличии ошибок. Код ошибки передается через переменную errno библиотеки dcl_std_lib: EACCES, ENOENT (расшифровка – согласно C ANSI).

Пример

```
f_cp_convert("c:/export/1.txt", "c:/export/2.txt", "CP1251", "CP866") ;
```

---

f_check – поиск ключевых слов в файле

Параметры:

- путь к анализируемому файлу
- выходной буфер
- ключевое слово 1
- ...
- ключевое слово N

Функция ищет по заданному файлу входы заданных ключевых слов и в случае обнаружения таковых выдает в выходной буфер хвост данных файла, начиная с первого по ходу файла входа одного из ключевых слов.
Таким образом, первое из найденных ключевых слов попадает в начало выходного буфера.

Возвращает:

0 - если ни одно из ключевых слов не найдено
1 - если одно из ключевых слов найдено
-1 - при наличии ошибок (код ошибки передается через переменную errno библиотеки dcl_std_lib, расшифровка – согласно C ANSI).

Пример использования

```
   TEST.TXT   :

123456789_

prefix Key_BB=123456789_123456789_
prefix Key_A=123456789_123456789_123456789_123456789_
prefix Key_CCC=123456789_123456789_123456789_123456789_123456789__123456789_;


      out:

Key_BB=123456789_123


   char  out[20] ;

     int  status ;


         status=f_check("TEST.TXT", out, "Key_A=", "Key_BB", "Key_CCC=") ;
```

```
if(status<0) {
//                          . . .
}


return ;
```

Библиотека работы с SQL-запросами - dcl_sql_lib

Использует объекты системной библиотеки dcl_std_lib.

Константы

E_CALL	-	ошибка структуры вызова
E_PAR_TYPE	-	неподдерживаемый тип переменной
E_UNKNOWN_IFACE	-	неизвестный тип интерфейса
E_CONNECT_HEADER	-	некорректный дескриптор соединения
E_CONNECT_IN_USE	-	повторного открытие активного соединения
E_CURSOR_MAX	-	нет свободных курсоров
E_CURSOR_ HEADER	-	некорректный дескриптор курсора
E_SQL_ERROR	-	ошибка исполнения SQL-оператора
E_FETCH_FILE	-	ошибка работы с файлом результата
E_NO_LOB	-	список параметров не содержит LOB-объектов
E_UNK_CP	-	неизвестная кодовая страница
E_DBF_FILE	-	ошибка работы с DBF-файлом

sql_codepages – задание преобразования кодовых страниц

Параметры:

- кодовая страница SQL-клиента
- кодовая страница выходных данных

Задает преобразование кодовых страниц для некоторых функций и методов, работающих с SQL-запросами. Необходимость использования функции указывается в описании каждой конкретной функции/метода.

Возвращает 0 при успешном завершении или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno.

Пример

```
            status=sql_codepages("CP1251","CP866") ;
         if(status) {
                        $error<=="Code pages assign error : " @ atos(errno) ;
                        return(-1) ;
                     }
```

sql_connect – установление соединение с базой данных

Параметры:

- дескриптор соединения (необязательный)
- тип используемых драйверов
- идентификатор базы данных
- логин: пользователь/пароль
- параметр соединения №1 (HOME) (необязательный)
- параметр соединения №2 (NLS) (необязательный)

Выполняет соединение с базой данных. Если дескриптор соединения не задан, то используется соединение

по-умолчанию.

Тип используемых драйверов:

для Windows – ODBC, DB2

для Solaris – OCI

Если запрошенный тип драйверов не поддерживается - выдается ошибка E_UNKNOWN_IFACE.

Параметр соединения 1:

для OCI – HOME-раздел Oracle

для ODBC – тип используемой базы данных

для DB2 – игнорируется

Параметр соединения 2

для OCI – NLS-раздел Oracle

для ODBC – игнорируется

для DB2 – игнорируется

Возвращает 0 при успешном завершении или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno. При попытке выполнить соединение «поверх» незакрытого соединения выдается ошибка E_CONNECT_IN_USE. При ошибке E_SQL_ERROR текст ошибки может быть запрошен через процедуру sql_error.

При успешном выполнении соединения автоматически выделяются ресурсы для 2 курсоров.

Пример

```
status=sql_connect("OCI", "CAPSTONE", "user/123", "/ora/9", "/ora/9/nls");

status=sql_connect(connect_2, "DB2", "ATLANT", "user/123");
```

sql_disconnect – закрытие соединения с базой данных

Параметры:

- дескриптор соединения (необязательный)

Закрывает соединение с базой данных. Если дескриптор соединения не задан, то используется соединение по-умолчанию.

Возвращает 0 при успешном завершении или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno. При ошибке E_SQL_ERROR текст ошибки может быть запрошен через процедуру sql_error.

> Важно!
> При завершении интерпретатора производится автоматическое закрытие всех незакрытых соединений, открытых с помощью sql_connect.

Пример

```
status=sql_disconnect();

status=sql_disconnect(connect_2);
```

sql_commit – операция COMMIT

Параметры:

- дескриптор соединения (необязательный)

Выполняет операцию COMMIT для заданного соединения. Если дескриптор соединения не задан, то используется соединение по-умолчанию.

Возвращает 0 при успешном завершении или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno. При ошибке E_SQL_ERROR текст ошибки может быть запрошен через процедуру sql_error.

Пример

```
status=sql_commit();

status=sql_commit(connect_2);
```

sql_rollback – операция ROLLBACK

Параметры:

- дескриптор соединения (необязательный)

Выполняет операцию ROLLBACK для заданного соединения. Если дескриптор соединения не задан, то используется соединение по-умолчанию.

Возвращает 0 при успешном завершении или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno. При ошибке E_SQL_ERROR текст ошибки может быть запрошен через процедуру sql_error.

Пример

```
status=sql_rollback();

status=sql_ rollback(connect_2);
```

sql_execute – выполнение модифицирующего SQL-оператора или SQL-блока

Параметры:

- дескриптор соединения (необязательный)
- SQL-оператор
- Список подставляемых переменных SQL-оператора (необязательный)

Выполняет SQL-оператор для заданного соединения. Если дескриптор соединения не задан, то используется соединение по-умолчанию.

Возвращает 0 при успешном завершении или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno. При ошибке E_SQL_ERROR текст ошибки может быть запрошен через процедуру sql_error.

Пример

```
status=sql_execute(connect_1, "delete from clients");

status= sql_execute("delete from clients where client_no=:num", num);
```

sql_writeLOB – занесение LOB_значения

Параметры:

- дескриптор соединения (необязательный)
- SQL-оператор
- Список подставляемых переменных SQL-оператора

Выполняет SQL-оператор для заданного соединения. Если дескриптор соединения не задан, то используется соединение по-умолчанию.

Среди элементов списка параметров LOB-объектами считаются динамические символьные массивы. Список параметров должен включать хотя бы один LOB-объект, в противном случае выдается ошибка S_NO_LOB.

Возвращает 0 при успешном завершении или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno. При ошибке E_SQL_ERROR текст ошибки может быть запрошен через процедуру sql_error.

Пример

```
char  data[] ;   // LOB-объект

char   id[32] ;

…
```

```
status=sql_writeLOB("update Files set Data=:data where file_id=:id", data, id);
```

sql_fetch2csv – выборка данных в CSV-файл

Параметры:

- дескриптор соединения (необязательный)
- Путь к файлу результата
- Разделитель полей в строке данных
- SELECT-оператор
- Список подставляемых переменных SELECT-оператора (необязательный)

Производит выборку по SELECT-оператору для заданного соединения в CSV-файл. Если дескриптор соединения не задан, то используется соединение по-умолчанию.

Возвращает число записанных в файл строк данных или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno. При ошибке E_SQL_ERROR текст ошибки может быть запрошен через процедуру s ql_error.

Пример

```
cnt=sql_fetch2csv(connect_1, "C:\\Temp\1.dat", "\t", "select * from clients");
cnt=sql_ fetch2csv("C:\\Temp\1.dat", "\t", "select * from clients where client_no=:num", num);
```

sql_fetch2dbf – выборка данных в DBF-файл

Параметры:

- дескриптор соединения (необязательный)
- Путь к файлу результата
- Путь к файлу DBF-шаблона
- SELECT-оператор
- Список подставляемых переменных SELECT-оператора (необязательный)

Производит выборку по SELECT-оператору для заданного соединения в DBF-файл. Если дескриптор соединения не задан, то используется соединение по-умолчанию.
Привязка полей выборки к DBF-шаблону осуществляется по именам полей.
При необходимости выполнения перекодировки она должна быть задана функцией sql_codepages.

Возвращает число записанных в файл строк данных или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno. При ошибках E_SQL_ERROR и текст E_DBF_FILE ошибки может быть запрошен через процедуру sql_error.

Пример

```
            status=sql_fetch2dbf("C:\\DATA\\OUT.DBF", "C:\\TEMPLATES\\CLIENTS.DBF", "select * from
   clients") ;

           if(status>=  0         ) {
                                    }
      else if(status==E_SQL_ERROR ) {
                                        sql_error(v_error_text) ;
                                         $error<==v_error_text  ;
                                            return(-1) ;
                                    }
      else if(status==E_DBF_FILE  ) {
                                        sql_error(v_error_text) ;
                                         $error<==v_error_text  ;
                                            return(-1) ;
                                   }
      else if(status==E_FETCH_FILE) {
                                      $error<=="Fetch file error : " @ $file ;
                           return(-1) ;
                                    }

      else                          {
                                      $error<=="Fetch error " @ atos(status) ;
                                          return(-1) ;
                                    }


               sql_commit() ;
```

sql_open – открытие курсора SELECT-оператора

Параметры:

- дескриптор соединения (необязательный)
- SELECT-оператор
- Список подставляемых переменных SELECT-оператора (необязательный)

Открывает курсор и производит подготовку к выборке по SELECT-оператору для заданного соединения. Если дескриптор соединения не задан, то используется соединение по-умолчанию.

Возвращает дескриптор курсора (при успешном завершении) или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno. При ошибке E_SQL_ERROR текст ошибки может быть запрошен через процедуру sql_error.

Для освобождения ресурсов курсора после завершения выборки он должен быть закрыт процедурой sql_close.

Пример

```
cursor=sql_open(connect_1, "select * from clients");

cursor=sql_open("select * from clients where client_no=:num", num);
```

sql_close – закрытие курсора SELECT-оператора

Параметры:

- дескриптор курсора (необязательный)

Закрывает курсор, открытый процедурой sql_open и освобождает соответствующие ресурсы. Если дескриптор курсора не задан, то закрываются все открытые курсоры.

Возвращает 0 при успешном завершении или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno. При ошибке E_SQL_ERROR текст ошибки может быть запрошен через процедуру sql_error.

Пример

```
sql_close(cursor_1);

sql_close();
```

sql_error – запрос текста ошибки

Параметры:

- дескриптор соединения (необязательный)
- буфер для текста ошибки

Выдает текст последней SQL-ошибки для заданного соединения. Если дескриптор соединения не задан, то используется соединение по-умолчанию.

Возвращает 0 при успешном завершении или код ошибки – при неудаче.

Пример

```
sql_error(connect_1, error_text);

sql_error(error_text);
```

Библиотека работы с MS Office - dcl_office_lib

Использует объекты системной библиотеки dcl_office_lib.

Константы

E_OLE_ERROR - ошибка работы с OLE-сервером

excel_template – загрузка данных в Excel-шаблон

Параметры:

- Путь к файлу результата
- Путь к файлу шаблона
- Путь к файлу с загружаемыми данными
- Имя макроса загрузки

Производит исполнение заданного макроса с передачей ему в качестве параметра пути к файлу с загружаемыми данными. После исполнения макроса производится сохранение файла с путем, задаваемым первым параметром и закрытие MS Excel.

Возвращает 0 – при успешном завершении или код ошибки – при неудаче. При ошибке также выставляется значение системной переменной errno. При ошибке E_OLE_ERROR текст ошибки может быть запрошен через функцию get_errtext.

Пример

```
excel_template("C:\\DCL\1.xltm","C:\\Temp\1.dat", "C:\\Result\1.xlsm", "LoadData ");
```