Обзор

Главное меню
DBASIC

Главное меню

Главное меню программы содержит следующие пункты меню:

- Регистрация ХО
- Отчеты
- Настройка
- Утилиты
- Окна
- Выход

В процессе работы могут моявиться следующие пункты меню:

- Запись
- Поле
- Файл
- Правка
- Поиск
- Отчет
- Правка

Регистрация хозяйственных операций

Работа с программой осуществляется методом регистрации хозяйственных операций.

Примеры хоз. операций :

- получение денег в кассу,
- поступление денег на р/с за реализованную продукцию,
- покупка основных фондов,
- начисление и уплата налогов и т.д.

Для того, чтобы зарегистрировать хозяйственную операцию необходимо выбрать пункт меню "Регистрация ХО" и нажать Enter. После этого в правой части экрана появится список хозяйственных операций:

Код	Номер	Название
01	1	Получено с р/с в кассу
02	1	Выдана ЗП
03	1	Аванс выдан
03	2	
04	1	Закрыт выданный аванс
04	2	
05	1	Аванс получен
05	2	
06	1	Закрыт полученный аванс

Вы должны из этого списка выбрать ту операцию, которую хотели бы зарегистрировать. Например, были получены деньги с расчетного счета в сумме 10000 руб. - в этом случае Вы, очевидно, выберите операцию "Получено с р/с в кассу" и нажмете ENTER .

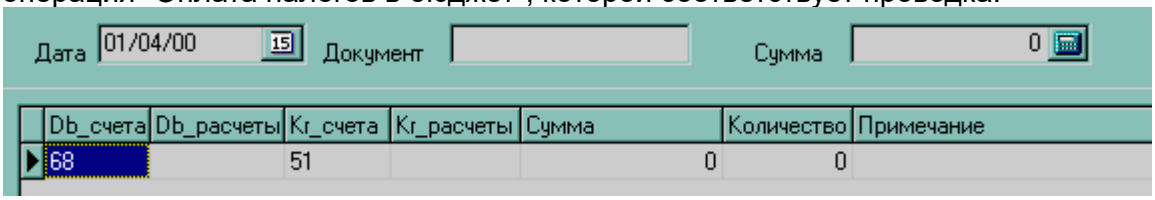
После этого Вам будет предложено ввести дату операции, наименование и номер соответствующего ей первичного документа, сумму, а также содержание операции (некоторое понятное для Вас пояснение регистрируемой операции).

Например, для нашей операции может быть введено:
Дата: 02/04/92 < ENTER >
Документ: ПКО 12(приходный кассовый ордер N 12) < ENTER >
Сумма: 10000 < ENTER >
Содержание: Получено на ЗП < ENTER >

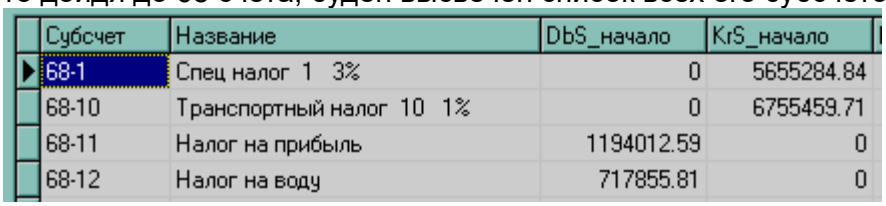
Далее в правом верхнем углу появится список проводок, соответствующих данной операции и Вам будет предложено:
1. Нажать Enter и зарегистрировать операцию - в том случае,если Вас устраивает предложенный вариант ее регистрации;
2. Нажать F4 и редактировать список проводок - если предложенный вариант Вас в чем-то не устроил.
3. Нажать Esc и откатиться - если Вы раздумали регистрировать операцию или решили откатиться на предыдущий уровень.

После того,как операция была зарегистрирована или произошла откатка Вы можете приступать к регистрации следующей операции и т.д.

И, наконец, последний момент, который необходимо здесь обсудить. Если при регистрации некоторой операции встретится аналитический счет, то Вам будет предложен список всех субсчетов данного счета. Например, если регистрируется операция "Оплата налогов в бюджет", которой соответствует проводка:



то дойдя до 68 счета, буден высвечен список всех его субсчетов в виде:



В этой ситуации пользователю предлагается выбрать необходимый субсчет и нажать Enter.

После того, как все проводки будут сформированы Вы можете:
1. Нажать 'Принять' и зарегистрировать операцию - в том случае,если Вас устраивает предложенный вариант ее регистрации;
2. Нажать 'Отменить' и откатиться - если Вы раздумали регистрировать операцию.

Справочник проводок

Все проводки для всех регистрируемых операций записываются в единую таблицу под названием "Справочник проводок".Этот справочник может Вами просматриваться и корректироваться.
После выбора подпункта "Справочник проводок" пункта главного меню "Регистрация ХО" появится таблица:

Номер	Дата	Докум	Дб.счета	Дб.субсчет	Кр.счета	Кр.субсчет	Сумма	Коли	Примечание
1	23/03/2000		20		10	0010042	7955.22	6	6 кг Лак БТ-577 1325.87
2	23/03/2000		20		10	001106	35326.5	6	6 кг Лак Каменноут 5887.75
3	23/03/2000		20		10	001106	459244.5	78	78 кг Лак Каменноут 5887.75

В этой таблице:
Номер - номер соответствующей проводки,
Дата - дата операции,
Документ - наименование и номер документа,
Сумма - проводимая сумма,
Примечание - примечание для проводки.

Для просмотра и корректировки справочника проводок пользуйтесь функциональными клавишами:
F3 - просмотр содержимого проводки;
F4 - редактирование проводки;
F6 - вставка новой проводки;
SHIFT+F8 - удаление проводки;
F9 - ввод значения для поиска по активному полю;
ALT+F9 - повторный поиск значения по активному полю;

Для сортировки проводок щелкните мышкой по заголовку соответствующего столбца.

Документы

Данный пункт предназначен для формирования и печати любых бухгалтерских документов. Многие документы, используеме при работе уже настроены и входят в комплект поставки. При необходимости изменить настройку или ввести новые документы необходимо воспользоваться соответствующим пунктом настройки.

Чтобы начать работу с документами установите курсор на пункт "Документы" и нажмите Enter. Появится перечень документов. Выберите нужный и вновь нажмите Enter. Появится реестр уже созданных ранее документов.

Регистрация отложенных операций (ОО)

Способ регистрации отложенных операций полностью совпадает с описанием, приведенным в пункте Регистрация хозяйственных операций. Визуально для пользователя разницы никакой нет. Отличие состоит лишь в том, что отложенная операция не обсчитывается системой в данный момент, временно, вплоть до особого указания система ничего не знает о ее наличии.

Просмотр и модификация справочника ОО

В этом режиме Вам предоставляется возможность просматривать и изменять зарегистрированные ранее отложенные операции.

Перенос отложенных проводок в систему

При переносе проводок из справочника ОО в систему Вам предоставляется возможность:

- переносить проводки выборочно;
- перенести все проводки.

Если был выбран пункт "выборочно", то перед Вами появится перечень всех отложенных проводок. Для переноса конкретной проводки в систему установите на нее черный курсор-полосу и нажмите ENTER. Для переноса группы проводок, отвечающих определенным условиям нажмите F10. Перед Вами появится карточка фильтра, позволяющая задать диапазон дат, номеров документов, указать через запятую номера счетов по дебету и кредиту.

Пример:

	Минимум	Максимум
Дата	01/01/97	01/02/97
Документ		
Счета по дебету	10,12	
Счета по кредиту	10,12	

При таком заполнении фильтра в систему будут перенесены все проводки датированные январем 1997 года у которых по дебету ИЛИ по кредиту встречаются счета 10 ИЛИ 12.

Перенос проводок из системы в отложенные

При использовании этого режима проводки могут быть изъяты из системы и перенесены в справочник отложенных операций.

Для переноса проводок из системы в справочник ОО выберите соответствующий пункт меню и нажмите ENTER. Вам предоставляется возможность:

- переносить проводки выборочно;
- перенести все проводки.

Если был выбран пункт "выборочно", то перед Вами появится перечень всех проводок. Для переноса проводки из системы в отложенные установите на нее черный курсор-полосу и нажмите ENTER.

Если был выбран пункт "по фильтру", то перед Вами появится карточка фильтра, позволяющая задать диапазон дат, номеров документов, указать через запятую номера счетов по дебету и кредиту.

Пример:

	Минимум	Максимум
Дата	01/01/97	01/03/97
Документ		
Счета по дебету	10,12	
Счета по кредиту		

При таком заполнении фильтра из систему будут перенесены все проводки датированные январем или февралем 1997 года, у которых по дебету встречаются счета 10 ИЛИ 12.

Сдать в архив

После того, как были зарегистрированы все хозяйственные операции, относящиеся к текущему отчетному периоду и распечананы отчетные документы, Вам необходимо сдать информацию в архив.

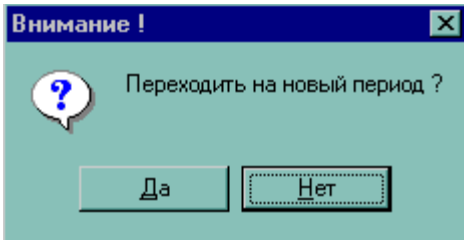
ВНИМАНИЕ! ЭТОТ ПУНКТ НЕОБХОДИМО ВЫПОЛНИТЬ НЕПОСРЕДСТВЕННО ПЕРЕД ТЕМ, КАК БУДЕТ ОСУЩЕСТВЛЕН ПЕРЕХОД НА НОВЫЙ ОТЧЕТНЫЙ ПЕРИОД.

Новый период

По окончании регистрации всех хозяйственных операций и получении необходимых документов за отчетный период(месяц,квартал) Вам понадобится один из режимов переходов:

- НОВЫЙ ПЕРИОД;
- НОВЫЙ ГОД.

ВНИМАНИЕ: ПРИ ПЕРЕХОДЕ В СОСТОЯНИЕ НОВОГО ПЕРИОДА ПРОИСХОДИТ СБРОС СПРАВОЧНИКА ОПЕРАЦИЙ И СПРАВОЧНИКА ПРОВОДОК. ПОЭТОМУ ДЛЯ ПРЕДОТВРАЩЕНИЯ СЛУЧАЙНОГО СБРОСА ВВЕДЕННЫХ ДОКУМЕНТОВ ПОЯВИТСЯ ОКНО, ТРЕБУЮЩЕЕ ПОДТВЕРЖДЕНИЯ ПРИНЯТОГО ВАМИ РЕШЕНИЯ :



Если Вы уверены в том, что нужно осуществить переход на новый период, то переместите курсор - на кнопку " ДА " и нажмите Enter.

Кроме сброса произойдет перевод остатков в оборотной ведомости и обнулится движение, а также произойдет перевод остатков в аналитических справочниках.

Новый год

При переходе на новый год будут проделаны те же действия, которые были произведены при переходе на новый период, а также:

- будут перенесены остатки в активе и пассиве баланса;
- будут очищены обороты в главной книге.

Оборотно - сальдовая ведомость

Оборотная ведомость содержит информацию по оборотам и конечным сальдо для всех счетов, указанных в плане счетов. После того, как перед Вами появится документ Вы сможете распечатать его.

Расшифровки (журналы - ордера)

Расшифровка - документ, позволяющий расшифровать каждую строку оборотной ведомости по дебету и кредиту. В пункте " Расшифровка" имеются два подпункта:

- По счету
- По всем счетам

Если Вы хотите получить расшифровки по всем счетам, то выберите соответствующий пункт меню и нажмите Enter.

При необходимости получить расшифровку по конкретному счету выберите первый пункт меню и нажмите Enter, когда появится список всех счетов, выберите интересующий и нажмите Enter.

Отчеты по аналитическим счетам

Для любого аналитического счета могут быть получены два вида отчетов:

1. Оборотно-сальдовая ведомость по субсчетам, содержащая информацию о входящих остатках, оборотах и исходящих остатках по каждому из субсчетов.

2. Расшифровки для субсчетов, включающие в себя информацию о движении средств по данному субсчету.

Анализ счета по датам

Анализ счета по датам - это документ, позволяющий проанализировать движение средств по счету внутри отчетного периода в зависимости от дат совершения хозяйственных операций.

Такой анализ имеет смысл производить, например, для 51 счета, 50 счета, а также для подобных им счетов.

Главная книга

Документ "главная книга" содержит информацию об оборотах счета с другими счетами за текущий месяц и с начала года. Документ представляет собой последовательность блоков для каждого счета. Каждый такой блок заменяет страницу в обыкновенной главной книге.

Актив баланса

Актив и Пассив - это две части одного и того же документа: Баланс предприятия. Входными данными для баланса являются результаты расчета оборотной ведомости.

Пассив баланса

Актив и Пассив - это две части одного и того же документа: Баланс предприятия. Входными данными для баланса являются результаты расчета оборотной ведомости.

Произвольные отчеты по аналитическим счетам

Примеры формирования произвольных отчетов по аналитическим счетам можно найти
в Редактирование отчета.

Произвольные выборки проводок

Произвольные выборки проводок могут быть получены для любого балансового счета за любой из 12 предыдущих месяцев, а также за текущий месяц.

Порядок формирования произвольных выборок полностью совпадает с описанием, приведенным в пунктах Редактирование отчета и Произвольные отчеты по СП.

Произвольные отчеты по СП

Порядок формирования произвольных отчетов по справочнику проводок полностью совпадает с описанием, приведенным в пункте Произвольные отчеты по аналитическим счетам. В качестве иллюстрации приведем следующий пример.

Пример: Выбрать из справочника проводок все проводки, у которых по дебету стоит 60 счет, а по кредиту 51. Расположить их в порядке возрастания дат.

Таблица будет выглядеть так:

Поле	Видим	Тип	Формат	Сумма	Разд	Сорт	Мин	Макс
Раб.место	-	N	5.0	-	-	-		
N_проводки	-	N	4.0	-	-	-		
Дата	+	D	8.0	-	-	1		
Документ	+	C	10.0	-	-	-		
Db_расчеты	-	C	10.0	-	-	-		
Kr_расчеты	-	C	10.0	-	-	-		
Db_счета	+	C	8.0	-	-	-	60	60
Kr_счета	+	C	8.0	-	-	-	51	51
Сумма	+	N	16.2	+	-	-		
Количество	-	N	10.3	-	-	-		
Примечание	+	C	50.0	-	-	-		

А сам отчет так:

Дата	Документ	Db_счета	Kr_счета	Сумма	Примечание
01/10/94	пп 123	60	51	1456.00	Предприятию 1
02/10/94	пп 34	60	51	2456.00	Предприятию 2
10/10/94	пп 56	60	51	3456.00	Предприятию 3
12/10/94	пп 678	60	51	4456.00	Предприятию 4
15/10/94	пп 555	60	51	5456.00	Предприятию 5
20/10/94	пп 679	60	51	6456.00	Предприятию 6
25/10/94	пп 345	60	51	7456.00	Предприятию 7
				XXXXX.XX	

Справки

Режим "Справки" позволяет получать отчеты о движении средств по любому из субсчетов любого аналитического счета за последние 12 месяцев.

Чтобы получить справку необходимо:
а) указать период формирования справки;
б) выбрать счет и субсчет, по которым необходимо сформировать справку.

Карточка объекта учета

Анализ оборотов по признакам

Субсчета в зависимости от принадлежности к тому или иному счету могут обладать различными признаками. Например, субсчета материальных счетов (10, 12) имеют признаки:
1. Склад - наименование склада или подотчетного лица.
2. Группа - наименование или код группы материалов (Основные материалы, тара и т.д.)

Субсчета счетов основных средств (01 ,04) имеют признаки:
1. Материально-ответственное лицо.
2. Подразделение.
3. Группа основных средств.
4. Шифр износа и т. д.

АНАЛИЗ СУБСЧЕТОВ ПО ПРИЗНАКАМ позволяет получать информацию об остатках и движении средств на субсчетах, имеющих сходные признаки. Например, для материальных счетов можно проанализировать:
1. Движение средств в разрезе складов.
2. Движение средств в разрезе групп.
3. Движение средств в разрезе складов и групп (т.е. по двум признакам одновременно).

Для счетов основных средств можно проанализировать движение ОС по видам, группам, подразделениям и т. д.

Чтобы получить необходимую информацию проделайте следующее:
1. Выберите интересующий Вас счет, например 10, и нажмите Enter.
2. При помощи клавиш стрелка вниз и стрелка вверх установите курсор напротив нужного Вам признака, например "Склад", и нажмите Enter. Если Вы хотите получить анализ одновременно по двум признакам, то установите курсор напротив второго признака, например "Группа", и вновь нажмите Enter. Анализ производится не более, чем по двум признакам одновременно.
3. Нажмите клавишу F9 и через некоторое время Вы получите отчет.

Дополнительные отчеты

Отчеты, которые могут формироваться системой весьма многообразны и во многом зависят от характера задач, решаемых на том или ином рабочем месте. Ввиду этого и применен режим "Дополнительные отчеты".

Система снабжена рядом стандартных отчетов:

- оборотная ведомость;
- расшифровки оборотной ведомости;
- главная книга;
- анализ счета по датам.

Помимо стандартных отчетов Вы можете при настройке указать и создать дополнительные que-отчеты. Кроме того, прикладные программисты могут написать и свои программы, формирующие тот или иной отчет.

Для того, чтобы добавить новый отчет в список отчетов, необходимо вставить новую карточку, поместив в нее информацию о названии и способе вызова дополнительного отчета.

В поле "Способ вызова" может быть внесено:
а) Имя btr файла в виде: имя.btr - если создается новый отчет по указанному btr - файлу;
б) имя que файла в виде: имя.que - если необходимо сформировать уже ранее созданный отчет (см. Редактирование отчета);
в) имя bas файла в виде: имя.bas - если необходимо сформировать отчет в налоговую инспекцию. Для того, чтобы войти в режим редактирования bas-файла необходимо нажать комбинацию клавиш Alt+E.
г) командная строка, активизирующая программу формирования и печати отчета.

Изменить отчетный период

Система "УЧЕТ ФИНАНСОВ" позволяет формировать документы как за текущий отчетный период, так и за предыдущие отчетные периоды. Это такие документы, как:

- оборотная ведомость;
- расшифровки;
- отчеты по аналитическим счетам;
- анализ счета по датам;
- главная книга;
- актив и пассив баланса;
- произвольные отчеты по аналитическим счетам,
- произвольные выборки проводок,
- справки.

Остальные документы: "Произвольные отчеты по СП", "Дополнительные отчеты", - никак не реагируют на изменение отчетного периода.

Пункт "Предыдущий" используется при необходимости получить документы за любой из предыдущих отчетных периодов.

Чтобы изменить отчетный период проделайте следующее:
1. Выберите пункт "ПРЕДЫДУЩИЙ" и нажмите Enter - появится перечень отчетных периодов.

Год	Месяц	Название
1999	1	Январь
1999	2	Февраль
1999	3	Март
1999	4	Апрель
1999	5	Май
1999	6	Июнь
1999	7	Июль
1999	8	Август

2. Выберите интересующий Вас период и нажмите Enter

Далее для получения соответствующих отчетов пользуйтесь пунктами: "Оборотно - сальдовая ведомость", "Расшифровки" и т.д.

После выбора пункта меню "Текущий", система будет формировать отчеты, относящиеся к текущему отчетному периоду.

План счетов

Настройка плана счетов - это то, с чего необходимо начинать работать с программой.

В эту таблицу Вы должны внести те счета, которые используются бухгалтерией Вашего предприятия. При вводе каждого счета вводится следующая информация:

- название балансового счета;
- номер балансового счета;
- код статьи актива, на которую будет отнесено сальдо счета;
- код статьи пассива, на которую будет отнесено сальдо счета;
- признак аналитического учета. В качестве признака необходимо проставить одно из значений:

" " - пустое поле будет означать, что счет не является аналитическим, т.е. не имеет субсчетов, обычно это такие счета, как: 50, 51 и т.д. Пустое поле может использоваться также и в том случае, когда счет является аналитическим, но Вы по каким-то причинам не хотели бы пока настраивать его аналитику;

"РС" - это значение указывает на то, что счет является аналитическим, причем по счету ведется только стоимостной учет (рекомендуется использовать для счетов: 67, 68, 69 и им подобным);

"МТ" - счет является аналитическим и по нему ведется стоимостной и количественный учет (рекомендуется использовать для материальных счетов: 10, 12, 41 и т.д.);

"ОС" - счет является аналитическим и на нем будут учитываться основные средства (рекомендуется использовать для счетов: 01, 04).

- признак накопления оборотов по субсчетам. Этот признак применяется в том случае, если Вы хотели бы видеть нарастающим итогом с начала года обороты не только по счету вцелом (эту информацию содержит обычная главная книга), но и в разрезе его субсчетов.

Приведем пример:

Рассмотрим счет номер 68 - расчеты с бюджетом. Этот счет имеет субсчета:
68-1 Подоходный налог
68-2 Налог на прибыль
68-3 Еще какой-то налог и т.д.

Очевидно, что бухгалтеру необходимо знать не только, сколько он, например, заплатил всего налогов с начала года, но также и то, сколько было заплачено подоходного налога, налога на прибыль и т.д.
Для того, чтобы накапливались обороты не только по счету, но и в разрезе его субсчетов необходимо в поле "Обороты" ввести одно из значений: "Д" или "д".

Если в ходе работы возникает необходимость модифицировать план счетов, то рекомендуем это сделать в начале отчетного периода, перед тем, как Вы начнете регистрировать хозяйственные операции.

Редактирование таблицы счетов разрешено только в режиме главного бухгалтера.

Субсчета

При настройке плана счетов некоторые счета Вы пометили значениями "РС" "МТ" или "ОС" в колонке "Расчеты". Тем самым Вы указали, что по этим счетам будет вестись соответствующий аналитический учет. Для настройки аналитики вышеупомянутых счетов и используется режим "Субсчета".

Каждый аналитический счет может иметь неограниченное количество субсчетов.

Если счет имеет "РС" аналитику, то для каждого его субсчета необходимо определить следующую информацию:
1. В графе "Субсчет" проставить код субсчета (любая строка длиной до 9 символов).
2. В графе "Код" проставить код предприятия (сотрудника и т.д.). Эта графа не является обязательной для заполнения. Ее нужно использовать в тех случаях, когда хочется указать на принадлежность вводимого субсчета какой-то конкретной организации, сотруднику и т. д. С одной и той же организацией может быть связан целый ряд субсчетов (в том случае, когда на каждый новый договор с организацией заводится отдельный субсчет).
3. В графе "Название" проставить название субсчета (это может быть название налога, предприятия, ФИО работника и т.д.).
4. В графе "DbS_начало" - дебетовый остаток по субсчету на начало отчетного периода.
5. В графе "KrS_начало" - кредитовый остаток по субсчету на начало отчетного периода.
6. В графах "Актив" и "Пассив" - кодовые строки актива и пассива, на которые будет отнесено сальдо данного субсчета. (См. настройку актива и пассива).

Если счет имеет "МТ" аналитику, то для каждого его субсчета необходимо определить следующую информацию:
1. В графе "Субсчет" проставить код субсчета (в данном случае это, очевидно, можно рассматривать, как код карточки материала, МБП, товара и т. д.)
2. В графе "Склад" проставить код (название) склада (т.е. того места, где находится материал (товар)).
3. В графе "Группа" проставить код группы материалов (товаров).
4. В графе "Название" - название материала (товара).
5. В графе "Начальное количество" - количество материала (товара) на начало отчетного периода.
6. В графе "Начальная сумма" - общая стоимость материала (товара) на начало отчетного периода.

Если счет имеет "ОС" аналитику, то для каждого его субсчета необходимо определить следующую информацию:
1. В графе "Субсчет" проставить код субсчета (рекомендуем субсчет завести в виде: 0200034, где 020 - код материально-ответственного лица, а 0034 - инвентарный номер соответствующего основного средства).
2. В графе "Название" - название основного средства.
3. В графе "Дата" - дату ввода в эксплутацию.

4. В графе "МО лицо" - ФИО или табельный номер материально-ответственного лица, на котором числится данное основное средство;

5. В графе "Группа" - код группы основных средств согласно справочника норм амортизации;

6. В графе "Бал.стоимость" - балансовая стоимость основного средства;

7. В графе "Износ" - общая сумма износа данного основного средства;

8. В графе "Остаточная" - остаточная стоимость основного средства;

9. В графе "Бал.счет" - номер балансового счета, на который будет относиться амортизация данного ОС (20, 23, 26);

10. В графе "Шифр" - шифр амортизации согласно справочника норм амортизации;

11. В графе "Процент амортизации" - годовой процент амортизации от стоимости или от пробега;

12. В графе "Краткая характеристика" - краткая характеристика основного средства;

13. В графе "DbS_начало" - дебетовый остаток по субсчету на начало отчетного периода (очевидно, что он должен быть равен балансовой стоимости);

14. В графе "KrS_начало" - кредитовый остаток по субсчету на начало отчетного периода (в подавляющем большенстве случаев для счетов основных средств он будет равен 0.00).

Справочник операций

Внесите в эту таблицу те типовые хозяйственные операции, которые часто используются Вашим предприятием на настраиваемом рабочем месте.

В поле "Код операции" вносится код хозяйственной операции;
в поле "Номер проводки" - номер проводки в операции (операция может состоять из нескольких проводок);
в поле "Название" - название операции;
в поле "Db_счета", "Db_субсчет" и "Kr_счета", "Kr_субсчет" - номера счетов и субсчетов всех проводок, соответствующих данной операции.
Поля "Db_субсчет" и "Kr_субсчет" необходимо заполнить лишь в том случае, когда в проводке всегда используется один и тот же субсчет;
в поле "Процент" - процент, на который будет домножаться сумма, вводимая при регистрации операции;
в поле "Программа" - имя файла, содержащего программу, которая должна расчитать и вернуть сумму текущей проводки регистрируемой операции;
в столбец "Шифр" - шифр текущей проводки регистрируемой операции;
в столбец "Примечание" - краткое пояснение для каждой из проводок вводимой операции.

Это краткое пояснение может представлять собой:
а) Обычное простое пояснение в виде: "Аванс выдан" или "Получено с расчетного счета в кассу".
б) Символьное выражение, целью которого является автоматическое формирование примечания. В символьное выражение могут входить:

- Строки символов, заключенные в кавычки. Например: "Поступили деньги от " или "Аванс выдан ".
- Содержимое полей аналитических справочников счетов, входящих в проводку. Например: #ДНазвание - будет взято содержимое поля "Название", взятое из аналитического справочника счета, указанного в дебете проводки; #КЕд.изм. - будет взято содержимое поля "Ед.изм.", взятое из аналитического справочника счета, указанного в кредите проводки.
- Содержимое поля "Количество" регистрируемой проводки. Например: #ПКоличество. Остальная информация проводки попадает во все необходимые документы и нет необходимости указывать ее дважды.
- Строка #В. Указание данной строки в символьном выражении будет означать, что при регистрации операции не нужно делать остановку с целью дополнительного редактирования содержимого примечания.

Элементы символьного выражения разделяются знаком +.

Примеры:

Дебет	Кредит	Примечание

1. Проводка 10 60 "Материал "+#ДНазвание+" поступил от "+#КНазвание
Сформированное примечание может выглядеть так:
Материал Краска масляная поступил от Лакокрасочный завод

Дебет Кредит Примечание
2. Проводка 68 51 "Уплачен в бюджет "+#ДНазвание+" за "
Сформированное примечание может выглядеть так:
Уплачен в бюджет Налог на прибыль за.

Некоторым хозяйственным операциям может соответствовать несколько проводок. В этом случае справочник будет выглядеть следующим образом:

Код	Номер	Название	Db_счета	Kr_счета	Процент	П
1	1	Получ. с Р/с	50	51	1	
2	1	Оплата услуг	76#	51	1	
2	2		19#	76#	0.1667	
2	3		20	76#	0.8333	
2	4		68	19#	0.1667	
3	1	Выдана ЗП	70	50	0	
4	1	Начислена амор-ия	20	02	0	

Видно, что операции 2 соответствует четыре проводки.

Обратите внимание на счета, помеченные символом "#". В сложных операциях, включающих в себя по нескольку проводок, очень часто могут повторяться аналитические счета. В нашем случае во второй операции трижды повторяется 76 счет. Он является аналитическими, а это означает, что при регистрации операции необходимо будет три раза указывать, какой субсчет имеется ввиду. Очевидно, что если субсчет один и тот же, тот нет смысла указывать его трижды. Пометка аналитических счетов символом "#" и позволяет избежать указанной ситуации. Кроме символа "#" аналитические счета могут быть также помечены символом "@", например, указанная выше таблица выглядела бы следующим образом:

Код	Номер	Название	Db_счета	Kr_счета	Процент
1	1	Получ. с Р/с	50	51	1
2	1	Оплата услуг	76@	51	1
2	2		19#	76@	0.1667
2	3		20	76@	0.8333
2	4		68	19#	0.1667
3	1	Выдана ЗП	70	50	0
4	1	Начислена амор-ия	20	02	0

Пометка аналитического счета символом "@" означает, что при повторной регистрации хозяйственной операции (в нашем случае это операция с кодом 2) аналитический справочник 76 счета будет автоматически выставляться на субсчет, который был указан при предыдущей регистрации этой хоз. операции.

Примечание: Указанное выше правило работает лишь в тех случаях, когда к аналитическому справочнику счета НЕ ПРИМЕНЯЕТСЯ никакая фильтрация.

Обсудим подробнее заполнение столбцов "Процент", "Программа" и "Шифр".
1. Поле "Процент" используют с целью:

- начисления налогов на сумму;

- выделения налогов из суммы;
- распределения суммы по процентам и т.д.

При заполнении таблицы Вы должны напротив каждой из проводок в поле "Процент" проставить коэффициент, на который будет домножаться сумма, вводимая Вами при регистрации соответствующей хозяйственной операции.

2. Поле "Программа". (Работа с программами требует определенной квалификации, поэтому бухгалтера могут лишь бегло ознакомиться с содержимым этого пункта.)

Это поле используется для автоматизации таких участков учета, как:

- расчет амортизации основных средств;
- начисление зарплаты и т.д.

Например, для того, чтобы сделать проводку по начислению амортизации за текущий месяц, бухгалтер должен сначала рассчитать амортизацию для каждого основного средства и сложить полученные величины. Можно, конечно, проделать эти действия вручную и провести уже готовую сумму, но в программе предусмотрена и иная возможность - автоматический расчет.

Рассмотрим поэтапно, как это можно сделать.

а) Введем в справочник операций строку:

Код	Номер	Название	Db_счета	Kr_счета	Процент	Программа	Примечание
4	1	Начислена амор-ия	20	02	0	amort.bas	Нач.аморт.

Обратите внимание на содержимое поля "Программа". Файла "amort.bas" с программой расчета амортизационных отчислений пока не существует, мы лишь продекларировали его название.

б) Создадим файл, содержащий программу. Для этого поставим при помощи стрелочек маркер на поле "Программа" и нажмем клавиши Alt+E. При этом мы окажемся в текстовом редакторе, где и произведем ввод текста программы:

dbas.hlpПример программы с разбором

См. также dbas.hlpDBASIC

Главная книга

Настройка главной книги производится в том случае, когда Вы устанавливаете ПП в середине года, но хотели бы получать обороты с начала года.

Для того, чтобы произвести настройку Вам нужно будет последовательно ввести итоговые обороты по дебету каждого счета с кредита счетов (согласно данных главной книги).

Для тех счетов, по которым необходимо накапливать обороты в разрезе субсчетов, произведите соответствующую настройку с указанием номеров субсчетов в полях "Дб.субсчет" и "Кр.субсчет".

Актив

Настройка актива состоит в том, что необходимо ввести все статьи активной стороны баланса, их коды (согласно соответствующих бланков бухучета), а также остатки по статьям на начало года.

Текущие остатки формируются системой автоматически, согласно настройки плана счетов и аналитики "РС" счетов.

В графе "Метка" некоторые статьи могут быть помечены одним из двух значений:

- '+' (эта статья является результирующей для всех нижеследующих статей, вплоть до очередной статьи, помеченной символом '+');
- '=' (эта статья является итогом раздела).

Порядок формирования текущих остатков следующий:

а) Для всех активных или пассивных счетов:
Текущий остаток = Текущий остаток + (Дебетовое сальдо счета на конец месяца - Кредитовое сальдо счета на конец месяца).

б) Для всех активно-пассивных счетов с ЗП аналитикой:
Текущий остаток = Текущий остаток + Дебетовое сальдо счета на конец месяца.

в) Для всех активно-пассивных счетов с РС аналитикой:
Программа просматривает все субсчета, при этом действует следующим образом:

- если у субсчета имеется ненулевой дебетовый остаток на конец месяца и в карточке субсчета указан код статьи актива, то остаток будет отнесен именно на эту статью;
- если у субсчета имеется ненулевой дебетовый остаток на конец месяца и в карточке субсчета не указан код статьи актива, то остаток будет отнесен на статью, указанную для счета в графе Актив при настройке плана счетов.

Пассив

Настройка пассива состоит в том, что необходимо ввести все статьи пассивной стороны баланса, их коды (согласно соответствующих бланков бухучета), а также остатки по статьям на начало года.

Текущие остатки формируются системой автоматически, согласно настройки плана счетов и аналитики "РС" счетов.

В графе "Метка" некоторые статьи могут быть помечены одним из двух значений:

- '+' (эта статья является результирующей для всех нижеследующих статей, вплоть до очередной статьи, помеченной символом '+');
- '=' (эта статья является итогом раздела).

Порядок формирования текущих остатков следующий:
а) Для всех активных или пассивных счетов:
Текущий остаток = Текущий остаток + (Кредитовое сальдо счета на конец месяца - Дебетовое сальдо счета на конец месяца).
б) Для всех активно-пассивных счетов с ЗП аналитикой:
Текущий остаток = Текущий остаток + Кредитовое сальдо счета на конец месяца.
в) Для всех активно-пассивных счетов с РС аналитикой.
Программа просматривает все субсчета, при этом действует следующим образом:

- если у субсчета имеется ненулевой кредитовый остаток на конец месяца и в карточке субсчета указан код статьи пассива, то остаток будет отнесен именно на эту статью;
- если у субсчета имеется ненулевой кредитовый остаток на конец месяца и в карточке субсчета не указан код статьи пассива, то остаток будет отнесен на статью, указанную для счета в графе Пассив при настройке плана счетов.

Остатки счетов

При настройке остатков необходимо ввести остатки неаналитических счетов. Для аналитических счетов сальдо формируется автоматически при настройке соответствующей аналитики. Если сальдо некоторого аналитического счета не совпадает с ожидаемым значением, то необходимо корректировать соответствующий этому счету аналитический справочник.

Документы

Настройка документов - это режим, позволяющий модифицировать уже существующие документы, а также создавать новые. Работа по настройке документов предусматривает, что Вы владеете знаниями баз данных, а также освоили макроязык, включенный в систему "УЧЕТ ФИНАНСОВ".

Порядок создания документа разберем на примере приходного кассового ордера.

1. Установите курсор на пункт "Документы" и нажмите Enter. Появится перечень уже имеющихся в системе документов.
2. Для создания карточки нового документа нажмите F6 (Если необходимо модифицировать уже существующий документ, то нажмите F4.).
3. Перед Вами появится пустая карточка документа, которая содержит поля:

3.1. П/номер - порядковый номер документа. Введем: "001".

3.2. Название - название документа.

Введем: "ПРИХОДНЫЙ КАССОВЫЙ ОРДЕР".

3.3. База - имя файла БД, который описывает структуру документа и, одновременно, будет являться реестром документов. Файл БД должен содержать два обязательных поля:

а) Номер_док. C 5.0 (Будет содержать номер документа. Следующий по порядку номер присваивается автоматически.)

б) Дата D 8.0 (Будет содержать дату документа. По умолчанию устанавливается текущая дата.)

Остальные поля вводятся Вами исходя из структуры документа.

Введем: "pko.btr". Файл pko.btr пока отсутствует. Для того, чтобы создать его нажмите Alt+F. Мы вошли в режим создания (или модификации, если файл уже существует) файла БД. Работая в этом режиме пользуйтесь подсказкой по F1.

При модификации существующего файла БД допустимо изменять названия, типы и длины полей, а также менять поля местами. При этом информация пропадать не должна. Старый btr файл сохраняется с расширением bak, а старый файл описания структуры записи получает новое имя: остарое_имя.str.

При вводе названий, длин и типов полей, заполнение которых будет происходить при помощи справочников, необходимо помнить, что такое заполнение станет возможным только при соответствии, как минимум, названий и типов.

3.4. Документ DOS- имя файла с расширением bas, который содержит программу формирования документа DOS.

Документы DOS настраиваются и печатаются на тех рабочих местах, где установлены матричные принтера, а также в том случае, если не нужна "особая"

красота отчета.
Редактирование программы по нажатию Ctrl+D.

3.5. Документ WIN- имя файла с расширением bas, который содержит программу формирования документа WINDOWS.

Документы WIN настраиваются и печатаются на тех рабочих местах, где установлены лазерные или струйные принтера.
Редактирование программы по нажатию Ctrl+W.

3.6. Структура документа - имя файла, хранящего структуру документа WINDOWS.
При помощи специального редактора Вы можете нарисовать документ так, как Вам хочется(например, включить логотип фирмы и т.д.). Далее нарисованный Вами макет
используется программой формирования документа WINDOWS.
Вызов редактора структуры документа по нажатию CTRL+I.

3.7. Проводки - имя файла с расширением bas, который содержит программу формирования проводок, связанных с первичным документом. Если проводки не нужны, то данный пункт можно не заполнять.
Макроязык описан в разделе dbas.hlpDBASIC, а для того, чтобы разобраться с порядком создания программы, необходимо обратиться к документам, которые уже настроены.
Введем в поле "Проводки" значение: "pko_p.bas". Для того,чтобы войти в режим редактирования нажмите CTRL+P.

3.8. Реестр - имя que-файла. Этот файл необходим для того, чтобы распечатать реестр документов. Que - файл создается при помощи генератора отчетов, который имеется в системе. Для того, чтобы вызвать генератор нажмите CTRL+R.
Настройте отчет и запомните его в виде que-файла при выходе. Имя que - файла введите в поле "Печать".

На этом настройка документа окончена. Текущая работа с документами ведется из подпункта Документы пункта Регистрация ХО главного меню.

Объекты учета

В качестве объектов учета в программе обычно выступают:
- предприятия;
- сотрудники;
- материалы.

Объект учета может иметь любое количество субсчетов на различных балансовых счетах.
Например, одно и то же предприятие может выступать в качестве поставщика (счет 60),
покупателя (счет 62), оказывать нам некие услуги (счет 76) и т.д. По этой причине возникает
задача сбора информации о предприятии (сотруднике) "вцелом". Часто бухгалтера называют
подобную выборку актом сверки. В нашей бухгалтерской программе с этой задачей хорошо

справляется отчет "Анализ объекта учета".

Для каждого вида объектов учета ведется свой номенклатурный справочник:
- справочник предприятий;
- справочник сотрудников;
- справочник материалов.

Чтобы открыть интересующий справочник дважды щелкните по нему левой кнопкой мыши.

Список рабочих мест

Список рабочих мест ведется с целью получения информации об установленных в системе рабочих местах. Каждое рабочее место должно иметь:

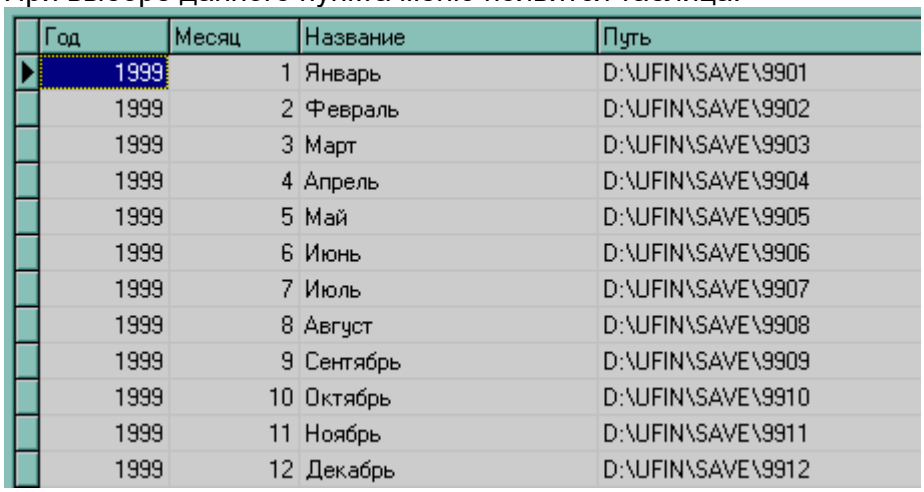
- код;
- ФИО ответственного лица;
- перечень задач, решаемых на данном рабочем месте.

Отчетные периоды

Под отчетным периодом понимается промежуток времени (обычно месяц или квартал), за который вводится движение по счетам и формируются, распечатываются все необходимые отчетные документы. По окончании отчетного периода производится переход на новый отчетный период с автоматическим переносом остатков по счетам, субсчетам, увеличением оборотов в главной книге и т.д.

Перед переходом на новый отчетный период рекомендуется текущий отчетный период сдать в архив.

При выборе данного пункта меню появится таблица.



Год	Месяц	Название	Путь
1999	1	Январь	D:\UFIN\SAVE\9901
1999	2	Февраль	D:\UFIN\SAVE\9902
1999	3	Март	D:\UFIN\SAVE\9903
1999	4	Апрель	D:\UFIN\SAVE\9904
1999	5	Май	D:\UFIN\SAVE\9905
1999	6	Июнь	D:\UFIN\SAVE\9906
1999	7	Июль	D:\UFIN\SAVE\9907
1999	8	Август	D:\UFIN\SAVE\9908
1999	9	Сентябрь	D:\UFIN\SAVE\9909
1999	10	Октябрь	D:\UFIN\SAVE\9910
1999	11	Ноябрь	D:\UFIN\SAVE\9911
1999	12	Декабрь	D:\UFIN\SAVE\9912

Внесите в нее номера года, месяца и названия отчетных периодов. В поле "Путь" укажите полный путь к директории, в которой будет содержаться архивная информация для данного отчетного периода. Перенос текущей информации в архив будет осуществляться при выполнении подпункта Сдать в архив пункта Регистрация ХО главного меню.

Реквизиты предприятия

Используя этот пункт, Вам необходимо настроить основные реквизиты предприятия: название, ИНН, банк и расчетный счет и т.д. Эта информация будет использоваться в дальнейшем при печати различных документов: платежные поручения, приходные и расходные кассовые ордера, требования, счета и т.д.

Параметры рабочего места

В этом пункте необходимо указать порядок формирования отчетов и проводок (с копейками или без), а также установить флаг разрешения внесения изменений в предыдущие периоды.

Прочие справочники

Прочие справочники - это перечень дополнительных вспомогательных справочников. В качестве таковых могут выступать, например:

- справочник коэффициентов для переоценки основных средств;
- номенклатурный справочник материалов;
- справочник шифров амортизации и т.д.

Для того, чтобы внести новый справочник в таблицу, необходимо:
1. Создать соответствующий btr - файл.(В комплект поставки входит программа создания и модификации btr файлов btrdesk.exe).
2. Создать новую карточку дополнительного справочника, в которую внести название справочника и способ его вызова.

Утилиты

Под словом "утилиты" понимается определенный набор сервисных процедур. Сервисные процедуры - это, например, программы печати платежных поручений, счетов, мемориальных ордеров, бланков налогов в налоговую инспекцию и т.д.

Для того, чтобы добавить новую утилиту в список утилит необходимо проделать следующее:
1. Вставить новую строку в таблицу.
2. В поле "Содержание" ввести название утилиты.
3. В поле "Вызов" - командную строку, активизирующую эту утилиту.

Каскад

Расположить окна стопкой

Мозаика

Расположить окна рядом, замостив площадь стола.

Закрыть все

Закрыть все открытые окна

Смотреть

Просмотр записи без возможности ее изменения

Вставить

Вставить новую пустую запись в базу данных

Копировать

Создать копию текущей записи и выполнить ее редактирование.

Изменить

Редактировать значения полей текущей записи

Удалить

Удалить текущую запись.

Найти

Найти запись с указанным значением активного (выбранного) поля.

Изменить фильтр

В некоторых режимах просмотра базы данных позволяет выбрать другое значение для фильтрации записей базы данных.

Запомнить

Запомнить значение текущего поля

Вспомнить

Вставить замомненное ранее значение в текущее поле

Очистить

Очистить (обнулить) значение теущего поля

Прибавить

Позволяет добавлять или вычитать к значению текущего числового поля

Выбор

Выбор значения текущего поля из списка.
Аналогично нажатию мышкой на кнопку с ... справа от поля.

Вызов редактора

Изменение структуры

Просмотреть отчет

Создать

Очистить текст в окне редактора. Создать новый текст.

Открыть...

Открыть существующий файл для просмотра и/или редактирования.

Обновить

Сформировать файл отчета заново и перечитать полученный файл с диска.
Полезно при одновременной работе нескольких пользователей.

Сохранить

Сохранить отредактированный текст в файле с тем же именем.

Сохранить как...

Сохранить отредактированный текст в файле с другим именем.

Печать

Напечатать получившийся текст с соответствии с настройками в Настройка печати

Закрыть

Закрыть окно редактора.

Правка

Стандартные возможности редактирования

Поиск

Поиск и замена текста

Восстановить

Считать предыдущее сохраненное состояние отчета. Полезно, если была сделана какая-то ошибка при редактировании отчета.

Сохранить

Сохранить изменения в отчете.

Настройка принтера

Выполнить настройку принтера (выбрать размеры бумаги и направление печати).

Печатать

Предварительный просмотр как печатается отчет.

Параметры страницы

Задать поля (отступ) от краев страницы.

Поля

Опередилить поля и форматы полей, содержащиеся в отчете.

Закрыть

Закрыть окно редактора отчета

Положение и размеры

После выполнения пункта меню "Положение и размеры" все элементы отчета можно сдвигать и изменять размеры с помощью мыши. Для передвижения нужно тянуть мышью за среднюю часть объекта. Для изменения размеров нужно тянуть мышью за середину правой стороны или за правый нижний угол объекта.

Свойства

Изменить свойства (текст, положение, цвет и т.п.) выбранного объекта.

Метка

В отчет в указанном месте вставить метку (небольшой текст, обычно одна строка). Текст может быть неизменяемым и изменяемым, связанным с полем из базы данных. При попадании при создании метки в ячейку размеры и положение метки совпадают с размерами и положением этой ячейки.

Линия

В отчет в указанном месте вставить линию или прямоугольник (возможно заполненный).

Текст

В отчет в указанном месте вставить текст (большой текст, несколько строк). Текст может быть неизменяемым и изменяемым, связанным с полем из базы данных.

При попадании при создании текста в ячейку размеры и положение текста совпадают с размерами и положением этой ячейки.

Рисунок

В отчет в указанном месте вставить рисунок, например, логотип фирмы.

Ячейка

В отчет в указанном месте вставить ячейку.
Ячейки существенно упрощают редактирование отчета, позволяя одним движением мыши изменять положение и размеры целой группы ячеек и расположенных на них объектах.

Блок

Отчет строится обычно из нескольких блоков:
Вверху страницы - этот блок печатается вверху на каждой странице отчета.
Внизу страницы - этот блок печатается внизу на каждой странице отчета.
Шапка - этот блок печатается один раз в начале отчета.
Итоги - этот блок печатается один раз в конце отчета.
Содержание - этот блок является повторяемым.

В отчете может содержаться несколько блоков одного типа, в этом случае они будут напечатаны последовательно друг за другом. Это может быть полезно для уменьшения пустых мест на бумаге, если блок не помещается на остатке пустого места на текущей странице, то он печатается на следующей.

Удалить

Удалить выбранный элемент отчета.

Вперед/Назад

Поменять местами элементы отчета. Расположить один элемент поверх или позади другого.

Сцепить/Расцепить

Ячейку можно сцепить с ее соседом слева или сверху. После сцепления при сдвиге соседки слева или сверху сдвигается и сама ячейка. Для сцепления с соседкой слева верхний левый угол ячейки должен находиться достаточно близко (менее 5 пикселов) от правого верхнего угла ее соседки слева. Аналогично для сцепления сверху.

Пункт меню "Расцепить" используется для уничтожения связей ячейки с ее соседками слева или сверху.

Выход

Выход из программы, завершение работы.

Отчет

Восстановить
Сохранить
Настройка принтера
Печатать
Параметры страницы
Поля
Закрыть

Правка

Положение и размеры
Свойства
Создать
Удалить

Свойства блока

Можно указать высоту блока и шрифт по умолчанию для меток и текста.

Свойства ячейки

Можно указать

- координаты верхнего левого угла внутри блока с точностью 0.1 мм
- размеры с точностью 0.1 мм
- какие границы ячейки должны быть прорисованы
- цвет, толщину и тип линии для рамки
- цвет и тип заполнения
- прижим к блоку (при изменении высоты блока объект не изменяется либо изменяется высота на такую же величину либо изменяется координата Y на такую же величину).
- размер в процентах от размеров соседней ячейки слева и справа.

Поля и форматы

Свойства рисунка

Можно указать

- координаты верхнего левого угла внутри блока с точностью 0.1 мм
- размеры с точностью 0.1 мм
- прижим к блоку (при изменении высоты блока объект не изменяется либо изменяется высота на такую же величину либо изменяется координата Y на такую же величину).
- загрузить картинку из файла

Свойства метки

Можно указать

- координаты верхнего левого угла внутри блока с точностью 0.1 мм
- размеры с точностью 0.1 мм
- тип, размер и цвет шрифта
- прижим к блоку (при изменении высоты блока объект не изменяется либо изменяется высота на такую же величину либо изменяется координата Y на такую же величину).
- прозрачность
- режим автоматического изменения размеров взависимости от выводимой строки
- выравнивание выводимого текста (влево, вправо, по центру) в пределах заданной ширины объекта
- имя и формат поля для связи с базой данных

Свойства текста

Можно указать

- координаты верхнего левого угла внутри блока с точностью 0.1 мм
- размеры с точностью 0.1 мм
- тип, размер и цвет шрифта
- прижим к блоку (при изменении высоты блока объект не изменяется либо изменяется высота на такую же величину либо изменяется координата Y на такую же величину).
- имя и формат поля для связи с базой данных

Параметры страницы

Можно указать отступы от краев страницы

Свойства линии

Можно указать

- вид линии (прямугольник, отрезок, эллипс)
- координаты верхнего левого угла внутри блока с точностью 0.1 мм
- размеры с точностью 0.1 мм
- цвет, толщину и тип линии для рамки
- цвет и тип заполнения
- прижим к блоку (при изменении высоты блока объект не изменяется либо изменяется высота на такую же величину либо изменяется координата Y на такую же величину).

# #

Синтаксис:
 # текст

Команда печати.

Примеры:
а) # Оборот 51 счета [ОБ("51")]
Будет напечатано: Оборот 51 счета 1200000.00.
б) # Сальдо 01 счета [СНД("01")]

Команда печати работает следующим образом:
а) все, что не заключено в квадратные скобки, выводится на печать без всяких изменений;
б) в квадратные скобки заключается произвольное выражение (в том числе, макросы, выражения и т.д.). Значение этого выражения сначала вычисляется, а затем выводится на печать.

Пример:

```
# Остаточная стоимость основных средств:
# [СНД("01") - СНК("02")]
```

$include

Синтаксис:
$include <имя файла>

Оператор включает в текст программы содержимое указанного файла. Этот файл может содержать определение часто используемых подпрограмм. По умолчанию расширение файла .BAS

Пример:
$include macros

abort_trn

Синтаксис:
abort_trn

Отмена транзакции.

Пример:

```
d = open("EXAMPLE")
getfirst(d)
begin_trn
'DbS_конец'.d = 'DbS_конец'.d * 2
update(d)
if find(d,1,2,200000)<>0 then
  rem искомое значение не найдено, "откатка" изменений
  abort_trn
else
  rem поиск успешный, запомнить изменения
  end_trn
end if
close(d)
end
```

В приведенной программе значение поля 'DbS_конец' таблицы EXAMPLE.BTR увеличивается в 2 раза, если найдено значение 200000 2-го ключа, состоящего из одного сегмента.

abs

abs(<число>)

Функция возвращает абсолютное значение (модуль) числа.

array

Синтаксис:
array(<размер>,<элементы>)

Создать массив указанного размера, записать в него заданные значения и вернуть его дескриптор. Элементы массива нумеруются от 0 до <размер>. Получить элемент массива можно с помощью операции []. Уничтожение массива происходит автоматически при исчезновении ссылок на массив. В 0-м элементе рекомендуется помещать текущую длину массива (первоначально равно количеству значений элементов при создании массива, т.е. если указан только размер, в 0-м элементе 0). Максимальную длину можно узнать с с помощью функуции size.

Пример:

```
a=array(10)
for i=1 to 10
  a[i]=0
next
rem Уничтожить массив
a=0

rem Двумерный массив
a=array(10)
for i=1 to 10
  a[i]=array(20)
next
rem Обращаемся к элементу (5,12)
a[5][12]=10
```

```
MonNames=array(12,"январь","февраль","март","апрель","май","июнь","июль","август","
сентябрь","октябрь","ноябрь","декабрь")
```

begin_trn

Синтаксис:
begin_trn

Начало транзакции.

Пример:

```
d = open("EXAMPLE")
getfirst(d)
begin_trn
'DbS_конец'.d = 'DbS_конец'.d * 2
update(d)
if find(d,1,2,200000)<>0 then
  rem искомое значение не найдено, "откатка" изменений
  abort_trn
else
  rem поиск успешный, запомнить изменения
  end_trn
end if
close(d)
end
```

В приведенной программе значение поля 'DbS_конец' таблицы EXAMPLE.BTR увеличивается в 2 раза, если найдено значение 200000 2-го ключа, состоящего из одного сегмента.

bof

bof(<дескриптор файла>)

Данная функция возвращает 1, если достигнуто начало файла, и 0, в противном случае.

Пример.

```
d1=open("EXAMPLE")
oborot = 0
getlast(d1)
while bof(d1)=0
  oborot = oborot + 'Сумма'.d1
  getprev(d1)
wend
close(d1)
end
```

chr

Синтаксис:
chr(<код>)

Функция возвращает строку из одного символа с указанным ASCII-кодом.

Пример:
```
  rem печать "
  print chr(34)
```

clearbuf

Синтаксис:
clearbuf(<дескриптор файла>)

Очистка буфера текущей записи BTR - файла (таблицы) с заданным дескриптором.

Пример.

```
d1 = open("EXAMPLE")
getfirst(d1)
clearbuf(d1)
insert(d1)
close(d1)
end
```

Приведенная программа вставляет пустую запись в базу данных.

close

Синтаксис:
close(<дескриптор файла>)

Закрытие BTR - файла (таблицы) с заданным дескриптором.

closeall

Синтаксис:
closeall(<значение>)

Используется для закрытия таблиц BTR путем уменьшения числа открытых таблиц до указанного значения.

current_tables

Текущие открыттые таблицы доступны через этот массив
current_tables[0] - количество уровней
current_tables[1] - таблица для текущего окна, чем больше номером, тем выше уровень

date_add

date_add(<дата>, изм, ед)
date_add(<дата>, изм)

Функция возвращает дату, увеличенную (уменьшенную) на *изм*. *ед* по умолчанию 0 (дни), 1 (месяц), 2 (год).

date_diff

date_diff(<дата1>, <дата2>[, <разница>=0])

Функция возвращает разницу между двумя датами в днях (по умолчанию) или в месяцах (если разница=1)

date_set

date_set(<дата>, <изм>)

Функция возвращает дату, соответствующую <изм> из вариантов -1 (начало месяца), 1 (конец месяца), -2 (начало года), 2 (конец года).

datetofld

datetofld(<дата>)

Данная функция преобразует значение <дата>, из формата даты (DD/MM/YY) в формат поля типа Date базы данных.

Пример: d = datetofld("01/02/97")
Результат: d = "19970201"

DBASIC

Программы пишутся на языке, очень напоминающем Бейсик, но в него встроены операторы и функции, позволяющие оперировать с базами данных.

dde_close

Синтаксис:
dde_close

Закрыть соединение с Excel или Word

dde_execute

Синтаксис:
dde_execute(<команда>)

Выполнить указанную команду (полная информация о возможных командах доступна в документации по MS Excel)

dde_open

Синтаксис:
dde_open(<приложение> [,<документ>])

Открыть соединение с Excel или Word

Пример:

```
dde_open("EXCEL")
dde_execute("OPEN("+chr(34)+"SAMPLE.XLS"+chr(34)+")")
dde_poke("R1C1","Отчет за 2003г.")
dde_close

page="Книга1"
rem выбрать страницу
dde_execute("[WORKBOOK.ACTIVATE("+chr(34)+page+chr(34)+")]")
rem проверить выбор
s=dde_request("Selection")
if page=substr(s,strlen(fname)+4,strlen(page)) then
  dde_close
  rem подключиться для чтения
  dde_open("EXCEL","[SAMPLE.XLS]"+page)
  rem извлечь значение
  s=dde_request("R1C1")
  rem убрать cr-lf
  s=substr(s,1,strlen(s)-2)
  rem отключиться
  dde_close
  dde_open("EXCEL")
end if
```

dde_poke

Синтаксис:
dde_poke(<поле>,<значение>)

Вывести значение в указанное поле (адрес ячейки таблицы формируется как R#C#)

dde_request

Синтаксис:
dde_request(<поле>)

Функция возвращает значение указанного поля (адрес ячейки таблицы формируется как R#C#)

delete

Синтаксис:
delete(<дескриптор файла>)

Удаление текущей записи таблицы с заданным дескриптором.

Пример.

```
d1 = open("EXAMPLE")
getfirst(d1)
delete(d1)
close(d1)
end
```

Приведенная программа удаляет первую запись базы данных.

divstr

Синтаксис:
divstr(<строка>,<ширина>,<номер>)

Разбить строку на подстроки не более указанной ширины и вернуть подстроку с указанном номером

end

Синтаксис:
end

Конец программы.

Пример.

```
print "Удаление базы данных EX.BTR"
system("delete EX.BTR")
print "База данных удалена, нажмите любую клавишу... "
pause
end
```

end_trn

Синтаксис:
end_trn

Завершение транзакции.

Пример:

```
d = open("EXAMPLE")
getfirst(d)
begin_trn
'DbS_конец'.d = 'DbS_конец'.d * 2
update(d)
if find(d,1,2,200000)<>0 then
  rem искомое значение не найдено, "откатка" изменений
  abort_trn
else
  rem поиск успешный, запомнить изменения
  end_trn
end if
close(d)
end
```

В приведенной программе значение поля 'DbS_конец' таблицы EXAMPLE.BTR увеличивается в 2 раза, если найдено значение 200000 2-го ключа, состоящего из одного сегмента.

eof

eof(<дескриптор файла>)

Данная функция возвращает 1, если достигнут конец файла, и 0, в противном случае.

Пример.

```
d1=open("EXAMPLE")
oborot = 0
getfirst(d1)
while eof(d1)=0
  oborot = oborot + 'Сумма'.d1
  getnext(d1)
wend
close(d1)
end
```

error

Синтаксис:
error(<сообщение>)

Функция завершает выполнение программы и выводит сообщение об ошибке

executeflag

Синтаксис:
executeflag(<режим>)

Функция устанавливает режим реагирования на ошибки исполнения. Режим является суммой флагов. Флаг 1 означает реакцию на неинициализированную (пустую) переменную. Флаг 2 означает реакцию на ошибку отсутствия базы данных. Функция возвращает старый режим исполнения.

Пример:

```
e=executeflag(0)
db=open("table")
if vartype(db)!=0 then
  rem таблица существует
else
  rem такой таблицы нет
end if
executeflag(e)
```

fieldname

findfield(<дескриптор файла>, <номер поля>)

Данная функция возвращает имя поля с указанным номером. Нумерация полей начинается с 0.

find*

find*(<дескриптор файла>, <список полей>,
<значение 1 поля>, [<значение 2 поля>...])

Если ключ совпадает с текущим, вместо списка полей указывается пустая строка. Для временных таблиц допускается использовать find и findge. Для DBF и DB поиск по ключу невозможен.

find(<дескриптор файла>, <количество сегментов>, <ключ поиска>,
<значение 1 сегмента поиска>, [<значение 2 сегмента поиска>...])

Поиск записи по номеру ключа, с заданными значениями сегментов ключа. Точный поиск по ключу.

findge(<дескриптор файла>, <количество сегментов>, <ключ поиска>,
<значение 1 сегмента поиска>, [<значение 2 сегмента поиска>...])

Поиск записи с равным или большим значением ключа.

findgt(<дескриптор файла>, <количество сегментов>, <ключ поиска>,
<значение 1 сегмента поиска>, [<значение 2 сегмента поиска>...])

Поиск записи с большим значением ключа.

findle(<дескриптор файла>, <количество сегментов>, <ключ поиска>,
<значение 1 сегмента поиска>, [<значение 2 сегмента поиска>...])

Поиск записи с равным или меньшим значением ключа.

findlt(<дескриптор файла>, <количество сегментов>, <ключ поиска>,
<значение 1 сегмента поиска>, [<значение 2 сегмента поиска>...])

Поиск записи с меньшим значением ключа.

find_l(<дескриптор файла>, <количество сегментов>, <ключ поиска>,
<значение 1 сегмента поиска>, [<значение 2 сегмента поиска>...])

Поиск записи по номеру ключа, с заданными значениями сегментов ключа с блокировкой найденной записи.

ВНИМАНИЕ: нумерация ключей во всех функциях поиска начинается с 1. Ключ 0 означает использование текущего ключа.

Данные функции возвращают значение:
0 - если запись, удовлетворяющая заданным условиям, найдена;
1 - если запись не найдена.

Пример:

```
d = open("EXAMPLE")
if find(d,1,2,200000)<>0 then
  rem искомое значение не найдено
else
  rem поиск успешный
end_if
close(d)
end
```

findfield

findfield(<дескриптор файла>, <имя поля>)

Данная функция возвращает 1, если поле с заданным именем существует в файле, и 0, в противном случае.

fldtodate

fldtodate(<дата>)

Данная функция преобразует значение <дата>, из формата поля (19970201) в формат даты 01/02/97.

Пример: d = fldtodate("19970201")
Результат: d = "01/02/97"

fldtofulldate

fldtofulldate(<дата>)

Данная функция преобразует значение <дата>, из формата поля (19970201) в формат даты 01/02/1997.

Пример: d = fldtodate("19970201")
Результат: d = "01/02/1997"

fmod

fmod(<делимое>,<делитель>)

Функция возвращает остаток от деления при делении чисел нацело.

for

Синтаксис:
for <начальное выражение> to <конечное выражение>
** <операторы>**
next

Оператор цикла. Тело цикла выполняется до тех пор, пока начальное выражение, увеличиваемое при каждом выполнении цикла на единицу, не станет больше конечного выражения.

Пример:

```
d = open("EXAMPLE")
getfirst(d)
for i=1 to size(d)
  'DbS_конец'.d = 'DbS_конец'.d + 100
  update(d)
  getnext(d)
next
close(d)
end
```

fromeof

Синтаксис:
fromeof

Функция позволяет проверить, окончился ли текущий входной файл

Пример:

```
fromfile("TEXT.TXT")
while fromeof=0
  input s
  print s
wend
fromfile()
```

fromfile

Синтаксис:
fromfile([<имя файла>][, <кодировка>=0])

Подпрограмма позволяет выполнять ввод из указанного файла.
Если имя файла не указано, то восстанавливается режим интерактивного ввода.
По умолчанию используется кодировка DOS (866), для кодировки Windows (1251) вторым аргументом указать 1, для UTF8 - 4.

Пример:

```
fromfile("TEXT.TXT")
while fromeof=0
  input s
  print s
wend
fromfile()
```

function

Синтаксис:
function <имя функции> [(<имя1> [, <имя2>]...)]
** <операторы>**
** result=<выражение>**
end function

Определение пользовательской функции. Результат, возвращаемый функцией должен быть записан в переменную result. Для досрочного выхода можно использовать оператор return. Подпрограмма может иметь переменное количество аргументов, для этого в заголовке нужно указать символ * вместо списка параметров подпрограммы. Фактические аргументы функции будут помещены в массив arg. Элемент массива arg[0] содержит количество аргументов. Если аргументы разделялись символом ; или какой-либо аргумент пропущен (например, f(1,,3) или f(1;3)), то массиве будут содержаться пустые значения (для обработки пустых значений установите флаги исполнения).

Определения подпрограмм не могут быть вложенными. Функции не могут быть рекурсивными.

Пример:
```
  function cube(x)
    result=x*x*x
  end function
  print "4^3=";cube(4)
```

getbuffer

getbuffer(<дескриптор файла>)

Функция возвращает содержимое текущей записи в виде одной строки. Удобно для копирования записей.

getfirst

Синтаксис:
getfirst(<дескриптор файла>)

Переход к первой по текущему индексу записи BTR - файла (таблицы) с заданным дескриптором.

getfirst_l(<дескриптор файла>)

Переход к первой по текущему индексу записи BTR - файла (таблицы) с заданным дескриптором с блокировкой этой записи.

Пример.

```
d1 = open("EXAM_1")
d2 = open("EXAM_2 ")
getfirst(d1)
getfirst_l(d2)
for i=1 to 10
  'DbS_конец'.d2 = 'DbS_конец'.d1
  update(d2)
  getnext(d1)
  getnext_l(d2)
next
close(d1)
close(d2)
end
```

Приведенная программа осуществляет копирование значений поля 'DbS_конец' 10-ти первых записей базы данных EXAM_1.BTR в соответствующее поле 10-ти первых записей базы данных EXAM_2.BTR. Причем, записи дополняемой базы данных берутся с блокировкой.

getkey

getkey(<дескриптор файла>)
getkey(<дескриптор файла>, <номер индекса>)

Данная функция возвращает текущий индекс в таблице или индекс с указанным номером (нумерация с 0) .

getlast

Синтаксис:
getlast(<дескриптор файла>)

Переход к последней по текущему индексу записи BTR - файла (таблицы) с заданным дескриптором.

getlast_l(<дескриптор файла>)

Переход к последней по текущему индексу записи BTR - файла (таблицы) с заданным дескриптором с блокировкой этой записи.

Пример.

```
d1 = open("EXAM_1")
d2 = open("EXAM_2 ")
getlast(d1)
getlast_l(d2)
for i=1 to 10
  'DbS_конец'.d2 = 'DbS_конец'.d1
  update(d2)
  getprev(d1)
  getprev_l(d2)
next
close(d1)
close(d2)
end
```

Приведенная программа осуществляет копирование значений поля 'DbS_конец' 10-ти последних записей базы данных EXAM_1.BTR в соответствующее поле 10-ти последних записей базы данных EXAM_2.BTR. Причем, записи дополняемой базы данных берутся с блокировкой.

getnext

Синтаксис:
getnext(<дескриптор файла>)

Переход к следующей по текущему индексу записи BTR - файла (таблицы) с заданным дескриптором.

getnext_l(<дескриптор файла>)

Переход к следующей по текущему индексу записи BTR - файла (таблицы) с заданным дескриптором с блокировкой этой записи.

Пример.

```
d1 = open("EXAM_1")
d2 = open("EXAM_2 ")
getfirst(d1)
getfirst_l(d2)
for i=1 to 10
  'DbS_конец'.d2 = 'DbS_конец'.d1
  update(d2)
  getnext(d1)
  getnext_l(d2)
next
close(d1)
close(d2)
end
```

Приведенная программа осуществляет копирование значений поля 'DbS_конец' 10-ти первых записей базы данных EXAM_1.BTR в соответствующее поле 10-ти первых записей базы данных EXAM_2.BTR. Причем, записи дополняемой базы данных берутся с блокировкой.

getpath

Синтаксис:
getpath

Функция, возвращающая текущую базу данных в формате "UFINyymm" или "UFINMAIN"

getprev

Синтаксис:
getprev(<дескриптор файла>)

Переход к предыдущей по текущему индексу записи BTR - файла (таблицы) с заданным дескриптором.

getprev_l(<дескриптор файла>)

Переход к предыдущей по текущему индексу записи BTR - файла (таблицы) с заданным дескриптором с блокировкой этой записи.

Пример.

```
d1 = open("EXAM_1")
d2 = open("EXAM_2 ")
getlast(d1)
getlast_l(d2)
for i=1 to 10
  'DbS_конец'.d2 = 'DbS_конец'.d1
  update(d2)
  getprev(d1)
  getprev_l(d2)
next
close(d1)
close(d2)
end
```

Приведенная программа осуществляет копирование значений поля 'DbS_конец' 10-ти последних записей базы данных EXAM_1.BTR в соответствующее поле 10-ти последних записей базы данных EXAM_2.BTR. Причем, записи дополняемой базы данных берутся с блокировкой.

gettime

Синтаксис:
gettime
или
gettime(<формат>)

Получить текущее время на компьютере и дату. По умолчанию формат "dd/mm/yyyy hh:ii:ss"

gosub

Синтаксис:
gosub <метка>

Оператор перехода к подпрограмме. Передает управление подпрограмме, помеченной <метка>.

Пример:

```
d = open("EXAMPLE")
getfirst(d)
gosub 100
close(d)
end
100
for i=1 to 10
  'DbS_конец'.d = 'DbS_конец'.d + 100
  update(d)
  getnext(d)
next
return
```

goto

Синтаксис:
goto <метка>

Оператор безусловного перехода. Передает управление на оператор, помеченный <метка>.

Пример:

```
        d = open("EXAMPLE")
        getfirst(d)
        if 'Сумма'.d > 1000000 then goto 100
        print "Сумма остается прежней"
        goto 200
  100   'Сумма'.d = 1000000
        update(d)
  200   close(d)
        end
```

if-then

Синтаксис:
if <выражение> then <оператор-then>
или
if <выражение> then
** <операторы-then>**
[else if <выражение> then
** <операторы-then>]**
[else
** <операторы-else>]**
end if

Условный оператор. Если значение <выражения> истинно, то <операторы-then> выполняется, в противном случае, выполняется <операторы-else>. Часть else может быть пропущена.

Пример 1:

```
        d = open("EXAMPLE")
        getfirst(d)
        if 'Сумма'.d > 1000000 then goto 100
        print "Сумма остается прежней"
        goto 200
  100   'Сумма'.d = 1000000
        update(d)
  200   close(d)
        end
```

Пример 2:

```
        d = open("EXAMPLE")
        getfirst(d)
        if 'Сумма'.d <= 1000000 then
          print "Сумма остается прежней"
        else
          'Сумма'.d = 1000000
        end if
        update(d)
        close(d)
        end
```

input

Синтаксис:
input [<выражение>] <переменная>

Ввод с клавиатуры значения переменной.
<выражение> - сообщение выдаваемое при вводе.

Пример:

```
input "Введите строку:" string
input number
print "Строка: ",string;"Число: ",number
end
```

Результат работы программы:

```
Введите строку:ПРИМЕР СТРОКИ
?1234
Строка: ПРИМЕР СТРОКИ  Число: 1234
```

insert

Синтаксис:
insert(<дескриптор файла>)

Вставка записи в BTR - файл (таблицы) с заданным дескриптором и текущим содержимым буфера данных.

Пример.

```
d1 = open("EXAMPLE")
getfirst(d1)
clearbuf(d1)
insert(d1)
close(d1)
end
```

Приведенная программа вставляет пустую запись в базу данных.

keyarray

Синтаксис:

keyarray

Создать ассоциативный массив и вернуть его дескриптор. Получить элемент аасоциативного массива можно с помощью операции [], в качестве индекса указывается непустая строка. При обращении всегда создается новый элемент с неопределенным значением. Для проверки наличия элемента нужно использовать функцию keyfind.

Пример:

```
a=keyarray
a["first"]=1
a["second"]=2
print a["first"]+a["second"]
```

keyfind

Синтаксис:
keyfind(<ассоц.массив>,<ключ>)

Функция проверяет наличие ключа в ассоциативном массиве и возращает 0, если элемента с таким ключом нет, или 1, если есть.

Пример:

```
if keyfind(a,"third")=0 then
  print "Элемент с ключом third не существует"
else
  print "Элемент с ключом third существует"
end if
```

keyfirst,keynext

Синтаксис:
keyfirst(<ассоц.массив>)
keynext(<ассоц.массив>,<ключ>)

Функция keyfirst возвращает ключ первого элемента в ассоциативном массиве, а функция keynext - ключ следующего элемента за указанным. Если следующего элемента нет, то функция возвращает пустую строку.

Пример:

```
s=keyfirst(a)
while s<>""
   print s,a[s]
   s=keynext(a,s)
wend
```

keylast,keyprev

Синтаксис:
keylast(<ассоц.массив>)
keyprev(<ассоц.массив>,<ключ>)

Функция keylast возвращает ключ последнего элемента в ассоциативном массиве, а функция keyprev - ключ предыдущего элемента за указанным. Если предыдущего элемента нет, то функция возвращает пустую строку.

Пример:

```
s=keylast(a)
while s<>""
   print s,a[s]
   s=keyprev(a,s)
wend
```

kkm_command

kkm_command(<команда>)

Выполнение команды ККМ: "operator_login", "open_shift", "report", "open_receipt", "registration", "payment", "close_receipt", "continue_print", "check_document_closed", "print_text", "payment", "cancel_receipt", "fn_query_data", "query_data","receipt_total", "util_form_tlv"

Пример:

```
kkm_set_str(65536, "Печать строки")
kkm_command("print_text")
```

kkm_get_...

kkm_get_str(<номер_параметра>)

Чтение строкового параметра

kkm_get_int(<номер_параметра>)

Чтение целого параметра

kkm_get_bool(<номер_параметра>)

Чтение булевого параметра

kkm_get_double(<номер_параметра>)

Чтение числового параметра

kkm_get_datetime(<номер_параметра>)

Чтение параметра в формате yyyymmdd hh:nn:ss

kkm_get_bytearray(<номер_параметра>)

Чтение байтового параметра

kkm_set_...

kkm_set_str(<номер_параметра>, <строка>)

Установка строкового параметра

kkm_set_int(<номер_параметра>, <число>)

Установка целого параметра

kkm_set_bool(<номер_параметра>, <0 или 1>)

Установка булевого параметра

kkm_set_double(<номер_параметра>, <число>)

Установка числового параметра

kkm_set_datetime(<номер_параметра>, <строка>)

Установка параметра в формате yyyymmdd hh:nn:ss или yyyymmdd

kkm_set_bytearray(<номер_параметра>, <строка>)

Установка байтового параметра

kkm_setup, kkm_open, kkm_close

kkm_setup(<номер модели ККМ>, <адрес_порт>, <пароль>, <проверка>)

Установка параметров соединения с ККМ.

kkm_open

Соединение с ККМ.

kkm_close

Отключение от ККМ.

Пример:

```
kkm_setup(67,"COM3","",1)
kkm_open
...
kkm_close
```

local

Синтаксис:
local <имя1>[, <имя2>]...

Оператор объявляет указанные переменные локальными в данной подпрограмме. По умолчанию все переменные являются глобальными.

lower

lower(<строка>)

Данная функция возвращает строку, в которой все буквы переведены в нижний регистр.

Пример:

```
print lower("Строка")
end
```

Результат:

строка

max

max(<выражение1>,<выражение2>)

Данная функция возвращает большее из двух значений выражений.

Пример.

```
d1 = open("EXAMPLE")
getfirst(d1)
num1 = min(100000,'DbS_конец'.d1)
num2 = max('DbS_начало'.d1,num1)
'DbS_конец'.d1 = num1
'DbS_начало'.d1 = num2
update(d1)
close(d1)
end
```

min

min(<выражение1>,<выражение2>)

Данная функция возвращает меньшее из двух значений выражений.

Пример.

```
d1 = open("EXAMPLE")
getfirst(d1)
num1 = min(100000,'DbS_конец'.d1)
num2 = max('DbS_начало'.d1,num1)
'DbS_конец'.d1 = num1
'DbS_начало'.d1 = num2
update(d1)
close(d1)
end
```

move_...

move_spoop(<дескриптор файла>)

Перенос текущей проводки в отложеннные

move_spprov(<дескриптор файла>)

Перенос из отложенных в справочник проводок

npro

spro(<число>)

Данная функция возвращает "Число прописью".

Пример: sp = spro(12.34)
Получится sp = "Двенадцать целых тридцать четыре сотых"

num

num(<строка>)

Данная функция возвращает числовое представление заданной строки.

Пример:

```
print trim("36.6")+100
end
```

Результат:

136.6

numfields

numfields(<дескриптор файла>)

Данная функция возвращает количество полей в базе данных.

numkeys

numkeys(<дескриптор файла>)

Данная функция возвращает количество индексов в базе данных.

numtables

Синтаксис:
numtables

Узнать число открытых таблиц BTR

Пример:

```
n=numtables
db=open("table")
closeall(n)
```

open

open(<имя базы данных>[, <режим>])

Данная функция открывает BTR или DBF таблицу и возвращает дескриптор.
Режим 0 позволяет открыть файл для чтения и записи, режим -2 - только для чтения, а режим -4 - для эксклюзивного использования.
По умолчанию используется режим 0.
Для обращения к полям можно использовать синтаксис d["имя поля"], d[номер_поля] или 'имя поля'.d
Последний вариант работает только для btr.

Для создания временной таблицы в памяти вместо имени указывается список полей через ;
После имени поля указывается символ : и тип поля: C (строка), N (число) и D (дата). Для C обязательно указывается размер, для N - количество цифр и необязательная точность. Временная таблица может иметь один ключ сортировки. Для вставки записи в упорядоченную таблицу необходимо выполнить поиск места для вставки с помощью функции findge.

Пример.

```
d1 = open("EXAMPLE")
getfirst(d1)
'DbS_конец'.d1 = 'DbS_начало'.d1
d1["DbS_начало"] = 0
update(d1)
close(d1)
end
```

pause

Синтаксис:
pause

Ожидание нажатия клавиши.

Пример.

```
print "Удаление базы данных EX.BTR"
system("delete EX.BTR")
print "База данных удалена, нажмите любую клавишу... "
pause
end
```

print

Синтаксис:
print <выражение> [;] [,] [<выражение>]...

Вывод данных на монитор.
; - начиная с очередной позиции табуляции;
, - начиная с текущей позиции.

Пример:

```
d = open("EXAMPLE")
getfirst(d)
string = " печати"
print "Пример оператора",string;;'Дата'.d
close(d)
end
```

Результат работы программы:

```
Пример оператора печати        10/11/95
```

printque

Синтаксис:
printque(<имя файла>)

Печать QUE-отчета в текущий файл.

putkey

putkey(<дескриптор файла>,<номер ключа>)
putkey(<дескриптор файла>,<список полей>)
putkey(<дескриптор файла>,<номер ключа>,<массив выделения по ключу>)
putkey(<дескриптор файла>,<список полей>,<массив выделения по ключу>)

Данная функция устанавливает активный ключ для указанного файла. Ключи нумеруются от 0. По умолчанию в качестве активного устанавливается нулевой ключ. Вызов с <массив выделения по ключу> может пригодиться при работе с w_select.

```
rem Проход по одной дате
rng=array(1,"Дата=20100120")
putkey(db,"Дата;Документ",rng)
getfirst(db)
while eof(db)=0
  ...
  getnext(db)
wend
rem Проход по диапазону дат
rng=array(2,"Дата>20100120","Дата<20100130")
putkey(db,"Дата;Документ",rng)
getfirst(db)
while eof(db)=0
  ...
  getnext(db)
wend
```

redirect

Синтаксис:
redirect([<имя файла>][,<режим>=0])

Подпрограмма позволяет перенаправить дальнейший вывод в указанный файл.
Если имя файла не указано, то восстанавливается вывод в стандартный файл, содержимое которого будет показано на экране после завершения выполнения программы.
Режим является суммой флагов, кодировка: DOS (866) - флаг 0, Windows (1251) - флаг 1, UTF8 - флаг 4, BOM для UTF8 - флаг 8, способ открытия: перезапись - флаг 0, добавление - флаг 2.

Пример:

```
redirect("REPORT.TXT")
print "Отчет за "+data
redirect()
```

regex_find_all

Синтаксис:
regex_find_all(<рег.выр.>,<строка>)

Найти все подстроки, соответствующие регулярному выражению. Возвращается массив подстрок. Если ни одной подстроки не найдено, то в 0-м элементе массива будет число 0.
Пример:

```
rem обработка файлов ;
a=regex_find_all("(([^;\\]|\\.)+$|([^;\\]|\\.)*;)",";a\;aa;d;;b\\bb")
print a[0] rem количество элементов
for i=1 to a[0]
   print regex_replace("\\(.)","$1",regex_replace(";$","",a[i])) rem убрать символы \ и завершающий ;
next
```

regex_flags

Синтаксис:
regex_flags(<строка>)

Установить флаги для обработки регулярных выражений в соответствии со строкой. Флаг i - игнорировать регистр, флаг s - метасимвол **.** (точка) включает и переход на новую строку, флаг m - строка содержит символы перехода новую строку, на которые должны реагировать метасиволы ^ и $. Включаемые флаги нужно уазать в начале строки, а выключаемые -после символа - (минус).

Пример:

```
regex_flags("i-ms")
```

regex_match

Синтаксис:
regex_match(<рег.выр.>,<строка>)

Проверить наличие в строке подстроки, соответствующей регулярному выражению. Функция возвращает 0, если такой подстроки нет, и 1, если она есть.

Пример:

```
if regex_match("^[0-9]+/[0-9]+/[0-9]+$",s)=1 then rem строка является датой
```

regex_replace

Синтаксис:
regex_replace(<рег.выр.>,<замена>,<строка>)

Функция производит замену подстрок, соответствующих регулярному выражению, на указанную строку, в которой можно ссылаться на части подстроки, соответствующие подвыражениям в (). $0 - вся подстрока, $1 - первая скобка и т.д.

Пример:

```
print regex_replace("([0-9]{4})([0-9]{2})([0-9]{2})","$3/$2/$1","20110201")   rem
01/02/2011
```

regex_replacef

Синтаксис:
regex_replace(<рег.выр.>,<имя функции>,<строка>)

Функция производит замену подстрок, соответствующих регулярному выражению, на строку, возвращаемую заданной функцией.
Функции передается массив, в котором содержатся подстрока и ее части, соответствующие подвыражениям в ().
i-й скобке в регулярном выражении соответствует (i+1)-й элемент массива. 1-й элемент - всей найденной подстроке. Функция должна вернуть строку.

Пример:

```
  function chgdat(m)
    result=m[2]+"/"+m[3]+"/"+str(num(m[4])+1)
  end function

  print regex_replacef("([0-9]+)/([0-9]+)/([0-9]+)","chgdat","Сдвиг даты 01/02/2011 на 1 год")
```

regex_split

Синтаксис:
regex_split(<рег.выр.>,<строка>)

Разделить строку на подстроки. Функция возвращает массив подстрок.

Пример:

```
a=regex_split("[ ,;.]+","Разделить текст на слова.")
print a[0] rem количество слов
for i=1 to a[0]
  print a[i] rem слова
next
```

regex_submatch

Синтаксис:
regex_submatch(<рег.выр.>,<строка>)

Найти первую подстроку, соответствующую регулярному выражению, и выделить ее части, соответствующие подвыражениям. Функция возвращает массив. i-й скобке в регулярном выражении соответствует (i+1)-й элемент массива. 1-й элемент - всей найденной подстроке.
Если подстрока не найдена, то в 0-м элементе массива будет число 0.

Пример:

```
rem обработка файлов ;
a=regex_submatch("([0-9]+)/([0-9]+)/([0-9]+)","Здесь есть дата 01/05/2007")
print a[0] rem количество элементов = 4
print a[1] rem вся дата 01/05/2007
print a[2] rem день 01
print a[4] rem год 2007
```

rem

Синтаксис:
rem [<текст до конца строки>]
' [<текст до конца строки>]

/' [<многострочный текст>] '/

Комментарии в программе. После ' должен быть пробел или переход на новую строку

Пример:

```
d = open("EXAMPLE") ' открыть файл
getfirst(d)
begin_trn
'DbS_конец'.d = 'DbS_конец'.d * 2
update(d)
if find(d,1,2,200000 /' или 100000 '/ )<>0 then
  rem искомое значение не найдено, "откатка" изменений
  abort_trn
else
  rem поиск успешный, запомнить изменения
  end_trn
end if
close(d)
end
```

resultfield

Синтаксис:
resultfield(<имя функции>,<таблица>,<имя поля>)

Используется для возрата поля таблицы из функции для последующих действий (обычно функция может вернуть только значение).

Пример:

```
function ПАРАМ(n)
local i
   db=open("param")
   getfirst(db)
   for i=1 to n-1
      getnext(db)
   next
   resultfield("ПАРАМ",db,"Значение")
end function
```

return

Синтаксис:
return

Возврат из подпрограммы, управление которой было передано оператором перехода к подпрограмме.

Пример:

```
d = open("EXAMPLE")
getfirst(d)
gosub 100
close(d)
end
100
for i=1 to 10
  'DbS_конец'.d = 'DbS_конец'.d + 100
  update(d)
  getnext(d)
next
return
```

round, floor, ceil

Синтаксис:
round(<число>,<кол-во знаков>)
floor(<число>,<кол-во знаков>)
ceil(<число>,<кол-во знаков>)

Функции предназначены для округления к ближайшему числу, вниз или вверх с указанным количеством знаков после запятой (по умолчанию <кол-во знаков> равно 0, т.е. округление до целого).

rpt_close

Синтаксис:
rpt_close

Закрытие файла отчета.

Пример:

```
d = open("EXAMP")
rpt_open("TEST.REP")
  sum=0
  while not eof(d)
    rpt_write("Сумма",'Сумма'.d)
    rpt_line
    sum = sum+ 'Сумма'.d
    getnext(d)
  wend
  rpt_write("Итого",sum)
rpt_view
rpt_close
close(d)
end
```

rpt_line

Синтаксис:
rpt_line

Вывести сформированный повторяющийся блок отчета.

Пример:

```
d = open("EXAMP")
rpt_open("TEST.REP")
  sum=0
  while not eof(d)
    rpt_write("Сумма",'Сумма'.d)
    rpt_line
    sum = sum+ 'Сумма'.d
    getnext(d)
  wend
  rpt_write("Итого",sum)
rpt_view
rpt_close
close(d)
end
```

rpt_open

Синтаксис:
rpt_open(<имя файла>[, <иниц_подпр>, <начало_стр>, <конец_стр>])

Открытие файла отчета. Печать происходит в два прохода. В первый проход готовится результат для первоначального просмотра, для печати на принтер программа повторно выполняется с команды rpt_open до rpt_view.

Пример 1:

```
d = open("EXAMP")
rpt_open("TEST.REP")
  sum=0
  while not eof(d)
     rpt_write("Сумма",'Сумма'.d)
     rpt_line
     sum = sum+ 'Сумма'.d
     getnext(d)
  wend
  rpt_write("Итого",sum)
rpt_view
rpt_close
close(d)
end
```

Пример 2:

```
sub init_rep
  npage=1
end sub
sub start_page
   rpt_write("N_page",str(npage,3,0))
end sub
sub end_page
   npage=npage+1
end sub
rpt_open("TEST.REP","init_rep","start_page","end_page")
...
rpt_close
```

rpt_view

Синтаксис:
rpt_view

Показ сформированного отчета.

Пример:

```
d = open("EXAMP")
rpt_open("TEST.REP")
  sum=0
  while not eof(d)
    rpt_write("Сумма",'Сумма'.d)
    rpt_line
    sum = sum+ 'Сумма'.d
    getnext(d)
  wend
  rpt_write("Итого",sum)
rpt_view
rpt_close
close(d)
end
```

rpt_write

Синтаксис:
rpt_write(<имя поля>, <выражение>)

Вывод значения в указанное поле отчета. Формат поля определяется в отчете.

Пример:

```
d = open("EXAMP")
rpt_open("TEST.REP")
  sum=0
  while not eof(d)
     rpt_write("Сумма",'Сумма'.d)
     rpt_line
     sum = sum+ 'Сумма'.d
     getnext(d)
  wend
  rpt_write("Итого",sum)
rpt_view
rpt_close
close(d)
end
```

selectfiles

Синтаксис:
selectfiles(<фильтр>)

Выбрать файлы для дальнейших действий. Функция возвращает массив, в котором под индексом 0 содержится количество выбранных файлов, а затем следуют элементы с именами выбранных файлов.

Пример:

```
files=selectfiles("Файлы с платежками|*.XLS")
for i=1 to files[0]
  print files[i]
next
```

setbuffer

setbuffer(<дескриптор файла>, <буфер>)

Подпрограмма устанавливает новое содержимое текущей записи. Удобно для копирования записей.

setfilter

setfilter(<дескриптор файла>.<фильтр>)

Установить фильтр для записей в таблице. Имена полей записываются в [], строковые константы в ' '.
В качестве фильтра можно указать имя функции, возвращающей 1 для видимых записей и 0 для игнорируемых.
Для временных таблиц допускается фильтрация только через функцию.

setpath

Синтаксис:
setpath(<имябд>)

Задать имя базы данных, используемой в макросах

size

size(<дескриптор файла>)

Данная функция возвращает количество записей в BTR - файле (базе данных) с заданным дескриптором.
Для массивов возвращается номер последего элемента в массиве (размер массива).

Пример.

```
d1=open("EXAMPLE")
oborot = 0
for i = 1 to size(d1)
  if i = 1 then getfirst(d1)
  if i>1 then getnext(d1)
  oborot = oborot + 'Сумма'.d1
next
close(d1)
end
```

sleep

Синтаксис:
sleep(<время в мс>)

Задержка на указанное время.

spro

spro(<число>)

Данная функция возвращает "Сумму прописью".

Пример: sp = spro(12.34)
Получится sp = "Двенадцать рублей 34 копейки."

str

str(<выражение>, <длина строки>, <число десятичных позиций>)

Данная функция возвращает представление числового <выражения> в виде строки длиной <длина строки> символов и с заданным числом десятичных позиций.
По умолчанию длина строки равна 1, число позиций 0.
Если длина строки отрицательна, то вместо пробелов в начало строки добавляются 0.

Пример:

```
print str(12.5,10,3)
print str(12.9)
print str(7,-3)
end
```

Результат:

```
12.500
13
007
```

strlen

strlen(<строка>)

Данная функция возвращает длину указанной строки.

Пример:

```
print strlen("СТРОКА")
end
```

Результат:

6

strpos

Синтаксис:
strpos(<подстрока>, <строка>)

Функция возвращает индекс вхождения подстроки в строке. Если подстроки, возвращается 0.

sub

Синтаксис:
sub <имя подпрограммы> [(<имя1> [, <имя2>]...)]
** <операторы>**
end sub

Определение пользовательской подпрограммы. Для досрочного выхода можно использовать оператор return. Подпрограмма может иметь переменное количество аргументов, для этого в заголовке нужно указать символ * вместо списка параметров подпрограммы. Фактические аргументы будут помещены в массив arg. Элемент массива arg[0] содержит количество аргументов. Если аргументы разделялись символом ; или какой-либо аргумент пропущен (например, f(1,,3) или f(1;3)), то массиве будут содержаться пустые значения (для обработки пустых значений установите флаги исполнения).

Определения подпрограмм не могут быть вложенными. Подпрограммы не могут быть рекурсивными.

Пример:
```
  sub clear(db)
    local i
    getfirst(db)
    for i=1 to size(db)
      delete(db)
    next
  end sub
```

substr

substr(<строка>, <начальная позиция>, <длина подстроки>)

Данная функция возвращает подстроку <строки> заданной длины, начиная с заданной позиции.

Пример:

```
print substr("СТРОКА",4,2)
end
```

Результат:

```
ОК
```

system

Синтаксис:
system(<строка>)

Выполнение <строки> как команды MS-DOS.

Пример.

```
print "Удаление таблицы EX.BTR"
system("delete EX.BTR")
print "Таблица удалена, нажмите любую клавишу... "
pause
end
```

trim

trim(<строка>)

Данная функция возвращает строку с отсеченными концевыми пробелами.

Пример:

```
print trim("СТРОКА          ")+"!"
end
```

Результат:

СТРОКА!

trimfull

trimfull(<строка>)

Данная функция возвращает строку с отсеченными начальными и конечными пробелами.

Пример:

```
print trimfull("     СТРОКА    ")+"!"
end
```

Результат:

СТРОКА!

triml

triml(<строка>)

Данная функция возвращает строку с отсеченными начальными пробелами.

Пример:

```
print triml("     СТРОКА ")+"!"
end
```

Результат:

```
СТРОКА !
```

unlock

Синтаксис:
unlock(<дескриптор файла>)

Разблокировка текущей записи базы данных с указанным дескриптором.

Пример.

```
getfirst_l(d1)
'DbS_конец'.d = 'DbS_конец'.d * 2
update(d1)
unlock(d1)
```

В приведенной программе блокируется 1-я запись таблицы, увеличивается ее поле 'DbS_конец' в 2 раза, обновляется эта запись на диске, а затем блокировка снимается.

update

Синтаксис:
update(<дескриптор файла>)

Запись на диск текущего содержимого буфера данных (обновление текущей записи).

Пример:

```
d = open("EXAMPLE")
getfirst(d)
'DbS_конец'.d = 'DbS_конец'.d + 100
update(d)
close(d)
end
```

upper

upper(<строка>)

Данная функция возвращает строку, в которой все буквы переведены в верхний регистр.

Пример:

```
print upper("Строка")
end
```

Результат:

СТРОКА

vartype

Синтаксис:
vartype(<значение>)

Функция определяет тип указанного значения и возвращает
0 - если значение является пустым (только если разрешено обращение к неинициализированным переменным)
1 - если значение является числом
2 - если значение является строкой
3 - если значение является таблицей BTR
4 - если значение является массивом
5 - если значение является таблицей DBF
6 - если значение является ассоциативным массивом

Пример:

```
e=executeflag(0)
db=open("table")
if vartype(db)!=0 then
  rem таблица существует
else
  rem такой таблицы нет
end if
executeflag(e)
```

w_button

Синтаксис:
w_button(<номер>,<x>,<y>,<w>,<h>,<текст>)

Создать в окне кнопку в указанных координатах

w_close

Синтаксис:
w_close

Закрыть окно

w_edit

Синтаксис:
w_edit(<номер>,<x>,<y>,<w>,<h>,<текст>[,<режим>,<маска>])

Создать в окне поле значения в указанных координатах. Если режим не 0, то создается нередактируемое поле ввода
Последний аргумент содержит маску для редактирования
Например, маска для даты "99/99/9999"
В случае маски "+-" создается чекбокс. Текст должен быть "+" ИЛИ "-".

w_gettext

Синтаксис:
w_gettext(<номер>)

Функция возвращает текст, связанный с элементом окна под указанным номером. Таким образом можно получить введенные значения из полей ввода.

w_label

Синтаксис:
w_label(<номер>,<x>,<y>,<w>,<h>,<текст>)

Создать в окне метку в указанных координатах

w_message

Синтаксис:
w_message(<заголовок>,<сообщение>,<кнопки>)

Функция выводит сообщение и ожидает нажатие кнопки. Возвращает номер нажатой кнопки.

Комбинации кнопок и значков (3-й аргумент)

0	OK
1	OK, Cancel
2	Abort, Retry, Ignore
3	Yes, No, Cancel.
4	Yes, No
5	Retry, Cancel
16	Знак STOP
32	Знак ?
48	Знак !
64	Знак i

Номер нажатой кнопки

1	OK
2	Cancel
3	Abort
4	Retry
5	Ignore
6	Yes
7	No

w_open

Синтаксис:
w_open(<x>,<y>,<w>,<h>,<заголовок>)

Создать окно

```
w_open(300,250,300,220,"Введите старую и новую фамилии МОЛ")
w_label(1,10,5,100,40,"ФИО:")
w_edit(4,10,25,100,24,old,0)
w_label(5,10,55,100,40,"Изменить на ФИО:")
w_edit(6,10,75,100,24,new,0)
w_button(2,10,160,50,25,"Принять")
w_button(3,70,160,50,25,"Отмена")
```

w_progress

Синтаксис:
w_progress(<заголовок>,n1,n2)
w_progress(n)
w_progress

Функция предназначена для показа хода выполнения. Функция возвращает 0, если нажата кнопка Отменить

Пример:

```
w_progress("Обработка файла",0,size(db))
        getfirst(db)
        n=1
        whle eof(db)=0
  rem обработка
          w_progress(n)
          getnext(db)
          n=n+1
wend
        w_progress
```

w_select

Синтаксис:
w_select(<x>,<y>,<w>,<h>,<заголовок>,<дескриптор файла>,<список полей>)

Показать окно для выбора значения из таблицы. Функция возвращает 1, если пользователь выбрал запись и нажал кнонку Принять, или 0, если пользователь нажал Отменить. При нажатии клавиши F1-F12 функция возвращает коды 112-123.

Пример:

```
subdb=open("listpr")
 if w_select(100,50,200,100,"Выберите запись",subdb,"Номер;Наименов")=1 then
    res=subdb["Наименов"]
 else
   res=""
end if
close(subdb)
```

w_settext

Синтаксис:
w_settext(<номер>,<текст>)

Функция изменяет текст, связанный с элементом окна под указанным номером

w_show

Синтаксис:
w_show

Показать окно. Функция возвращает номер нажатой кнопки.

Пример:

```
k=1
while k<>0
  k=w_show
  if k=1 then
    subdb=open("listpr")
    if w_select(100,50,200,100,"Выберите запись",subdb,"Номер;Наименов") then
      w_settext(4,subdb["Наименов"])
    end if
    close(subdb)
  else if k=2 then
    rem нажата OK
    db["Сумма"]=w_gettext(5)
    k=0
  else if k=3 then
    rem нажата Отмена
    k=0
  end if
wend
```

while

Синтаксис:
while <выражение>
** <операторы>**
wend

Оператор цикла. Тело цикла выполняется до тех пор, пока выражение является истинным.

Пример:

```
d = open("EXAMPLE")
getfirst(d)
while eof(d)=0
  'DbS_конец'.d = 'DbS_конец'.d + 100
  update(d)
  getnext(d)
wend
close(d)
end
```

ДО

ДО("Счет"[,"Субсчет"])

Вычисляет дебетовый оборот счета или, при необходимости, его субсчета за отчетный период.

Примеры:
а) ДО("51") - вычисляет дебетовый оборот 51 счета за отчетный период;
б) ДО("68","68-1") - вычисляет дебетовый оборот субсчета 68-1 за отчетный период.

Макрос ДО может находиться только в правой части оператора присваивания.

ДОНГ

ДОНГ("Счет"[,"Субсчет"])

Вычисляет дебетовый оборот счета (субсчета) с начала года.

Все остальное аналогично макросу ДО.

КО

КО("Счет"[,"Субсчет"])

Вычисляет кредитовый оборот счета или, при необходимости, его субсчета за отчетный период.

Все остальное аналогично макросу ДО.

Команды округления

Совместно с командой печати # используются команды округления:
а) **ТЫС** - при печати все числовые значения будут округляться до тысяч;
б) **РУБ** - при печати все числовые значения будут округляться до рублей;
в) **КОП** - до копеек.

Пример:

```
ТЫС
# Сальдо 51 счета на начало в тысячах [СНД("51")]
РУБ
# Сальдо 51 счета на начало в рублях [СНД("51")]
```

Будет напечатано:

```
Сальдо 51 счета на начало в тысячах 1200.00
Сальдо 51 счета на начало в рублях 1200000.00
```

КОНГ

КОНГ("Счет"[,"Субсчет"])

Вычисляет кредитовый оборот счета (субсчета) с начала года.

Все остальное аналогично макросу КО.

КС

КС("имя константы")

Возвращает символьное значение константы с именем "имя константы". Константа с указанным именем должна содержаться в справочнике констант. Константы обычно используются для указания названия предприятия, фамилии директора, главного бухгалтера, налоговых ставок.

База данных CONST.BTR должна содержать поля Код, Название, Значение.

Пример:
КС("А1") - возвращает символьное значение константы с именем А1.

КЧ

КЧ("имя константы")

Возвращает числовое значение константы с именем "имя константы". Все константы в справочнике констант хранятся в символьном виде. Пользователь сам может определить, как ему удобнее использовать ту или иную константу.

База данных CONST.BTR должна содержать поля Код, Название, Значение.

Пример:
КЧ("А1") - возвращает числовое значение константы с именем А1.

Логические и арифметические операции

В выражениях можно использовать операции:
Арифметические действия: *, /, +, -.
(+ для строк означает сцепление строк)
Операции сравнения чисел и строк: =, ==, !=, <>, >, <, >=, <=.
Логические действия: not, and, &, or, |.
Обращение к полю базы данных:
  '<имя поля>'.<дескриптор файла> (только для BTR)
или
  <дескриптор файла>["<имя поля>"] (BTR и DBF)
  <дескриптор файла>[<номер поля>] (BTR и DBF, нумерация полей с 1)

Обращение к элементу массива:
  <дескритор массива>[<номер элемента>]

ОБ

ОБ("Счет1"[,"Субсчет1"];"Счет2"[,"Субсчет2"])

Данный макрос вычисляет оборот между счетами или, при необходимости, их субсчетами за отчетный период. Квадратные скобки означают, что все, что находится внутри них, не является обязательным и может отсутствовать.

Пример:

ob = ОБ("68","68-1"; "51").

Переменной ob будет присвоено значение оборота за отчетный период между субсчетом 68-1 68 счета и 51 счетом.

Обороты субсчетов могут быть вычислены только для тех счетов, для которых при настройке плана счетов проставлена буква "д" в графе "Обороты".

Макрос ОБ может находиться только в правой части оператора присваивания.

ОБНГ

ОБНГ("Счет1"[,"Субсчет1"];"Счет2"[,"Субсчет2"])

Вычисляет оборот между счетами или,при необходимости,их субсчетами с начала года.

Примеры:
а) ОБНГ("69","69-1"; "51") - оборот по дебету 69-1 с кредита 51 с начала года;
б) ОБНГ("68";"51") - оборот по дебету 68 с кредита 51 с начала года.

Все остальное аналогично макросу ОБ.

Операторы языка DBASIC

Операторы для управления последовательностью выполнения программы:

- if-then
- for
- while
- goto
- gosub
- return
- end

Операторы для определения подпрограмм пользователя:

- sub
- function
- local
- $include

Команды для ввода-вывода:

- print
- input
- pause
- #
- Команды округления

Комментарий:

- rem

Подпрограммы для бухгалтерских программ

Эти подпрограммы (макросы) определены в файле **macros.bas** и предназначены для упрощения программ, выполняющих бухгалтерские расчеты.

Большинство макросов предназначены для получения различной информации о счетах:

- ДО
- ДОНГ
- КО
- КОНГ
- ОБ
- ОБНГ
- ПР
- СКД
- СКК
- СНД
- СНК

Прочие:

- УП

Все макросы допускают в качестве параметров выражения.

Примеры:
а) ОБ("6"+"8");
б) ОБНГ("68",'Db_расчеты'.d);
в) рег = "9703"
   УП(рег).

Подпрограммы для работы с окнами

w_open
w_close
w_select
w_show
w_button
w_edit
w_label
w_gettext
w_settext
w_progress
w_message
selectfiles

Подпрограммы для работы с базами данных

Подпрограммы для работы с базами данных:

- close
- begin_trn
- end_trn
- abort_trn
- clearbuf
- delete
- getfirst
- getlast
- getnext
- getprev
- unlock
- update
- closeall
- setbuffer
- setfilter

Функции для работы с базами данных:

- open
- size
- eof
- bof
- putkey
- find*
- numtables
- findfield
- getbuffer
- numfields
- numkeys
- fieldname
- getkey

- КС
- КЧ

Предопределенные переменные

- current_tables

Подпрограммы для работы с ККМ

Подпрограммы для работы с отчетами

Подпрограммы для работы с REP-отчетами:

- rpt_open
- rpt_view
- rpt_close
- rpt_write
- rpt_line

Подпрограммы для работы с QUE-отчетами:

- printque

Подпрограммы для работы с XLS- и DOC-отчетами:

- dde_open
- dde_close
- dde_execute
- dde_poke
- dde_request

Подпрограммы для создания текстовых отчетов

- redirect
- fromfile
- fromeof

Подпрограммы для работы с файлами

file_exists(имя) - файл существует?
copyfile(имя1,имя2) - копировать файл имя1 в имя2
renamefile(имя1,имя2) - переименовать файл имя1 в имя2
deletefile(имя) - удалить файл

Подпрограммы для работы со строками и преобразования

Функции преобразования:

- fldtodate
- fldtofulldate
- datetofld
- num
- str
- spro
- npro
- chr

Функции для работы со строками:

- strlen
- substr
- trim
- chr
- divstr
- strpos
- triml
- trimfull
- upper

Функции для работы с регулярными выражениями:

- regex_find_all
- regex_flags
- regex_match
- regex_replace
- regex_replacef
- regex_split
- regex_submatch

ПР

ПР(N,"имя_поля")

Макрос работы с проводками. Он используется в момент формирования проводок регистрируемой операции.

N - номер проводки;
имя_поля - имя поля проводки.

Например, операция состоит из двух проводок:

60 51
19 60.

При этом сумма второй проводки должна рассчитываться по формуле:

(Сумма 1 проводки / 120) * 20.

Файл prim.bas будет содержать:

ПР(2,"Сумма") = (ПР(1,"Сумма") / 120) * 20.

Макрос ПР может находиться как в левой, так и в правой части оператора присваивания.

Пример программы с разбором

Проведенный разбор программы позволит понять вдумчивому читателю механизм работы программ.

```
     d2 = open("tdata")
     getfirst(d2)
     data = substr('Тек.дата'.d2,1,6)
     close(d2)
     d = open("c:\btr_main\01")
     d1 = open("vrem")
     getlast(d1)
     sch = 'Db_счета'.d1
     sum = 0
     for i = 1 to size(d)
        if i = 1 then
          getfirst(d)
        else
          getnext(d)
        end if
        if trim(sch) != trim('Бал.счет'.d) then goto 100
        if substr('Дата'.d,1,6) >= data then goto 100
        'Износ'.d = 'Износ'.d - 'Износ_ТМ'.d
        'Износ_ТМ'.d = 0.0
        'Остаточная'.d = 'Бал.ст-ть'.d - 'Износ'.d
        if 'От_пробега'.d = 0.0 then
          'Износ_ТМ'.d = 'DbS_конец'.d * 'От_стоим.'.d / 1200
        else
          'Износ_ТМ'.d = 'DbS_конец'.d * ('От_пробега'.d/100) *
('Пробег'.d/1000)
        end if
        izn_old = 'Износ'.d
        'Износ'.d = 'Износ'.d + 'Износ_ТМ'.d
        if 'Износ_ТМ'.d >= 0.0 then 'Остаточная'.d = 'Бал.ст-ть'.d
- 'Износ'.d
        if 'Остаточная'.d < 0.0 then
          'Износ'.d = 'Бал.ст-ть'.d
          'Остаточная'.d = 0.0
          'Износ_ТМ'.d = 'Износ'.d - izn_old
        end if
        sum = sum + 'Износ_ТМ'.d
100     update(d)
     next
     close(d)
     getlast(d1)
     'Сумма'.d1 = sum
     update(d1)
     close(d1)
     end
```

Разберем построчно представленную выше программу.

d2 = open("tdata") - открывается база данных tdata.btr, содержащая текущую дату. При этом создается новая переменная d2, которой присваивается номер открытой базы данных.

getfirst(d2) - читается первая запись БД с номером d2.

data = substr('Тек.дата'.d2,1,6) - создается переменная data и ей присваивается значение подстроки поля 'Тек.дата' БД tdata.btr начиная с позиции 1 и длиной 6 символов.

close(d2) - Закрывается БД с номером d2.

d = open("c:\btr_main\01") - открывается база данных 01.btr, содержащая картотеку основных средств. При этом создается новая переменная d, которой присваивается номер открытой базы данных.

d1 = open("vrem") - открывается временная база данных vrem.btr, содержащая информацию об уже сформированных проводках текущей регистрируемой операции. Структура vrem.btr:

1. *Номер* - номер проводки
2. *Db_расчеты*
3. *Kr_расчеты*
   2 и 3 - содержат субсчета соответствующих счетов (4 и 5), если счета являются аналитическими;
4. *Db_счета*
5. *Kr_счета*
   4 и 5 - номера счетов;
6. *Шифр* - шифр проводки
   Шифры используются при написании программ, использующих уже посчитанные данные предыдущих проводок регистрируемой операции.
7. *Количество* - проводимое количество
8. *Сумма* - проводимая сумма
   Поле "Сумма" последней строки таблицы vrem.btr имеет нулевое значение. Задача по подсчету реального значения этого поля и возлагается на нашу программу.

getlast(d1) - прочитать последнюю запись БД с номером d1.

sch = 'Db_счета'.d1 - создается переменная sch и ей присваивается значение поля 'Db_счета' БД с номером d1.

sum = 0 - создается новая переменная sum и ей присваивается значение 0.

for i = 1 to size(d) - заголовок цикла, который должен будет перебрать записи от 1 до последней (size(d)) базы данных с номером d.

if i = 1 then

getfirst(d) - прочитать первую запись БД с номером d.

else

getnext(d) - прочитать следующую запись БД с номером d.

end if

if trim(sch) != trim('Бал.счет'.d) then goto 100 - если содержимое поля 'Бал.счет' не совпадает со значением переменной sch, то переход на конец цикла.

if substr('Дата'.d,1,6) >= data then goto 100 - если содержимое поля 'Дата' больше или совпадает с текущей датой, то переход на конец цикла.

'Износ'.d = 'Износ'.d - 'Износ_ТМ'.d - восстанавливается значение поля "Износ" (это делается на тот случай, если производится повторный перерасчет амортизации за текущий месяц).

'Износ_ТМ'.d = 0.0 - содержимое поля 'Износ_ТМ' БД с номером d устанавливается равным 0.0

'Остаточная'.d = 'Бал.ст-ть'.d - 'Износ'.d -расчет остаточной ст-ти ОС

if 'От_пробега'.d = 0.0 then

'Износ_ТМ'.d = 'DbS_конец'.d * 'От_стоим.'.d / 1200 - расчитывается износ текущего месяца от бал. стоимости.

else

'Износ_ТМ'.d = 'DbS_конец'.d * ('От_пробега'.d/100) * ('Пробег'.d/1000) - расчитывается износ текущего месяца от пробега.

end if

izn_old = 'Износ'.d

'Износ'.d = 'Износ'.d + 'Износ_ТМ'.d - расчитывается общий износ основного средства (с учетом текущего месяца).

if 'Износ_ТМ'.d >= 0.0 then 'Остаточная'.d = 'Бал.ст-ть'.d - 'Износ'.d

if 'Остаточная'.d < 0.0 then - проверка и расчет износа для тех ОС, которые полностью амортизировались.

'Износ'.d = 'Бал.ст-ть'.d

'Остаточная'.d = 0.0

'Износ_ТМ'.d = 'Износ'.d - izn_old

end if

sum = sum + 'Износ_ТМ'.d - накапливается общая сумма износа за текущий месяц для всех основных средств, амортизация которых относится на бал. счет, запомненный в переменной sch.

100 update(d) - запись изменений на диск.

next - конец цикла.

close(d) - закрыть БД с номером d.

getlast(d1) - прочитать последнюю запись БД с номером d1.

'Сумма'.d1 = sum - значение поля 'Сумма' БД с номером d1 устанавливается равным переменной sum.

update(d1) - запись изменений БД d1 на диск.

close(d1) - закрыть БД с номером d1.

end - конец программы.

Прочие подпрограммы

Прочие:

- min
- max
- round, floor, ceil
- abs
- system
- array
- error
- executeflag
- vartype
- resultfield
- getpath
- setpath
- keyarray
- keyfind
- keyfirst
- keynext
- keylast
- keyprev
- gettime
- sleep
- date_add
- date_diff
- date_set

СКД

СКД("Счет"[,"Субсчет"])

Вычисляет дебетовое сальдо счета (субсчета) на конец отчетного периода.

СКК

СКК("Счет"[,"Субсчет"])

Вычисляет кредитовое сальдо счета (субсчета) на конец отчетного периода.

СНД

СНД("Счет"[,"Субсчет"])

Вычисляет дебетовое сальдо счета (субсчета) на начало отчетного периода.

Примеры:
а) СНД("50");
б) СНД("68","68-2").

СНК

СНК("Счет"[,"Субсчет"])

Вычисляет кредитовое сальдо счета (субсчета) на начало отчетного периода.

УП

УП("Период")

Данный макрос устанавливает указанный отчетный период.

Период может иметь значения:
а) "Т", тогда УП("Т") - установить текущий отчетный период (по умолчанию всегда устанавливается текущий отчетный период);
б) "П", тогда УП("П") - установить предыдущий отчетный период;
в) "С", тогда УП("С") - установить следующий отчетный период;
г) "ГГММ"(где ГГ - год,ММ - месяц), тогда УП("9701") - установить январь 1997 года.

Возвращаемые значения:
1) в случае успеха макрос УП возвращает числовое значение соответствующее установленному отчетному периоду.
Например: S = УП("9703"), S будет равно 9703.00.
2) в случае неудачи возвращается 0.0.

DBASIC

Программы пишутся на языке, очень напоминающем Бейсик, но в него встроены операторы и функции, позволяющие оперировать с базами данных.

Регистрация ХО

Регистрация хозяйственных операций
Справочник проводок
Документы
Отложенные операции
Сдать в архив
Новый период
Новый год

Отчеты

Оборотно - сальдовая ведомость
Расшифровки (журналы - ордера)
Отчеты по аналитическим счетам
Анализ счета по датам
Главная книга
Актив баланса
Пассив баланса
Произвольные отчеты по аналитическим счетам
Произвольные выборки проводок
Произвольные отчеты по СП
Справки
Карточка объекта учета
Анализ оборотов по признакам
Дополнительные отчеты
Изменить отчетный период

Настройка

Окна

Каскад
Мозаика
Закрыть все

Запись

Смотреть
Вставить
Копировать
Изменить
Удалить
Найти
Найти далее
Изменить фильтр

Поле

Запомнить
Вспомнить
Очистить
Прибавить
Выбор

Файл

Создать
Открыть...
Обновить
Сохранить
Сохранить как...
Печать
Закрыть

Оборотно-сальдовая ведомость

Содержит информацию о входящих остатках, оборотах и исходящих остатках по каждому из субсчетов.

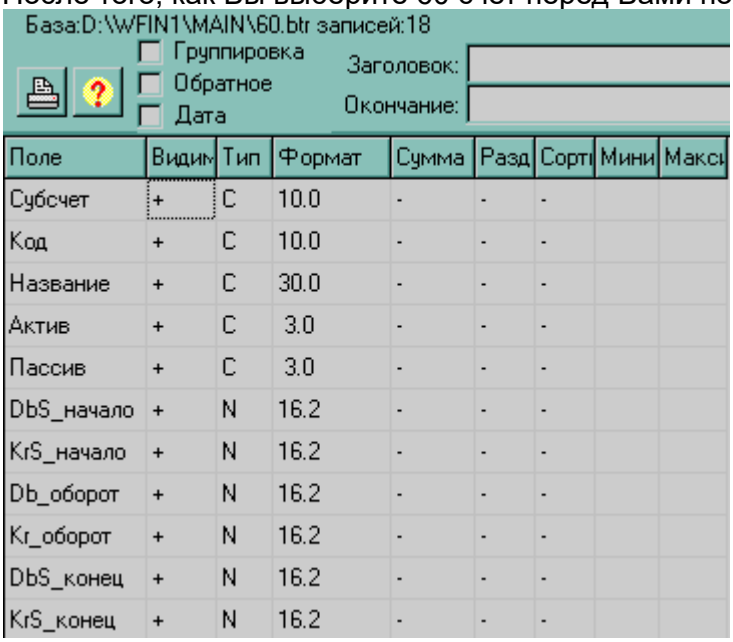
Расшифровки

По субсчету
По всем субсчетам (Форма 1)
По всем субсчетам (Форма 2)

Редактирование отчета

Порядок формирования произвольных отчетов рассмотрим на примере двух счетов: 60 и 10.

После того, как Вы выберите 60 счет перед Вами появится таблица:



Поле	Видим	Тип	Формат	Сумма	Разд	Сорт	Мини	Макс
Субсчет	+	C	10.0	-	-	-		
Код	+	C	10.0	-	-	-		
Название	+	C	30.0	-	-	-		
Актив	+	C	3.0	-	-	-		
Пассив	+	C	3.0	-	-	-		
DbS_начало	+	N	16.2	-	-	-		
KrS_начало	+	N	16.2	-	-	-		
Db_оборот	+	N	16.2	-	-	-		
Kr_оборот	+	N	16.2	-	-	-		
DbS_конец	+	N	16.2	-	-	-		
KrS_конец	+	N	16.2	-	-	-		

Рассмотрим, что обозначает каждый из столбцов:
1. Столбец "Поле" - содержит названия полей выбранной базы данных;
2. Столбец "Видимо" может содержать одно из двух значений:
   - "+" - означает, что соответствующее поле БД будет включено в отчет;
   - "-" - означает, что соответствующее поле БД не будет включено в отчет;
3. Столбец "Формат" - содержит формат, по которому будет выводиться соответствующее поле БД;
4. Столбец "Сумма" может содержать одно из двух значений:
   - "+" - означает, что нужно просуммировать соответствующее численное поле БД;
   - "-" - означает, что не нужно суммировать соответствующее численное поле БД;
5. Столбец "Разделять" может содержать одно из двух значений:
   - "+" - означает, что нужно завершить формирование таблицы, как только изменилось значение помеченных полей БД;
   - "-" - остальные поля БД должны быть помечены знаком "-";
6. Столбец "Сортировать" помечает те поля, по которым необходимо отсортировать БД перед формированием отчета.
7. Столбцы "Минимум" и "Максимум" могут содержать минимальное и максимальное значение поля (полей) для того, чтобы включить в формируемый отчет только те записи БД, которые удовлетворяют данному условию.

При перемещении по таблице необходимо пользоваться стрелками. При изменении "+" на "-" и наоборот пользоваться клавишей Enter. Для просмотра построенного отчета пользуйтесь клавишей F9.

А теперь рассмотрим несколько примеров отчетов.

Пример 1: *Выбрать все организации, которые должны нам более 100000 рублей.*

Таблица будет выглядеть так:

Поле	Видим	Тип	Форма	Сумма	Разд	Сорт	Минимум	Максимум
Субсчет	+	C	10.0	-	-	-		
Код	-	C	10.0	-	-	-		
Название	+	C	30.0	-	-	-		
Актив	-	C	3.0	-	-	-		
Пассив	-	C	3.0	-	-	-		
DbS_начало	-	N	16.2	-	-	-		
KrS_начало	-	N	16.2	-	-	-		
Db_оборот	-	N	16.2	-	-	-		
Kr_оборот	-	N	16.2	-	-	-		
DbS_конец	+	N	16.2	+	-	-	100000	9999999999
KrS_конец	-	N	16.2	-	-	-		

А сам отчет так:

Субсчет	Название	DbS_конец
001	Предприятие1	600000.00
002	Предприятие2	500000.00
010	Предприятие10	400000.00
021	Предприятие21	300000.00
045	Предприятие45	200000.00
066	Предприятие66	100000.00
		2100000.00

А теперь несколько примеров, связанных с 10 счетом.

Пример 2: *Сформировать сальдовые ведомости отдельно для каждого из складов. Оборотные ведомости отсортировать по группе материалов и названию.*

Таблица будет выглядеть так:

Поле	Видимо	Тип	Формат	Сумма	Разделять	Сортировать
Субсчет	+	C	10.0	-	-	-
Склад	-	C	12.0	-	+	1
Группа	+	C	8.0	-	-	2
Код	-	C	10.0	-	-	3
Название	+	C	37.0	-	-	-
Ед.изм.	-	C	4.0	-	-	-
Цена	-	N	12.2	-	-	-
Нач.кол-во	-	N	10.3	-	-	-
Нач.сумма	-	N	16.2	-	-	-
Приход_кол	-	N	10.3	-	-	-
Приход_сум	-	N	16.2	-	-	-
Расход_кол	-	N	10.3	-	-	-
Расход_сум	-	N	16.2	-	-	-
Ост.кол-во	+	N	10.3	-	-	-
Ост.сумма	+	N	16.2	+	-	-

А сам отчет так:

```
  Склад Центральный
+---------+--------+----------------------+------------+-------------+
¦ Субсчет ¦ Группа ¦ Название             ¦ Ост.кол-во ¦  Ост.сумма  ¦
+---------+--------+----------------------+------------+-------------+
¦ 001     ¦ 10-1   ¦Материал1             ¦  123.000   ¦   5765.00   ¦
¦ 002     ¦ 10-1   ¦Материал2             ¦   34.000   ¦   8768.00   ¦
¦ 003     ¦ 10-1   ¦Материал3             ¦  456.678   ¦    876.00   ¦
¦ 004     ¦ 10-2   ¦Материал1             ¦  876.089   ¦     67.00   ¦
¦ 005     ¦ 10-2   ¦Материал2             ¦   78.000   ¦    466.00   ¦
¦ 006     ¦ 10-2   ¦Материал3             ¦    0.000   ¦      0.00   ¦
+---------+--------+----------------------+------------+-------------+
¦         ¦        ¦                      ¦  XXX.XXX   ¦   XXXX.XX   ¦
+---------+--------+----------------------+------------+-------------+

  Склад Участок 1
+---------+--------+----------------------+------------+-------------+
¦ Субсчет ¦ Группа ¦ Название             ¦ Ост.кол-во ¦  Ост.сумма  ¦
+---------+--------+----------------------+------------+-------------+
¦ 0011    ¦ 10-1   ¦Материал11            ¦ 5123.000   ¦  55765.00   ¦
¦ 0021    ¦ 10-1   ¦Материал21            ¦  634.000   ¦  48768.00   ¦
¦ 0031    ¦ 10-2   ¦Материал31            ¦ 8456.678   ¦   3876.00   ¦
¦ 0041    ¦ 10-2   ¦Материал11            ¦ 1876.089   ¦    267.00   ¦
¦ 0051    ¦ 10-2   ¦Материал21            ¦  978.000   ¦   2466.00   ¦
¦ 0061    ¦ 10-2   ¦Материал31            ¦   10.000   ¦     10.00   ¦
+---------+--------+----------------------+------------+-------------+
¦         ¦        ¦                      ¦  XXX.XXX   ¦   XXXX.XX   ¦
+---------+--------+----------------------+------------+-------------+
```

После того, как Вы добились необходимой формы отчета, нажмите Esc. Появится окно с именем que - файла. Если Вы хотели бы сохранить построенный отчет, то нажмите Enter, если нет - то нажмите Esc.

Настройка печати

Быстрая печать - флаг для вывода на матричный принтер в текстовом (не графическом) режиме.
Использовать псевдографику - флаг вывода линий псевдографики без замены на + - |
Выбранный столбец - флаг вывода только выделенного столбца (части текста). Столбец выделяется с помощью стрелок в верхней части окна. Левая кнопка мыши перемещает левую границу, а правая - правую
С продолжением - флаг позволяющий напечатать несколько коротких отчетов на одной странице
Сжатый - флаг для печати сжатым (уменьшенным) шрифтом.

Также можно задать:
Размер шрифта
Поля (отступы от краев бумаги)
Интервал между строками

Создать

Метка
Линия
Текст
Рисунок
Ячейка
Блок

Отложенные операции

Отложенные операции - это перечень хозяйственных операций, которые Вы временно хотели бы "отложить" (не вносить в систему). Такая потребность может возникнуть, например, в следующих случаях:

- При вводе хозяйственных операций следующего отчетного периода.
- При вводе хозяйственной операции, которая, возможно, будет иметь место в дальнейшем.
- При необходимости проверить: " а что если " данный пакет проводок внести в систему.

Режим работы "ОТЛОЖЕННЫЕ ОПЕРАЦИИ" предоставляет пользователю следующие возможности:

- Регистрация отложенных операций (ОО)
- Просмотр и модификация справочника ОО
- Перенос отложенных проводок в систему
- Перенос проводок из системы в отложенные