VisiCAR

Руководство разработчика VisiCAR™

Разработчики:

Программное обеспечение:
ООО «КИГЛИ» 03150, г. Киев, ул. Боженко 11, оф. 507 т/ф (044) 227-87-23
E-Mail: info@uamap.net

Картографическое обеспечение:
Научно-исследовательский институт геодезии и картографии (НИИ ГК)
03150, г. Киев, ул. Красноармейская, 69 т. (044) 227-06-84 ф. (044) 227-42-52
E-Mail: info@gki.com.ua

Оглавление

Введение **5**

Глава 1 **Работа с отчетами** **7**

1.1 Создание и удаление отчета 7
Установка параметров экземпляра источника. 9
1.2 Редактирование отчета 10
1.3 Редактор отчета 11
Основные компоненты окна дизайнера отчетов 11
Общие свойства стандартных объектов 12
Общие методы стандартных объектов 13
Объект “Текст” 14
Объект “Секция” (бэнд) 18
Объект “Рисунок” 20
Объект “Линия” 21
Объект «RichText» 22
Объект “Фигура” 22
Объект “Прямоугольник с тенью” 23
Объект “Флажок” (CheckBox) 23
Объект “Диаграмма” 24
Объект “Штрихкод” 26
Объект “OLE” 26
Объект “Вложенный отчет” (SubReport) 27
Объект “Кросс-таб” 27
Панель инструментов “Стандартная” 27
Панель инструментов “Текст” 29
Панель инструментов “Прямоугольник” 30
Панель инструментов “Выравнивание” 31
Клавиши управления 31
Управление мышью 32
1.4 Просмотр и печать отчета. 32
1.5 Виды отчетов 33
Отчет с одним уровнем (список) 33
Отчет с двумя уровнями 33
Отчет с тремя уровнями 33
Cross-tab отчет 34
Динамические отчеты 34
Разрываемые бэнды 35
Многоколоночный отчет 36
Отчет с титульным листом 36
Вложенный отчет 36
Master-Detail-Detail отчет 37
Отчет без бэндов 37
Отчет с группами 37
Отчет с диаграммами 38
1.6 Параметры отчета 38
Параметры страницы 39
Опции дизайнера 41
Инспектор объектов 43
Словарь данных 43
1.7 Конструктор выражений 46
1.8 Диалоговые формы 48
Элементы управления 49

Общие свойства 49
Общие методы 50
Label 50
Edit 51
Memo 51
Button 52
CheckBox 53
RadioButton 54
ListBox 54
ComboBox 55
Диалоговая форма 56
Передача информации в отчет 57
1.9 Языковые средства 58
Скрипты и объекты 59
Написание кода 60
Использование переменных 60
Константы 61
Обращение к объектам 61
Использование процедур и функций 62
Модификация объектов 62
1.10 Описание встроенных процедур и функций, а также свойств и методов источников данных 63
Встроенные переменные и функции. 63
Агрегатные функции 63
Функции работы со строками 64
Арифметические функции 66
Класс ARGS 67
Управление построением отчета 69
Системные переменные 70
Интерфейсы источников данных 71
1.11 Применение 151
Масштаб, проекция, система координат. 153

Введение

Как работать с этим руководством

Руководство разработчика состоит из отдельных глав. Как правило, каждая глава состоит из нескольких разделов.

Принятые обозначения

Названия управляющих элементов интерфейса (например, кнопок, различных опций, закладок и т.д.) в руководстве приводятся стандартным шрифтом с полужирным начертанием (например **ОК**). Таким же начертанием указывается сочетание клавиш. Если подразумевается одновременное нажатие нескольких клавиш – то такое действие обозначается при помощи знака “+” (например **Ctrl+1** означает одновременное нажатие клавиш “Ctrl” и “1”).
Пункты меню отображены стандартным шрифтом с наклонным полужирным начертанием (например ***Сервис > Конфигурирование***).

Некоторые абзацы руководства помечены слева картинкой с изображением лампочки. Так обозначается материал, который следует воспринимать как совет для более эффективного использования программы VisiCAR™.

Другие абзацы помечены слева картинкой с изображением знака вослицания. Так обозначается материал, требующий особого внимания. Также в таких абзацах речь идет о различных ограничениях при работе с программой.

Терминология

главное меню	меню самого верхненго уровня, содержащее перечень основных разделов программы
меню	контекстное меню, содержащее список некоторых функций для выбора.
[Д]	панель инструментов диспетчера
[К]	панель инструментов работы с картой

Последовательность выбора в меню обозначается символом “ > ” (например ***Пуск > Программы***).
Если необходимо нажать кнопку (например, открытие карты) на панели инструментов работы с картой – в руководстве это будет обозначено следующим образом : [К: открыть новую карту].

Глава 1

Работа с отчетами

В этой главе рассматриваются основные функции создания и редактирования отчетов о произошедших событиях, координатах объекта, содержимом пользовательской базы данных и т.п.

1.1 Создание и удаление отчета

Для создания отчета:

* Выберите в главном меню ***Отчеты>Новый отчет...*** или воспользуйтесь кнопкой ***Новый отчет***  панели инструмсентов отчетности. На экране появится мастер создания отчетности.
* В появившемся окне укажите имя создаваемого документа. Оно должно быть уникальным. В противном случае на экране появится сообщение об ошибке. Следует указывать такие название, которые бы соответствовали содержанию отчета (например “Координаты объекта ...”, “Событие ***Нападение на водителя***” и т.п.).
* Для создания отчета с использованием уже готового шаблона необходимо активизировать опцию **Создать отчет по шаблону**, а затем (после нажатия кнопки **Далее**) выбрать из перечня существующих отчетов тот, который будет использоваться в качестве образца (шаблона).
* Если необходимо создать оригинальный отчет (не используя образцы) — флажок опции **Создать отчет по шаблону** необходимо снять.
* Нажмите кнопку **Далее**.

Выберите категорию, к которой необходимо отнести создаваемый документ (категориями будем называть условные группы, на которые можно разделить все создаваемые отчеты). Для этого активизируйте опцию **Отнести к уже существующей**, а затем выберите нужную категорию. Существует ряд зарезервированных групп:

- Окно карты
- Запросы на картах
- План
- Объекты
- Архив событий
- Сообщения сервера
- Измерение расстояния
- Пользователь
- Права доступа
- Карта
- Водитель

* Если соответствующей категории в перечне не оказалось — создайте новую. Для этого активизируйте опцию **Создать новую**. Укажите название создаваемой категории.
  Если оставить это поле незаполненным — отчет не будет отнесен ни к одной категории.
* Нажмите кнопку **Далее**.

* В появившемся окне необходимо выбрать источники информации, которые могут быть использованы при формировании документа. Под источниками данных будем понимать условные объекты, от которых может потребоваться информация при составлении отчета. Доступные источники данных:
  — **any_table -** инициализация любой таблицы базы данных в качестве источника данных.

*Нельзя инициализировать источник данных **any_table**, указав параметры инициализации по умолчанию (необходимо выбрать таблицу) (см. Источник данных: any_table на стр. 150).*

  — **car_pj** - проекция мобильных объектов (см. Источник данных: car_pj на стр. 73);
  — **current_fix_archive** - источник данных текущего состояния мобильного объекта на открытой карте при просмотре архива (см. Источник данных: current_fix_archive на стр. 88);
  — **dataset_functions** - таблица функций источников данных (см. Источник данных: dataset_functions на стр. 123);
  — **dataset_props** - таблица переменных источников данных (см. Источник данных: dataset_props на стр. 125);
  — **datasets_list** - таблица всех существующих источников данных (см. Источник данных: datasets_list на стр. 128);
  — **events_archive** - архив координат перемещений (см. Источник данных: events_archive на стр. 73);
  — **fix_archive** - весь архив (см. Источник данных: fix_archive на стр. 78);
  — **fix_current** - источник данных текущего состояния мобильного объекта на открытой карте при наблюдении или просмотре архива перемещений (см. Источник данных: fix_current на стр. 83);
  — **fix_last** - последнее в архиве (см. Источник данных: fix_last на стр. 97);
  — **img_process** - обработка изображения (см. Источник данных: img_process на стр. 115);
  — **map_proj** - картографическая проекция (см. Источник данных: map_proj на стр. 101);
  — **map_window -** видимый участок карты (см. Источник данных: map_window на стр. 118);
  — **obj_inf** - информация о мобильных объектах (см. Источник данных: obj_inf на стр. 107);
  — **poll_error** - ошибки (см. Источник данных: poll_error на стр. 110);
  — **virtual_dataset** - виртуальная таблица(см. Источник данных: virtual_dataset на стр. 127);
  — **table_<имя таблицы>** - источник данных, созданный пользователем (встроенная база данных). Этот источник является таблицей, содержащей информацию пользователя. Пользователь может создавать, редактировать и удалять таблицы. Все источники данных типа **table_<имя таблицы>** имеют одинаковые функции. Переменные источников зависят от типа и количества полей таблицы (см. Источник данных: table_<имя таблицы> на стр. 134).
  — **user** - информация о пользователе и его правах(см. Источник данных: user на стр. 151).
  — **dbb_storadge** - источник, используемый для получения информации о таблицах, содержащих данные пользователя. Таблицы могут

создаваться удаляться и редактироваться пользователем (см. Источник данных: dbb_storadge на стр. 129).

- По умолчанию будет создано по одному экземпляру каждого выбранного источника информации. Для того, чтоб создать несколько экземпляров хотя бы одного источника — блокируйте действие опции **Создать по одному экземпляру всех выбранных источников**. Нажмите кнопку **Далее**.
- Если на предыдущем шаге действие опции **Создать по одному экземпляру всех выбранных источников** было отменено — в новом окне укажите количество создаваемых экземпляров источника информации. Несколько экземпляров одного источника нужно создавать при необходимости получения информации: об одном и том же мобильном объекте, но за различные промежутки времени; от разных мобильных объектов, для обработки рисунков (один рисунок — один источник данных) и т.п.
- Выберите экземпляры источника информации, которые необходимо инициализировать (задать параметры) для создаваемого отчета. Не отмеченные источники информации могут быть инициализированы из программного кода. При инициализации будут установлены параметры получения информации (см. Установка параметров экземпляра источника. на стр. 9).
- Имя экземпляра источника информации можно изменить. Для этого в поле **Уникальный идентификатор экземпляра** укажите новое название, а затем нажмите клавишу <***Enter***>. Если имя экземпляра не уникально — на экране появится сообщение об ошибке.
- Выберите источники информации, которые необходимо инициализировать. Нажмите кнопку **Далее** и установите параметры каждого отмеченного экземпляра (табл. 1.1 на стр. 10).

Установка параметров экземпляра источника.

Опции установки параметров:

- **Использовать параметры по умолчанию**.
  При активизации этой опции будут использоваться параметры, заданные по умолчанию для выбранного экземпляра (например, для источника fix_archive — это архив координат для всех доступных мобильных объектов за текущий день).

Нельзя инициализировать экземпляр источника данных ***any_table,*** *выбрав опцию* ***Использовать параметры по умолчанию.***

- **Задать параметры сейчас и использовать их каждый раз при формировании отчета**.
  Эта опция позволяет инициализировать экземпляр мобильного объекта на стадии проектирования отчета для использования установленных параметров всякий раз при генерировании документа. Для этого активизируйте опцию и нажмите кнопку ... . После чего установите необходимые параметры (табл. 1.1 на стр. 10).
  Если у экземпляра нет параметров, которые можно было бы указать — при нажатии кнопки ... ничего не произойдет.
- **Инициализировать экземпляр в процессе формирования отчета**.
  При активизации этой опции установка параметров будет происходить

непосредственно при составлении отчета, что позволяет гибко изменять параметры информации, содержащейся в отчете

Таблица 1.1: Параметры экземпляра источника данных

источник	*параметры*
car_pj	нет
events_archive	мобильный объект (объекты) период просмотра событие (события)
fix_archive	мобильный объект (объекты) период просмотра
fix_last	мобильный объект
map_proj	нет
obj_inf	нет
poll_error	мобильный объект (объекты) период просмотра
map_window	нет
current_fix_archive	нет
dataset_functions	нет
dataset_props	нет
datasets_list	нет
fix_current	нет
img_process	нет
virtual_dataset	нет
dbb_storadge	нет
table_<имя таблицы>	нет
user	нет
any_table	таблица, которая будет инициализирована в качестве источника данных.

* Нажмите кнопку **Далее**.
* На экране появится окно, отображающее дерево всех используемых источников информации, а также их экземпляров.
* Нажмите кнопку **Готово**. На экране появится окно **Редактора отчета**.
* Создайте проект отчета, воспользовавшись инструментами окна **Редактора отчета** (см. Руководство пользователя, раздел Редактор отчета на стр. 12).
* Сохраните отчет.

* Для удаления отчета выберите в главном меню ***Отчеты>Удалить отчет...*** или воспользуйтесь кнопкой  меню отчетности.
* Укажите необходимый отчет.
* Нажмите кнопку **Удалить**.

1.2 Редактирование отчета

Для редактирования ранее созданного отчета:

* Выберите в главном меню ***Отчеты>Редактор отчетов...*** или воспользуйтесь кнопкой  меню отчетности.
* На экране появится окно, содержащее элементы, описанные в таблице:

Таблица 1.2: Элементы окна редактирования отчета

Перечень отчетов на выбор	Список отчетов, доступных для редактирования
Показать список всех отчетов, имеющихся в архиве	При активизации этой опции в перечне будут указаны названия всех доступных отчетов
Не изменять параметры источников данных	При активизации этой опции редактирование параметров источников данных будет запрещено. Вы будете иметь возможность просмотреть дерево используемых источников, их экземпляров, а также внести изменения непосредственно в проекты отчетов

* Выберите отчет для изменения.
* Активизируйте необходимые опции и нажмите кнопку **Далее**.
* Установите необходимые параметры экземпляров источников информации (доступно при блокировании опции **Не изменять параметры источников данных**):
  — выберите источники информации, необходимые для составления отчета;
  — блокируйте (активизируйте) опцию **Создавать по одному экземпляру всех выбранных источников**;
  — укажите количество создаваемых экземпляров источников (доступно при отмене действия опции **Создавать по одному экземпляру всех выбранных источников** на предыдущем шаге);
  — выберите экземпляры источников информации, которые необходимо активизировать;
  — установите параметры для каждого отмеченного экземпляра;
  — нажмите кнопку **Далее**.
* В появившемся окне отображено дерево всех источников информации, а также их экземпляров.
* Нажмите кнопку **Готово**.
* Внесите необходимые изменения в окне **Редактора отчета** (см. Руководство пользователя, раздел Редактор отчета на стр. 12)

1.3 Редактор отчета

Окно редактора отчета предоставляет пользователю удобные средства для разработки содержания отчета и позволяет сразу выполнять предварительный просмотр проектируемого документа. Интерфейс дизайнера выполнен на современном уровне с использованием панелей инструментов, расположение которых можно изменять по своему вкусу.

Рис. 1.1. *Основные компоненты окна дизайнера отчетов*

Таблица 1.3: Объекты Дизайнера

Иконка	*Название*	*Описание*
	Выбор объекта	Предназначен для выбора объекта в окне Дизайнера.
	Текст	Предназначен для отображения текста (однострочного или многострочного). В тексте могут присутствовать переменные (см. Руководство пользователя, раздел Языковые средства на стр. 59).
	Бэнд	Бэнд (англ. band - полоска). Позволяет объединять объекты в группы (заголовок отчета, страницы, многострочная часть, подвал и пр.).
	Рисунок	Предназначен для отображения рисунка в форматах BMP, ICO, WMF, EMF, JPG.
	Вложенный отчет (SubReport)	Предназначен для создания вложенных отчетов. Будучи вставленным в отчет, приводит к созданию новой страницы, содержимое которой будет выведено вместо объекта при формировании отчета.
	Линия	При выборе этого объекта курсор принимает форму карандаша. На странице отчета при этом можно рисовать вертикальные и горизонтальные линии.
	RichText	Отображает форматированный текст (формат RTF). В тексте могут присутствовать переменные/ выражения (см. Руководство пользователя, раздел Языковые средства на стр. 59).
	Фигура	Отображает один из четырех типов фигур: прямоугольник, прямоугольник с закругленными углами, эллипс и треугольник.

Таблица 1.3: Объекты Дизайнера (продолжение)

	Текст с тенью	Выполняет те же функции, что и объект "Текст". Кроме того, “умеет” отображать тень или градиентную заливку. Очень удобный объект для печати наклеек.
	OLE объект	Отображает объект OLE.
	Кросс-таб объект	Позволяет формировать кросс-таблицы.
	Флажок	Отображает крестик или галочку внутри прямоугольника.
	Диаграмма	Отображает данные в виде диаграммы.
	Штрихкод	Отображает данные в форме штрих-кода.

Все визуальные объекты отчета являются потомками класса TfrView. Он имеет ряд свойств и методов, которые являются общими для всех визуальных объектов.

Общие свойства стандартных объектов

- **Integer BandAlign**
  Выравнивание внутри бэнда. Одна из констант baNone, baLeft, baRight, baCenter, baWidth, baBottom
- **Boolean Enabled**
  Разрешает или запрещает работу объекта
- **Integer FillColor**
  Цвет фона объекта. Может быть задан константой clXXX
- **Integer FrameColor**
  Цвет рамки объекта
- **Integer FrameStyle**
  Стиль линии рамки. Одна из констант psSolid, psDash, psDot, psDashDot, psDashDotDot, psDouble
- **Double FrameWidth**
  Толщина рамки
- **Integer Height**
  Высота объекта
- **Integer Left**
  Координата X левого верхнего угла объекта
- **String Memo**
  Текст, находящийся в мемо объекта. К этому свойству можно также обращаться с использованием индекса, например: Memo[1]
- **Integer Memo.Count**
  Возвращает количество строк в мемо
- **String Name**
  Имя объекта

- **Boolean Stretched**
  Является ли объект растягиваемым
- **Integer Top**
  Координата Y левого верхнего угла объекта
- **Boolean Visible**
  Видимость объекта
- **Integer Width**
  Ширина объекта

Общие методы стандартных объектов

- **Hide()**

Аналогично установке Visible := False
Параметры:
Не передаются.
Возвращаемые значения:
Нет.

- **Memo.Add (String)**

Параметры:
String - строка, добавляемая в мемо.
Возвращаемые значения:
Нет.

- **Memo.Clear()**

Очищает мемо
Параметры:
Не передаются.
Возвращаемые значения:
Нет.

- **Memo.Delete(Integer)**

Удаляет строку с данным индексом из мемо
Параметры:
Integer - индекс строки, которую необходимо удалить.
Возвращаемые значения:
Нет.

- **Show()**

Аналогично установке Visible := True
Параметры:
Не передаются.
Возвращаемые значения:
Нет.

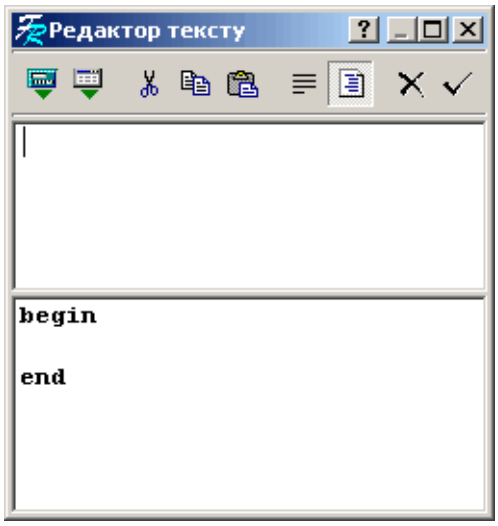
Объект “Текст”

Представляет собой произвольным образом обрамленный прямоугольник с многострочным текстом внутри. Задается цвет текста/заливка, тип рамки, толщина рамки и цвет, все параметры шрифта текста, выравнивание текста внутри прямоугольника и ориентация текста (норм./90 градусов). Для этого предназначены панели инструментов "Текст" (см. на стр. 30) и "Прямоугольник" (см. на стр. 31).

В мемо объекта можно поместить многострочный текст с переменными (см. Руководство пользователя, раздел Языковые средства на стр. 59). Переменные обрамляются квадратными скобками. При формировании отчета, когда в объекте обнаруживается переменная, вызывается событие, которому передается имя переменной (это может быть и не имя, а выражение) и ожидается возврат ее значения. Примеры использования переменных:

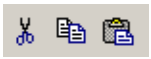
Длина, см: [Длина] – текст (Длина, см:) и переменная ([Длина])
Длина, см: [[Длина в дюймах] * 2.54] – текст и выражение с переменной

Рис. 1.2.



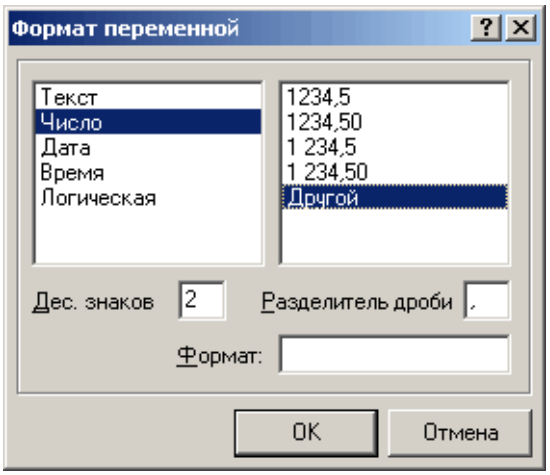
Редактор текста (рис.) позволяет изменять содержимое мемо, быстро вставлять переменные, выражения в мемо, а также редактировать скрипт. Для более удобной работы можно воспользоваться быстрыми клавишами: ***Insert*** - вставка переменной; клавиша ***Ctrl+Enter*** - аналог кнопки **OK**.

Таблица 1.4: Кнопки панели инструментов

Кнопка	*Функции*
	вызывает конструктор выражений (см. раздел Конструктор выражений на стр. 47.);
	вызывает диалог вставки поля БД в текст (не используется в рамках работы с программой **VisiCAR™**);
	операции с буфером обмена;
	включает или выключает перенос слов в длинных строках;
	включает или выключает окно со скриптом;
	кнопка **Отмена**;
	кнопка **ОК**.

Помимо основного редактора, имеется редактор, позволяющий выбрать формат отображения имеющихся в тексте переменных. Вызвать его можно либо из контекстного меню объекта, либо из **инспектора объектов** (свойство DisplayFormat).

Рис. 1.3.



Переменная может отображаться как текст, число, дата, время, булевское значение (список в левой части окна). Для каждого из этих типов (кроме текстового представления) можно выбрать один из нескольких способов форматирования (список в правой части окна), например, для числа можно задать количество знаков после запятой, знак-разделитель целой и дробной части, разделение на разряды.
Кроме того, для каждого типа может быть выбран произвольный формат, когда строка форматирования задается вручную (например, #,###,##0.000 для числа). Для этого необходимо в списке справа выбрать тип форматирования "Другой". Строки форматирования для каждого из типов описаны в справочной системе Delphi, раздел formatting strings (кроме булевского типа, где строка представлена в формате Значение1;Значение2). Если выбран текстовый тип (по умолчанию), то форматирование производиться не будет. Указанные опции форматирования будут воздействовать на все переменные, содержащиеся в объекте. Если переменная не может быть отформатирована указанным способом, она выводится как текст.
Если в объекте несколько переменных, для которых нужно разное форматирование, то можно применить другой способ форматирования. Суть его заключается в том, что к переменной дописывается тэг формата:

[Переменная формат], где формат - одно из следующих значений:

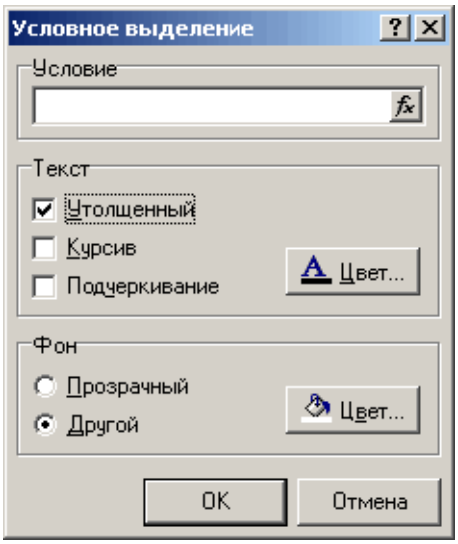
- #x.x или #Nx.x или #Nyyyyy - числовое представление. x.x - длина числа/ число разрядов после запятой; yyyyy - строка типа #,##0.00 (более подробную информацию можно получить в справочной системе Delphi, раздел formatting strings). Если строка x.x или yyyyy содержит символы ".", ",", "-", то последний такой символ будет использован в качестве разделителя целой и дробной части форматированного числа.
- #Dxxxxx, #Txxxxx - дата и время. xxxxx - строка типа dd.mm.yy.
- #Bxxxxx;yyyyy - булевское значение. Если значение - ложь, то выводится строка xxxxx, иначе yyyyy.

Вот примеры использования форматирования:

[Table1."N1" #9.2] [Table1."N2" #N9-2] [Table1."N3" #N#,##0.00] - для числа
[Table1."Date1" #Ddd.mm.yyyy] [Table1."Time" #Thh:mm:ss] - для даты и времени
[Table1."Bool1" #BFalse;True] [Table1."Bool2" #BНет;Да] - для булевского значения

Для того, чтобы отобразить объект другим шрифтом или цветом в зависимости от выполнения какого-либо условия, можно воспользоваться диалогом **Условное выделение** (рис. 1.4). Диалог можно вызвать с помощью кнопки  панели инструментов "Текст".

Рис. 1.4.



К примеру, нужно выделить суммы заказов, превышающие 1000$ жирным шрифтом. Для этого в объекте, содержащем сумму заказа, в диалоге указывается условие – Value > 1000 и задаются параметры шрифта. Условие выделения можно указать и по-другому – [Part total] > 1000. Другой пример использования выделения – для придания отчету презентабельного вида можно сделать чередующуюся раскраску строк данных. Для этого на бэнд надо поместить объект "Текст", растянуть его по ширине секции, в диалоге выделения набрать условие [LINE#] mod 2 = 0 и указать цвет фона (к примеру, серый).

В контекстном меню объекта можно задать следующие опции:

- "растягиваемый" (Stretched) — высота объекта будет зависеть от количества строк в нем. При этом опция "растягиваемый" должна быть включена у бэнда, на котором находится объект. При печати бэнда вычисляется его высота, и все объекты с включенным растягиванием выводятся таким образом, что их нижняя граница растягивается до нижней границы бэнда.
- "перенос слов" (WordWrap) — если слово не помещается в строке, оно переносится на следующую строку.
- "перенос по слогам" (WordBreak) — при переносе слова делается разбивка его на слоги. При этом опция "перенос слов" должна быть включена.
- "только текст" (TextOnly) — содержимое объекта трактуется как текст, переменные и выражения не обрабатываются.
- "скрывать повторяющиеся" (Suppress) — не выводить объекты с повторяющимися значениями.
- "автоширина" (AutoWidth) — при печати рамка растягивается по ширине объекта.

Кроме того, в инспекторе объектов можно задать следующие значения:
CharSpacing — расстояние между буквами;
GapX, GapY — отступы текста слева и сверху;
LineSpacing — расстояние между строками текста.

Свойства

- **Integer Alignment**
  Выравнивание текста внутри объекта. Набор значений frtaLeft, frtaRight, frtaCenter, frtaVertical, frtaMiddle, frtaDown.
- **Boolean AutoWidth**
  Если True, ширина объекта устанавливается равной ширине имеющегося в объекте текста.
- **Integer CharSpacing**
  Расстояние между символами.
- **String Font.Name**
  Имя шрифта, которым будет выведен текст объекта.

- **Integer Font.Size**
  Размер шрифта.
- **Integer Font.Style**
  Стиль шрифта. Набор констант fsBold, fsItalic, fsUnderline.
- **Integer Font.Color**
  Цвет шрифта.
- **Integer GapX**
  Промежуток между рамкой объекта и текстом.
- **Integer GapY**
  Промежуток между рамкой объекта и текстом.
- **Boolean HideZeros**
  Если True, не выводить нулевые значения переменных.
- **Integer LineSpacing**
  Расстояние между строками текста.
- **Boolean Suppress**
  Подавлять повторяющиеся значения.
- **Boolean TextOnly**
  Если True, обработка переменных не производится.
- **Boolean WordBreak**
  Переносить слова по слогам (только русские слова).
- **Boolean WordWrap**
  Переносить длинные слова на следующую строку.

Методы

Нет.

Объект “Секция” (бэнд)

Ниже приведен полный перечень бэндов и выполняемых ими функций.

Таблица 1.5: Бэнды

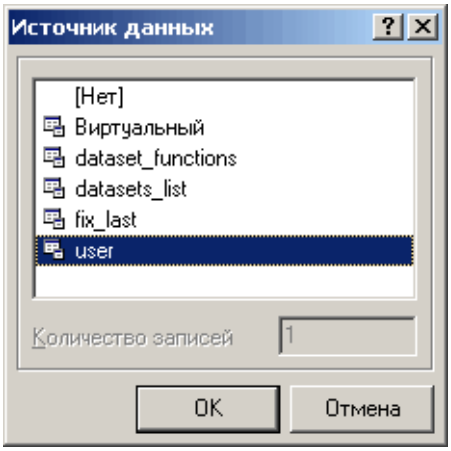
Название	*Функция*
Report title	печатается один раз в начале отчета
Report summary	печатается один раз в конце отчета
Page header	печатается вверху на каждой странице
Page footer	печатается внизу на каждой странице
Master header	печатается в начале списка 1-го уровня
Master data	данные списка 1-го уровня
Master footer	печатается в конце списка 1-го уровня
Detail header	печатается в начале списка 2-го уровня
Detail data	данные списка 2-го уровня
Detail footer	печатается в конце списка 2-го уровня
Subdetail header	печатается в начале списка 3-го уровня

Таблица 1.5: Бэнды (продолжение)

Subdetail data	данные списка 3-го уровня
Subdetail footer	печатается в конце списка 3-го уровня
Overlay	печатается на каждой странице нижним слоем
Column header	печатается в начале каждой колонки
Column footer	печатается в конце каждой колонки
Group header	заголовок группы
Group footer	печатается после группы
Cross header Cross data Cross footer	эта группа бэндов предназначена для создания отчетов с переменным количеством столбцов и разбивкой на страницы
Child	может быть прикреплен к любому из бэндов, кроме Page footer, и выводиться вместе с ним

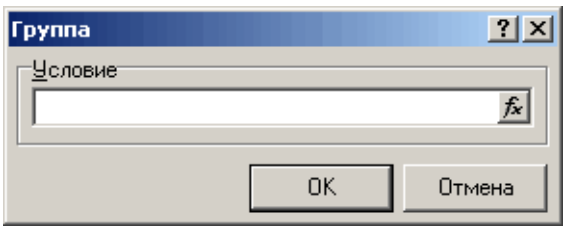
В зависимости от типа бэнда вызываются разные редакторы. Редактор для дата-бэндов (рис. 1.5) позволяет выбрать источник данных из списка доступных либо выбрать виртуальный источник данных. При выборе виртуального источника необходимо указать, из скольких записей он состоит. При формировании отчета бэнд с виртуальным источником будет напечатан столько раз, сколько записей было задано.

Рис. 1.5.



Редактор для бэнда GroupHeader (рис. 1.6) позволяет ввести условие группировки. Остальные бэнды не имеют редакторов.

Рис. 1.6.



В зависимости от типа бэнда в контекстном меню доступны следующие опции:

* "растягиваемый" (Stretched) — высота бэнда определяется максимальной высотой находящихся в нем объектов. У объектов также должна быть включена опция "растягиваемый";
* "разрываемый" (Breaked) — если бэнд не помещается целиком на странице, будут выведены только поместившиеся строки текста, а вывод остальных будет продолжен с новой страницы;
* "формировать новую страницу" (FormNewPage) — после печати бэнда формирование отчета продолжается с новой страницы. Если опция включена у дата—бэнда, то новая страница будет сформирована после вывода всех подбэндов;
* "на первой странице" (OnFirstPage) — если опция отключена, то бэнд не будет выведен на первой странице;
* "на последней странице" (OnLastPage) — если опция отключена, то бэнд не будет выведен на последней странице;
* "повторять на всех страницах" (RepeatHeader) — эта опция доступна у бэндов Master header, Detail header, Subdetail header, Group header, Cross header. При включении дублирует бэнд на новой странице или колонке.

Свойства

* **Boolean Breaked**
  Бэнд является разрываемым
* **String ChildBand**
  Дочерний бэнд
* **Integer ColumnGap**
  Промежуток между колонками
* **Integer Columns**
  Количество колонок у дата-бэнда
* **Integer ColumnWidth**
  Ширина колонки
* **String Condition**
  Условие группировки для бэнда group header
* **String DataSource**
  Источник данных для дата-бэнда
* **Boolean EOF**
  True, если достигнут конец набора записей, связанного с бэндом
* **Boolean FormNewPage**
  Формировать новую страницу после печати бэнда и всех его детальных бэндов
* **String Master**
  Бэнд, по которому выполняется группировка
* **Boolean OnFirstPage**
  Печатать на первой странице
* **Boolean OnLastPage**
  Печатать на последней странице
* **Boolean PrintChildIfInvisible**
  Печатать дочерние (child) бэнды, если родительский бэнд невидимый
* **Boolean PrintIfSubsetEmpty**
  Печатать бэнд, если его детальный список пуст

* **Boolean RepeatHeader**
  Повторять бэнд на новой странице

Методы

* **First()**

Устанавливает источник данных, связанный с бэндом, на первую запись
Параметры:
Не передаются.
Возвращаемые значения:
Нет.

---

* **Next()**

Устанавливает источник на следующую запись
Параметры:
Не передаются.
Возвращаемые значения:
Нет.

---

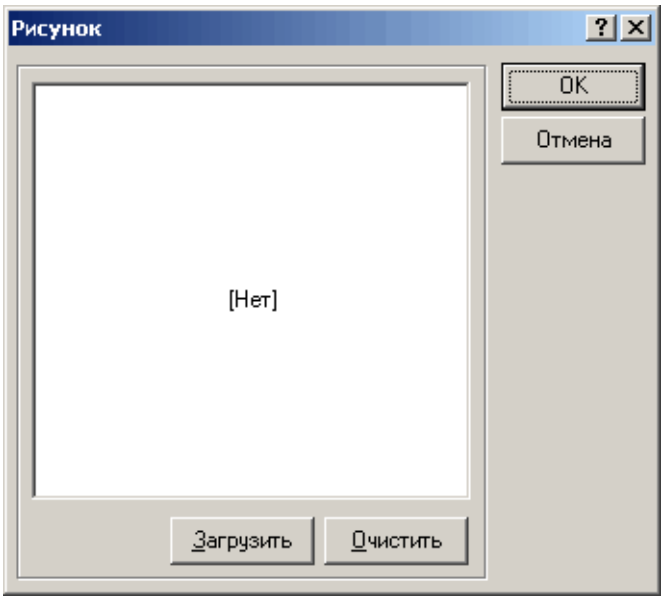
* **Prior()**

Устанавливает источник на предыдущую запись
Параметры:
Не передаются.
Возвращаемые значения:
Нет.

Объект "Рисунок"

Предназначен для вставки графического файла в формате BMP/WMF/ICO (и JPG, если установлен соответствующий модуль) в документ. Редактор объекта позволяет вставить рисунок и очистить содержимое объекта.

Рис. 1.7.



Редактор мемо может быть вызван с помощью инспектора объектов либо комбинацией клавиш Ctrl+Enter. Эту операцию можно также выполнить из инспектора объектов (свойство DataField).

В контекстном меню объекта и в инспекторе можно задать следующие опции:

- "растягиваемый" (Stretched) — рисунок растягивается по границам объекта;
- "сохранять пропорции" (KeepAspect) — если включено растягивание, пропорции рисунка сохраняются;
- "центровка" (Center) — рисунок выводится в центре объекта.
- BlobType — если рисунок хранится в поле таблицы БД, эта опция позволяет выбрать тип хранимого рисунка: BMP, WMF, ICO или JPG (не используется в рамках работы с программой **VisiCAR™**).

Свойства

- **Integer BlobType**
  Тип рисунка, содержащегося в BLOb-поле. Одна из констант btBMP, btJPG, btICO, btWMF
- **Boolean Center**
  Центрировать рисунок внутри объекта
- **String DataField**
  BLOb-поле таблицы, содержащее рисунок
- **Boolean KeepAspect**
  Сохранять пропорции рисунка при растягивании

Методы

Нет

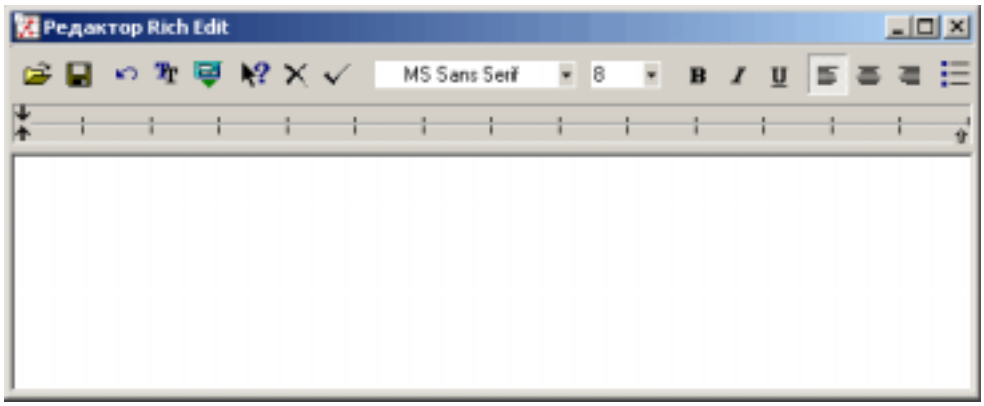
Объект “Линия”

Представляет собой вертикальную либо горизонтальную линию с изменяемым цветом и толщиной. Дизайнер имеет удобные средства для рисования линий. В частности, с помощью линий можно быстро задать внешнее обрамление для таблицы, отличное от внутреннего.
Для рисования линии выберите инструмент "Линия" на панели объектов. При этом курсор мыши примет форму карандаша при перемещении его над поверхностью листа. Установите курсор мыши в место начала линии, нажмите левую кнопку мыши и двигайте мышь до нужной позиции. После отпускания кнопки будет нарисована линия. После окончания рисования нужно нажать кнопку "выбор объекта" либо <***Esc***>.

Объект “RichText”

Предназначен для вставки текста в формате RTF (Rich Text format) в документ. В частности, таким образом можно вставить текст, подготовленный в текстовом редакторе MS Word, предварительно сохранив его в формате RTF. Редактор объекта (рис. 1.8) позволяет выполнять все базовые операции над текстом - менять шрифт, выравнивание, создавать маркированные списки.

Рис. 1.8.



В тексте могут находиться переменные (см. Руководство пользователя, раздел Языковые средства на стр. 59), обрамленные квадратными скобками — как и в объекте «Текст». Редактор мемо может быть вызван с помощью инспектора объектов либо комбинацией клавиш <***Ctrl + Enter***>. Эту операцию можно также выполнить из инспектора объектов (свойство DataField).

Свойства

- **Integer GapX**
  Промежуток между рамкой объекта и текстом
- **Integer GapY**
  Промежуток между рамкой объекта и текстом
- **Boolean TextOnly**
  Если True, обработка переменных не производится.
- **String DataField**
  BLOb-поле таблицы, содержащее данные

Методы

Нет.

Объект “Фигура”

Предназначен для вставки геометрических фигур (прямоугольник, прямоугольник с закругленными углами, эллипс, треугольник) в документ. Тип фигуры можно установить в инспекторе объектов (свойство Shape).

Свойства

- **Integer Shape**
  Тип фигуры. Одно из значений skRectangle, skRoundRectangle, skEllipse, skTriangle, skDiagonal1, skDiagonal2.

Методы

Нет.

Объект “Прямоугольник с тенью”

Выполняет те же функции, что и объект “Текст”. Отображает тень, градиентную заливку. Этот объект является потомком объекта "Текст".

Следовательно, он имеет тот же набор свойств и методов, что и родитель, и кое-что свое:

Свойства

- **Integer BeginColor**
  Начальный цвет градиентной заливки
- **Integer EndColor**
  Конечный цвет градиентной заливки
- **Boolean Gradient**
  Использовать градиентную заливку
- **Boolean RoundRect**
  Использовать закругление
- **Integer RoundSize**
  Размер закругения
- **Integer ShadowColor**
  Цвет тени
- **Integer ShadowWidth**
  Ширина тени
- **Integer Style**
  Стиль градиентной заливки. Одна из констант gsVertical, gsHorizontal, gsElliptic, gsRectangle, gsHorizCenter, gsVertCenter.

Методы

Нет.

Объект “Флажок” (CheckBox)

Представляет собой прямоугольник с крестиком внутри. Если в мемо объекта поместить ссылку на переменную, то крестик будет показан при ненулевом значении переменной. В мемо можно также поместить символ «0» либо отличный от нуля.
В инспекторе объектов можно выбрать тип CheckBox’a: с крестиком или с галочкой (свойство CheckStyle).

Свойства

- **Integer CheckColor**
  Цвет крестика
- **Integer CheckStyle**
  Стиль (крестик/галочка). Константы csCross, csCheck
- **String DataField**
  Поле таблицы, содержащее данные

Методы

Нет

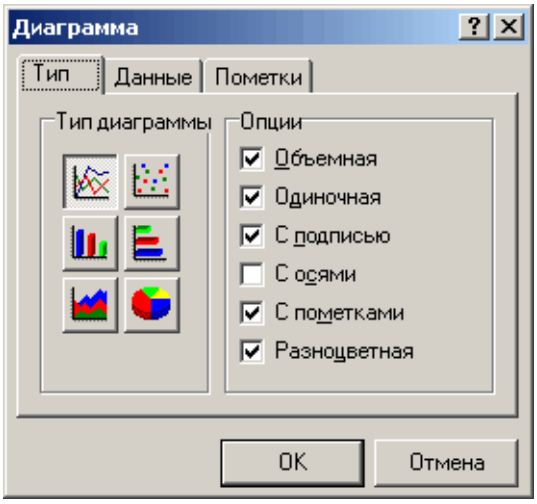
Объект “Диаграмма”

Предназначен для вставки графика или диаграммы в документ. Возможно использование шести типов графиков: линии, точки, столбцы горизонтальные и вертикальные, области и круговые диаграммы. Редактор объекта

представляет собой блокнот с тремя закладками: "Тип" (рис. 1.9), "Данные" (рис. 1.10) и "Пометки" (рис. 1.11). Здесь можно задать различные опции для графика, а также указывать объекты, содержимое которых будет использовано при построении графика.

Закладка "Тип"

Рис. 1.9.



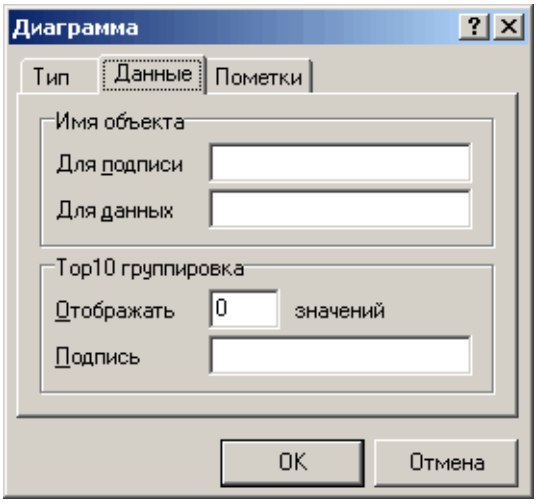
На этой закладке можно выбрать один из шести типов диаграммы, а также указать различные опции:

- "объемная" — при включении диаграмма становится трехмерной;
- "одиночная" — должна быть включена, если на диаграмме расположен один график;
- "с подписью" — при включенной опции выводит рядом с диаграммой поясняющую подпись;
- "с осями" — выводить оси (нужно отключать для круговой диаграммы);
- "с пометками" — выводит пометки над каждым значением
- "разноцветная" — если опция включена, каждое значение в диаграмме (столбец, сектор, линия) показывается разными цветами.

При построении отчета информация будет накапливаться в мемо:
Заголовок1;Заголовок2;Заголовок3
Число1;Число2;Число3
График можно построить и вручную, если занести в мемо объекта соответствующие значения. Если количество строк заголовка не соответствует количеству значений, график выведен не будет.

Закладка "Данные"

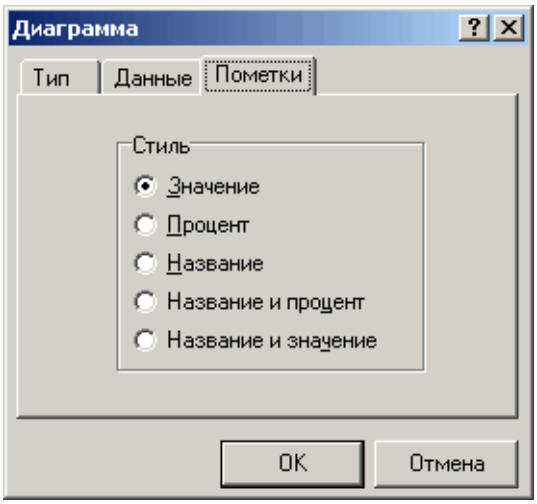
Рис. 1.10.



Для построения графика необходимо указать имена двух объектов типа "Текст", в которых содержится название и соответствующее числовое значение. При построении отчета содержимое этих объектов будет накапливаться в мемо диаграммы.
Если объект содержит число в форматированном виде или комбинацию числа и строки (например, "$1000" или "10 000"), то программа попытается выделить числовую часть из строки. Алгоритм выделения — удаляются все нецифровые символы в начале и в конце строки, после этого удаляются все символы - разделители разрядов (как правило, пробелы). То есть если задано более сложное форматирование, например: "10000км2", то такие объекты использовать уже нельзя. В этом случае следует создать невидимый объект с аналогичным содержимым, но без всякого форматирования, и указывать его имя. Невидимым объект можно сделать с помощью инспектора объектов в дизайнере, установив параметр Visible в 0.
Объект позволяет легко строить графики типа "Top10", т.е. график, в котором представлены только первые самые большие значения, а остальные значения просуммированы в отдельную группу с названием. Для этого необходимо указать количество выводимых значений и подпись (обычно в качестве подписи используется слово "Другие").

Рис. 1.11.

Закладка "Пометки"



На закладке “Пометки” можно указать, какие значения выводить в качестве пометок.

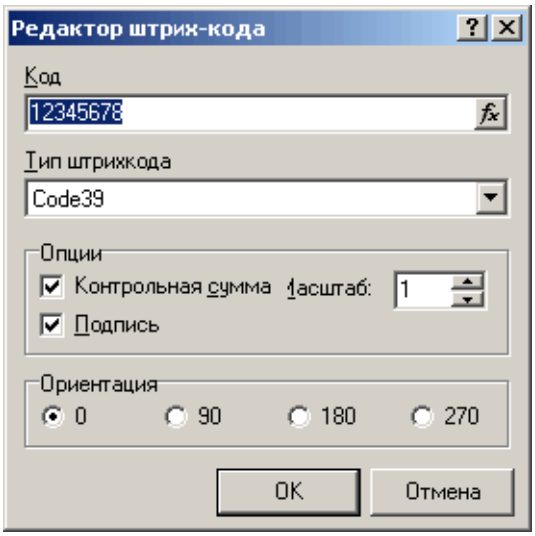
Объект “Штрихкод”

Предназначен для вставки штрихкодов в документ. Можно использовать следующие типы штрихкодов:

* 2 of 5 interleaved
* Code39

- Code39 Extended
- Code128A-C
- Code93
- Code93 Extended
- MSI
- PostNet
- Codebar
- EAN8
- EAN13
- EAN128A-C
- UPC A, E0, E1, Supp2, Supp5

Рис. 1.12.



Свойства

- **String DataField**
  Поле таблицы, содержащее данные.

Методы

Нет

Объект “OLE”

Предназначен для вставки OLE объектов в документ. Редактор объекта «OLE» позволяет вставить новый объект и редактировать существующий.
Вставка нового объекта производится в стандартном диалоговом окне. В контекстном меню объекта опция "растягиваемый" (Stretched) позволяет корректно отображать данные в некоторых случаях (например, при вставке объекта "Лист Excel").

Объект “Вложенный отчет” (SubReport)

Вложенный отчет — это отчет с размещенным на нем объектом SubReport, который представляет собой ссылку на другой отчет, расположенный на

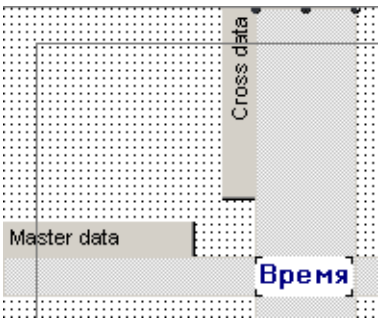
отдельной странице. При формировании такого отчета вместо объекта SubReport будет выведен соответствующий отчет.
Объекты SubReport можно располагать на листе друг под другом или рядом. В случае, если объекты надо расположить друг под другом, их необходимо разнести по разным дата-бэндам (т.е. первый отчет — в первом бэнде Masterdata, второй — во втором и т.д.). На использование SubReports накладываются следующие ограничения:

* нельзя использовать колонки;
* во вложенных отчетах (не в основном!) игнорируются бэнды ReportTitle, ReportSummary, PageHeader, PageFooter, ColumnXXX;
* нельзя использовать разрываемые бэнды;
* нельзя использовать группы.

Объект “Кросс-таб”

Предназначен для проектирования отчета, число столбцов которого на этапе построения не известно.
При формировании отчета столбцы, не поместившиеся на странице, будут перенесены на следующую. Для построения этого отчета разместите на странице бэнд "master data" и бэнд "cross data". Бэнды crossXXX размещаются на странице вертикально. На пересечении бэндов поместите объект, который будет отображаться в качестве ячейки таблицы. Необходимо поместить объект таким образом, чтобы он полностью помещался в границы области пересечения бэндов.

Рис. 1.13.



Если бэнд "cross data" пересекает несколько обычных бэндов, то необходимо назначить источники данных для каждого из бэндов.
Как видно, для построения cross-tab отчета необходимо как минимум два набора данных, связанных между собой отношением master-detail.
Назначение бэндов "cross header", "cross footer": footer печатается последним столбцом (например, если нужно отпечатать сумму по строке), a header — первым. Кроме того, если в опциях бэнда "cross header" включен флажок "Выводить на всех страницах", то при переносе широкой таблицы на смежный лист header будет продублирован.
Отчет этого типа может иметь переменную высоту строки данных. Для этого надо включить опцию "растягиваемый" у бэнда "master data". При этом формирование кросс-таблицы будет происходить в два прохода: на первом проходе определяется максимальная высота строки, на втором выполняется вывод данных.

Панель инструментов “Стандартная”

Рис. 1.14.

Таблица 1.6: Панель “Стандартная”

Иконка	*Название*	*Описание*
	Новый отчет	Создает новый пустой отчет
	Открыть отчет	Открывает существующий отчет из файла с расширением FRF. Клавиатурный аналог — ***Ctrl+O***
	Сохранить отчет	Сохраняет отчет в файл с расширением FRF (доступно из окна предварительного просмотра). Клавиатурный аналог — ***Ctrl+S***
	Предварительный просмотр	Выполняет построение отчета и его предварительный просмотр. Клавиатурный аналог — ***Ctrl+P***
	Вырезать	Вырезает выделенные объекты в буфер обмена. Клавиатурный аналог — ***Ctrl+X***
	Копировать	Копирует выделенные объекты в буфер обмена. Клавиатурный аналог — ***Ctrl+C***
	Вставить	Вставляет объекты из буфера обмена. Клавиатурный аналог — ***Ctrl+V***
	Отменить	Отменяет последнюю операцию. Число уровней отката — до 100. Клавиатурный аналог — ***Ctrl+Z***
	Повторить	Повторяет последнюю отмененную операцию. Клавиатурный аналог — ***Ctrl+Y***
	На передний план	Помещает выделенные объекты на передний план
	На задний план	Помещает выделенные объекты на задний план
	Выделить все	Выделяет все объекты на странице. Клавиатурный аналог — ***Ctrl+A***
	Новая страница	Создает новую страницу
	Новая форма диалога	Создает новую форму диалога (см. раздел Диалоговые формы на стр. 49.)
	Удалить страницу	Удаляет текущую страницу
	Свойства страницы	Вызывает диалог со свойствами страницы

Таблица 1.6: Панель “Стандартная” (продолжение)

	Показать сетку	Показывает точечную сетку на странице. Шаг сетки можно задать в опциях дизайнера. Клавиатурный аналог — ***Ctrl+G***
	Выравнивать объекты по сетке	При перемещении или изменении размеров объектов координаты/размеры будут изменяться скачкообразно в соответствии с шагом сетки
	Расположить в узлах сетки	Изменяет размеры/расположение выделенных объектов таким образом, чтобы они размещались в узлах сетки
	Справка	Показывает справку по элементам управления
Вихід	Выход	Закрывает окно дизайнера

Панель инструментов “Текст”

Рис. 1.15.

Таблица 1.7: Панель “Текст”

Иконка	*Название*	*Описание*
Arial	Шрифт	Позволяет выбрать название шрифта из выпадающего списка. “Помнит” пять последних использованных шрифтов. Двойной щелчок мышью вызывает стандартный диалог выбора шрифта
10	Размер шрифта	Позволяет выбрать размер шрифта из выпадающего списка. Размер можно также указать вручную
Ж	Утолщение	Устанавливает/снимает утолщение шрифта. Клавиатурный аналог — ***Ctrl+B***
К	Наклон	Устанавливает/снимает наклон шрифта. Клавиатурный аналог — ***Ctrl+I***
Ч	Подчеркивание	Устанавливает/снимает подчеркивание шрифта. Клавиатурный аналог — ***Ctrl+U***
	Цвет шрифта	Выбирает цвет шрифта из выпадающего списка
	Условное выделение	Показывает диалог с атрибутами выделения для выбранного объекта "Текст"

Таблица 1.7: Панель “Текст” (продолжение)

	Выравнивание влево	Устанавливает выравнивание текста влево
	Выравнивание по центру	Устанавливает выравнивание текста по центру
	Выравнивание вправо	Устанавливает выравнивание текста вправо
	Выравнивание по ширине	Устанавливает выравнивание текста равномерно по ширине
	Выравнивание по верхнему краю	Устанавливает выравнивание текста по верхнему краю
	Выравнивание по высоте	Устанавливает выравнивание текста по высоте
	Выравнивание по нижнему краю	Устанавливает выравнивание текста по нижнему краю
	Вертикальный текст	Переключает нормальный/вертикальный шрифт

Панель инструментов “Прямоугольник”

Рис. 1.16.

Таблица 1.8: Панель “Прямоугольник”

Иконка	*Название*	*Описание*
	Верхняя линия	Включает/выключает верхнюю линию рамки
	Левая линия	Включает/выключает левую линию рамки
	нижняя линия	Включает/выключает нижнюю линию рамки
	правая линия	Включает/выключает правую линию рамки
	Все линии	Включает все линии рамки
	Нет линий	Выключает все линии рамки
	Цвет фона	Выбирает цвет фона из выпадающего списка

Таблица 1.8: Панель “Прямоугольник” (продолжение)

	Цвет линии	Выбирает цвет линии из выпадающего списка
	Стиль линии	Выбирает стиль линии из выпадающего списка
	Толщина	Выбирает толщину линии из выпадающего списка

Панель инструментов “Выравнивание”

Рис. 1.17.

Таблица 1.9: Панель “Выравнивание”

Иконка	*Описание*
	Выровнять левые края
	Центрировать по горизонтали
	Центрировать по горизонтали в окне
	Расположить равномерно по ширине
	Выровнять правые края
	Выровнять верхние края
	Центрировать по вертикали
	Центрировать по вертикали в окне
	Расположить равномерно по высоте
	Выровнять нижние края

Клавиши управления

— Стрелки — перемещение по объектам.
— ***Ctrl*** + Стрелки — плавное перемещение выделенного объекта (объектов).
— ***Shift*** + Стрелки — уменьшение/увеличение размеров объекта.
— ***Alt*** + Стрелки — склеивание объекта с ближайшим в выбранном направлении.
— ***Enter*** — вызов редактора объекта.

- ***Delete*** — удаление объекта.
- ***Ctrl + Enter*** — вызов редактора мемо.
- ***Ctrl + 1..9*** — установка толщины рамки объекта.
- ***Ctrl + Z*** — отменить последнее действие. Число уровней отката — до 100.
- ***Ctrl + Y*** — повторить отмененное действие.
- ***Ctrl + G*** — показать сетку.
- ***Ctrl + B***, ***Ctrl + I***, ***Ctrl + U*** — управление стилем шрифта — аналоги кнопок ***Ж***, ***К***, ***Ч*** на панели текста.
- ***Ctrl + F*** — обрамить выделенный объект (объекты).
- ***Ctrl + D*** — снять обрамление с объекта.
- ***Ctrl + X*** — вырезать в буфер.
- ***Ctrl + V*** — вставить из буфера.
- ***Ctrl + C*** — копировать в буфер.
- ***Ctrl + A*** — выделить все объекты на листе.
- ***Ctrl + N*** — новый отчет.
- ***Ctrl + O*** — открыть файл с отчетом.
- ***Ctrl + S*** — сохранить отчет в файле.
- ***Ctrl + P*** — предварительный просмотр отчета.

Управление мышью

- ***Левая кнопка*** — выбор объекта; вставка нового объекта; перемещение и изменение размеров объекта (объектов).
- ***Правая кнопка*** — контекстное меню объекта (объектов), над которым находится указатель мыши.
- **Двойной щелчок левой кнопкой** — вызов редактора объекта. Если двойной щелчок сделать на пустом месте листа, то вызывается диалоговое окно **Параметры страницы**.
- ***<Shift + левая кнопка>*** — если объект уже выделен, то снимает с него выделение, иначе объект выделяется. Выделение остальных объектов не изменяется.
- ***<Ctrl + левая кнопка>*** — при нажатии и перемещении мыши рисуется рамка; после отпускания кнопки мыши выделяются все объекты, попавшие в рамку. Такого же эффекта можно добиться, если щелкнуть левой кнопкой мыши на пустом месте листа, и, не отпуская кнопки, потянуть мышь до нужной позиции.
- При перемещении бэнда его объекты перемещаются вместе с ним. Если это нежелательно, то перемещение следует выполнять при нажатой клавише <***Alt***>.
- Изменить масштаб выделенных объектов можно, потянув мышкой за красный квадратик на нижнем правом углу группы выделенных объектов.
- При приближении указателя мыши к границе двух соседних объектов вид указателя изменяется. Если при этом нажать левую кнопку и переместить указатель мыши, граница объектов также перемещается (горизонтальный сплиттинг).

1.4 Просмотр и печать отчета.

Для просмотра отчета:

* Выберите в главном меню ***Отчеты>Просмотр отчета...***.

- Выберите категорию, к которой принадлежит необходимый для просмотра отчет. Отчеты, не принадлежащие ни к одной категории, расположены в нижней части списка категорий.
- Некоторое время придется подождать пока пройдет определение параметров всех не инициализированных источников данных. Если на этапе проектирования при установке параметров (см. на стр. 9) была выбрана опция **Инициализировать экземпляр в процессе формирования отчета** — укажите необходимые параметры.
- На экране появится выбранный отчет.

Для печати отчета:

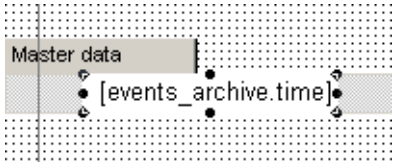
- Выберите в главном меню ***Отчеты>Печать отчета...***.
- Выберите категорию, к которой принадлежит необходимый для печати отчет. Отчеты, не принадлежащие ни к одной категории, расположены в нижней части списка.
- Установите все необходимые настройки выбранного принтера.
- Нажмите кнопку **ОК**.

1.5 Виды отчетов

Отчет с одним уровнем (список)

Это самый простой вид отчета. Для его создания необходимо разместить на листе бэнд "master data" и поместить в него объекты со ссылками на нужные данные. Бэнд необходимо подключить к источнику данных.

Рис. 1.18.



Отчет с двумя уровнями

Для построения этого отчета необходимо разместить на листе бэнды "master data" и "detail data", затем поместить в них объекты со ссылками на нужные данные. Следует отметить, что таблицы (или запросы), на которых строится такой отчет, должны быть связаны отношением master-detail.
Порядок, в котором бэнды размещаются на листе, не имеет значения — сначала будет напечатана строка из master, потом соответствующие ей строки из detail. Отчет будет формироваться до тех пор, пока есть данные для master. Если надо, чтобы master с его подчиненным списком печатался с нового листа, установите для бэнда флажок "Формировать новую страницу" (FormNew-Page). У отчета есть особенность — если для текущей строки из master список detail пуст, строка master печататься не будет. Если все же необходимо распечатать такие строки, для бэнда "master data" нужно установить флажок "Печатать, если detail пуст" (PrintIfSubsetEmpty).

Отчет с тремя уровнями

Для построения отчета разместите на листе бэнды "master data", "detail data" и "subdetail data". В остальном принципы построения такие же, как и для отчета с двумя уровнями детализации.

Cross-tab отчет

Этот вид отчета предназначен для печати таблицы, число столбцов которой на этапе построения отчета неизвестно. При формировании отчета столбцы, не поместившиеся на странице, будут перенесены на следующую. Для построения этого отчета разместите на странице бэнд "master data" и бэнд "cross data". Бэнды crossXXX размещаются на странице вертикально. На пересечении бэндов поместите объект, который будет отображаться в качестве ячейки таблицы. Необходимо поместить объект таким образом, чтобы он полностью помещался в границы области пересечения бэндов (рис. 1.13).
Это все, что нужно для построения базового отчета. Осталось назначить источники данных для "master data" и "cross data". Если бэнд "cross data" пересекает несколько обычных бэндов, то необходимо назначить источники данных для каждого из бэндов. Как видно, для построения cross-tab отчета необходимо как минимум два набора данных, связанных между собой отношением master-detail.
Назначение бэндов "cross header", "cross footer": footer печатается последним столбцом (например, если нужно отпечатать сумму по строке), а header — первым. Кроме того, если в опциях бэнда "cross header" включен флажок "Выводить на всех страницах", то при переносе широкой таблицы на смежный лист header будет продублирован.
Отчет этого типа может иметь переменную высоту строки данных. Для этого надо включить опцию "растягиваемый" у бэнда "master data". При этом формирование кросс-таблицы будет происходить в два прохода: на первом проходе определяется максимальная высота строки, на втором выполняется вывод данных.

Динамический отчет

В динамических отчетах высота бэндов зависит от высоты находящихся в них объектов. Объектами, которые могут растягиваться в зависимости от количества имеющегося в них текста, являются объекты "Текст", "Текст с тенью" и "RichText".
Чтобы сделать объект растягиваемым, надо включить его флажок "Растягиваемый" (Stretched). Кроме того, этот же флажок надо включить у бэнда, на котором расположен объект. При печати растягиваемого бэнда программа считает максимальную высоту объектов, находящихся на бэнде, и устанавливает высоту бэнда равной максимальной высоте объектов (рис. 1.19) и (рис. 1.20).

Рис. 1.19.

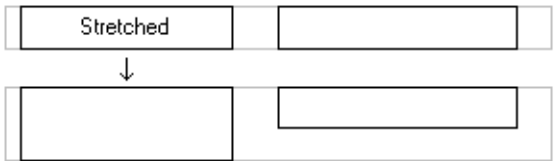
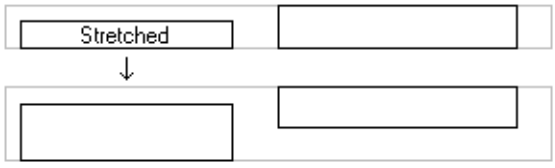
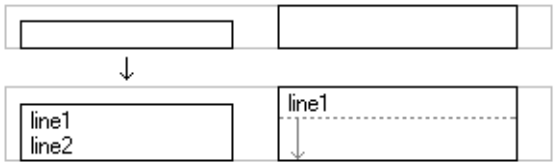


Рис. 1.20.



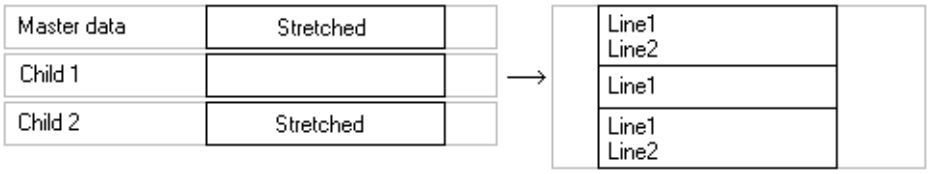
Если на бэнде несколько растягиваемых объектов, расположенных рядом, то при растягивании нижние границы всех объектов (независимо от их актуальной высоты) будут совпадать с нижней границей бэнда (рис. 1.21).

Рис. 1.21.



А как быть, если растягиваемые объекты нужно поместить друг под другом, либо под растягиваемым объектом поместить нерастягиваемый? Для этого применяется бэнд Child. Объекты, находящиеся друг под другом, один или несколько из которых растягиваемые, нужно “разнести” по разным бэндам .

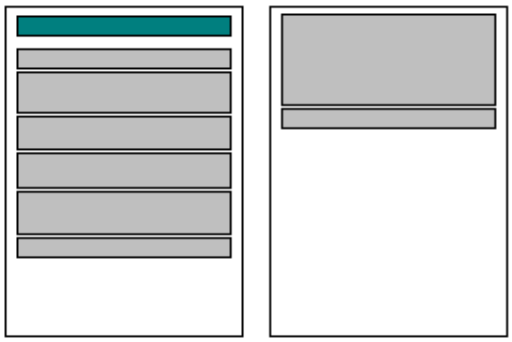
Рис. 1.22.



При этом всем Child — бэндам надо также установить флажок "Растягиваемый". Для связывания бэндов служит свойство "ChildBand" в инспекторе объектов. В вышеприведенном примере нужно связать Master data с Child 1 и Child 1 с Child 2.
Следует отметить, что все объекты имеют рамку. С помощью рамок отчету можно легко придать табличный вид. При растягивании объекта рамка растягивается вместе с ним.

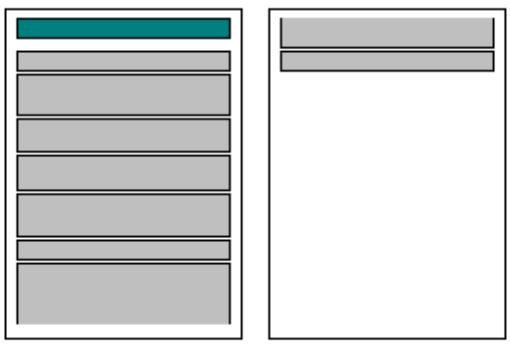
Разрываемые бэнды

Рис. 1.23.



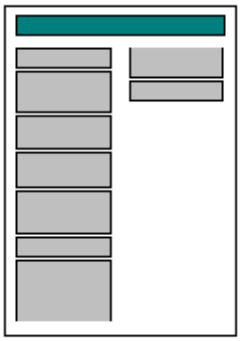
При печати бэнда программа проверяет наличие свободного места на странице, и, если оно меньше необходимого, формирует новую страницу и печатает бэнд на ней. При этом место на странице расходуется неэкономно, особенно в случаях, когда высота бэнда достаточно большая (рис. 1.23).

Рис. 1.24.



Для того, чтобы вывести на листе столько строк текста, сколько помещается, бэнд должен иметь помимо включенной опции "Растягиваемый" опцию "Разрываемый" (Breaked). При этом отчет будет выглядеть примерно так (рис. 1.24).

Рис. 1.25.



Если отчет многоколоночный (см. на стр. 37), то содержимое объектов будет переноситься на следующую колонку (рис. 1.25).
Разрывать содержимое могут объекты "Текст", "Текст с тенью" и "RichText". Все остальные объекты будут выведены на той странице, где для этого достаточно свободного места.

Многоколоночный отчет

Если в обычном отчете при достижении конца страницы формирование продолжается с нового листа, то в многоколоночном — на том же листе, но рядом, т.е. в несколько смежнорасположенных колонок. Количество колонок задается в опциях страницы. Обычный отчет можно превратить в многоколоночный, всего лишь задав количество колонок в опциях страницы. Бэнды "column header" и "column footer" позволяют снабдить каждую колонку заголовком и нижней частью.
Кроме того, количество колонок можно задавать отдельно для каждого дата-бэнда. Если установить свойство Columns > 1 у бэнда, то вывод строк данных будет происходить слева направо и сверху вниз. Можно также указать ширину колонки и промежуток между колонками. Бэнд отображает текущие установки в дизайнере штриховыми линиями.

Отчет с титульным листом

Можно легко делать многостраничные отчеты (например, первая страница — титульный лист, все остальное - прочая информация). Для того, чтобы создать/удалить лист, используйте кнопки на панели инструментов либо контекстное меню, появляющееся при нажатии правой кнопки мыши на закладке страницы.
По сути, многостраничный отчет представляет собой несколько отчетов в одном. На каждой странице может содержаться собственный отчет, со своим набором бэндов и настройками страницы.
Содержимое страницы может быть выведено на свободном месте предыдущего листа, если в настройках страницы включена опция "Печать на предыдущем листе" (свойство PrintToPrevPage в инспекторе).

Вложенный отчет

Вложенный отчет - это отчет с размещенным на нем объектом SubReport , который представляет собой ссылку на другой отчет, расположенный на отдельной странице. При формировании такого отчета вместо объекта SubReport будет выведен соответствующий отчет. При вставке объекта в отчет автоматически добавляется новая страница с названием "SubReport xx".
Объекты SubReport можно располагать на листе друг под другом или рядом. В случае, если объекты надо расположить друг под другом, их необходимо “разнести” по разным бэндам (например, первый отчет - в бэнде "master data", второй - в бэнде "child").
На использование SubReports накладываются следующие ограничения:

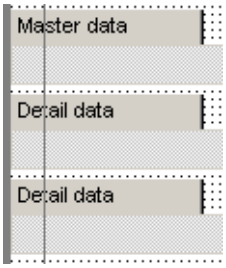
* нельзя использовать колонки;
* во вложенных отчетах (не в основном!) игнорируются бэнды ReportTitle, ReportSummary, PageHeader, PageFooter, ColumnXXX;

- нельзя использовать разрываемые бэнды;
- нельзя использовать группы.

Master-Detail-Detail отчет

Для построения этого отчета разместите на листе бэнд "master data" и два бэнда "detail data" (рис. 1.26). Для бэндов "detail data" назначьте соответствующие источники данных.

Рис. 1.26.



Разумеется, таким образом можно строить и отчеты типа Master-Master, Master-Detail-SubDetail-SubDetail. Отчеты типа Master-Detail-Master-Detail строить таким способом нельзя - нужно разнести разные master-detail по отдельным страницам (см. на стр. 37). Количество одинаковых дата-бэндов на листе может быть до 32.

Отчет без бэндов

Отчет может вообще не содержать бэндов. Объекты, расположенные непосредственно на листе, при печати выведутся на своих местах.

Отчет с группами

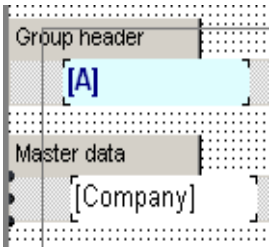
Группы применяются для группировки строк данных с использованием некоторого критерия. В качестве критерия обычно выступает выражение, основанное на полях из набора данных. При формировании отчета программа следит за значением критерия, и, как только оно изменится, формирует новую группу.

Для построения отчета разместите на листе бэнды "group header" и "master data". Назначьте источник данных для "master data", затем в редакторе бэнда "group header" укажите условие группировки. Например, чтобы распечатать список сотрудников в группированном виде (группировка по первой букве фамилии), надо указать примерно такое условие группировки:
Copy([Фамилия], 1, 1). При изменении этого условия будет сформирована новая группа.

На использование групп накладываются следующие ограничения:

- нельзя применять группы во вложенных отчетах;
- группа может выступать только в роли списка 1-го уровня, т.е. сначала идет группа, затем - master, detail и т.п.

Рис. 1.27.



Исходный отчет

Сформированный отчет

Примечание. Для правильной работы групп исходный набор данных должен быть уже отсортирован по условию группировки.

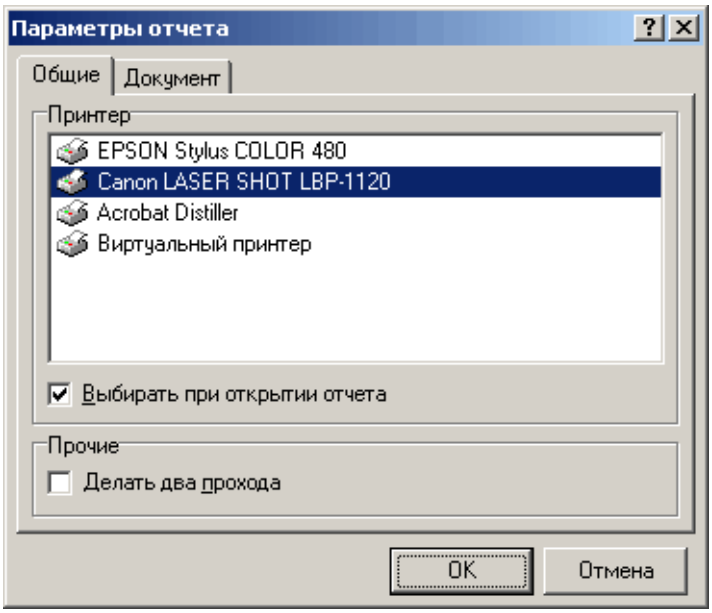
Отчет с диаграммами

Предположим, имеется отчет вида "Клиент - сумма заказа":
Для построения диаграммы по этим данным необходимо разместить объект "Диаграмма" в бэнде "report summary" (или любом другом, который печатается после вывода всех необходимых данных). В редакторе объекта нужно выбрать тип диаграммы (один из шести возможных - линии, точки, столбцы горизонтальные и вертикальные, области и круговые диаграммы). Для связывания диаграммы с данными необходимо указать имена объектов, которые содержат эти данные (в нашем случае это объект, содержащий строку наименования клиента и объект с суммой). Имя объекта можно увидеть в инспекторе объектов.

1.6 Параметры отчета

Окно **Параметры отчета** (рис. 1.28) доступно из меню ***Файл > Параметры отчета***.
В списке принтеров имеется "Виртуальный принтер". Он необходим в случае, если в системе нет установленных принтеров. При этом дизайнер и предварительный просмотр отчета функционируют нормально. Вы можете выбрать любой из форматов бумаги, поддерживаемой Windows (вплоть до формата A2), устанавливать ориентацию листа бумаги. Т.е. "виртуальный" принтер эмулирует функции реального принтера (разумеется, на нем нельзя печатать).
Другой случай, когда необходимо использовать виртуальный принтер - Вы разрабатываете форму отчета для принтера с форматом бумаги A3, а самого принтера у Вас нет. В этом случае выбор виртуального принтера снимает проблему.

Рис. 1.28.



Флажок **Выбирать при открытии отчета** позволяет привязать отчет к одному из принтеров, установленных в системе. Это значит, что при открытии такого отчета принтером по умолчанию станет выбранный принтер. Это полезно в случае, если в системе имеется несколько разных принтеров - текстовые

документы можно привязать к монохромному принтеру, документы с графикой - к цветному.

Если флажок **Делать два прохода** установлен, формирование отчета будет осуществляться в два этапа. На первом проходе отчет формируется, осуществляется его разбивка на страницы, но результат нигде не сохраняется. На втором проходе происходит обычное формирование отчета с сохранением результата в потоке.

Для чего нужно делать два прохода? Наиболее часто эта опция применяется в случае, если в отчете имеется упоминание об общем количестве страниц в нем, т.е. информация вида "Страница 1 из 15". Общее количество страниц подсчитывается на первом проходе и доступно через системную переменную **TOTALPAGES** (см. Руководство пользователя, раздел Языковые средства на стр. 59). Наиболее частая ошибка - попытка применить эту переменную в однопроходном отчете, в этом случае она возвращает 0.

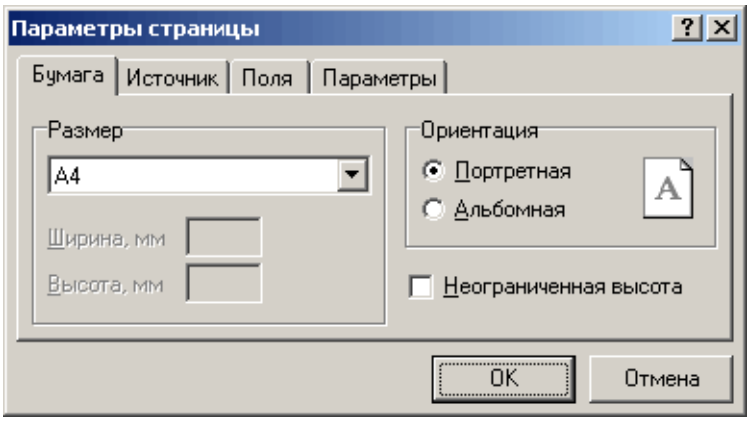
Другая область применения - выполнение каких-либо вычислений на первом проходе и отображение результатов на втором. Например, в случае, когда необходимо отобразить в заголовке группы сумму, которая обычно подсчитывается и отображается в подвале группы.

Параметры страницы

Окно с параметрами страницы доступно из меню ***Файл>Параметры страницы...*** либо при двойном щелчке левой кнопкой мыши на свободном месте листа отчета. Окно представляет собой блокнот с четырьмя закладками: "Бумага" (рис. 1.29), "Источник" (рис. 1.30), "Поля" (рис. 1.31) и "Параметры" (рис. 1.32).

Закладка "Бумага"

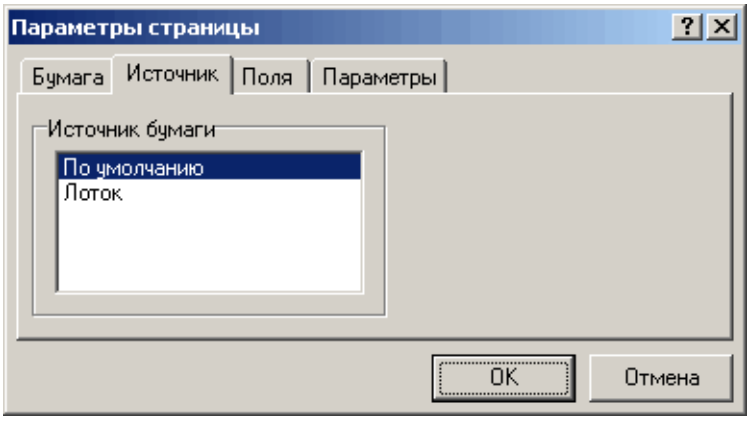
Рис. 1.29.



Поля ввода ширины и высоты становятся доступными, если из списка "Размер" выбран пункт "Размеры, определяемые пользователем". Следует отметить, что не все драйвера принтеров поддерживают пользовательский размер бумаги. Кроме того, не все принтеры поддерживают произвольные пользовательские размеры (например, драйвер принтера HP LaserJet 6L не позволяет устанавливать размеры листа меньше чем 76 * 127мм).

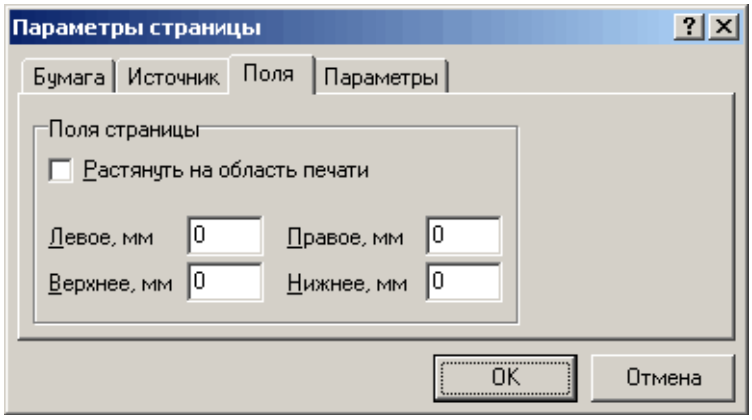
Закладка "Источник"

Рис. 1.30.



Закладка "Поля"

Рис. 1.31.



Здесь можно задать размеры отступа от краев листа, а также опцию “Растянуть на область печати”. Рассмотрим этот момент более подробно. Возможны два варианта страницы в дизайнере: с полями печати и без них. Если поля печати включены (по умолчанию), то они отображаются в дизайнере и все, что находится вне полей, на печать не выводится. Это позволяет добиться максимального соответствия между видом страницы в дизайнере и на бумаге. Координаты объектов исчисляются от верхнего левого угла листа.

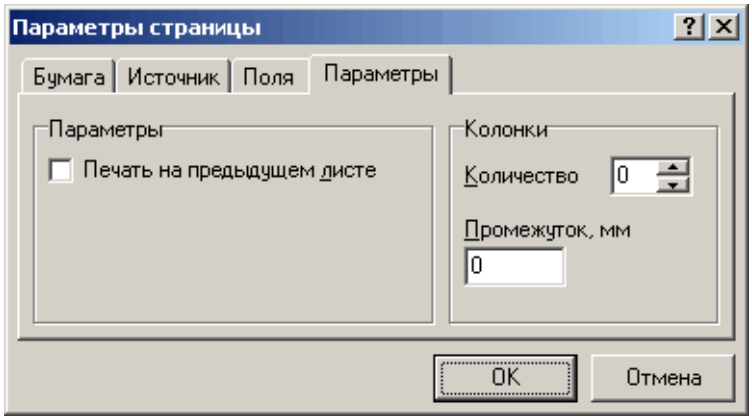
Но при переходе на другой принтер из-за изменившихся границ печати некоторые объекты могут вылезти за пределы полей и не будут напечатаны. Если включить флажок "Растянуть на область печати", то предполагается, что страница в дизайнере соответствует области печати, т.е. все, что находится на странице, будет корректно напечатано на любом принтере. Координаты объектов в этом случае исчисляются от верхнего левого угла области печати. Правда, при этом вид страницы в дизайнере будет немного отличаться от распечатанного документа.

Если поля страницы нулевые, то при построении отчета будет использована информация о полях, предоставляемая драйвером принтера. Если Вы используете матричный принтер, будьте внимательны. Дело в том, что во многих матричных принтерах одно или несколько полей имеет нулевой размер, в связи с чем при формировании отчета и при его распечатке может возникнуть ситуация, когда текст выходит за границы листа. Чтобы избежать

этого, явно задавайте поля станицы либо устанавливайте флажок " Растянуть на область печати".
Следует отметить, что на формирование отчета при включенных полях страницы оказывает непосредственное влияние только установка значений для верхнего и нижнего полей. Левое же и правое поля действуют только для Cross-tab отчетов (см. Cross-tab отчет на стр. 35).

Закладка "Параметры"

Рис. 1.32.



На этой закладке можно указать количество колонок для печати многоколоночных отчетов. Текущие установки отображаются в дизайнере. Флажок “Печатать на предыдущем листе” позволяет начать печать страницы, начиная со свободного места на предыдущем листе. Эта опция может применяться в случае, если шаблон отчета состоит из нескольких листов либо при печати составных (composite) отчетов.

Опции дизайнера

Чтобы установить опции дизайнера, воспользуйтесь командой меню ***Сервис>Опции...*** (рис. 1.33) и (рис. 1.34).

Рис. 1.33.

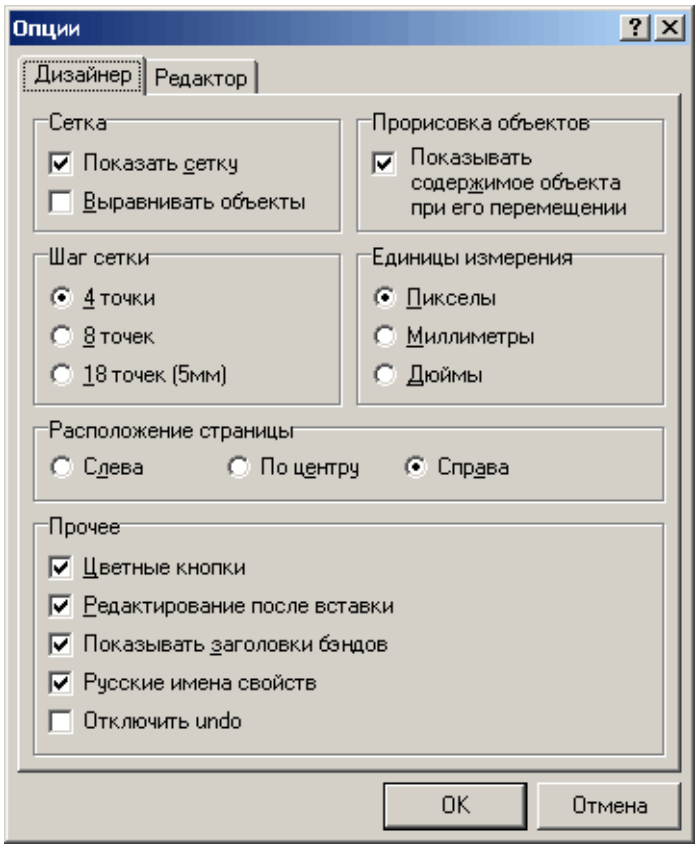
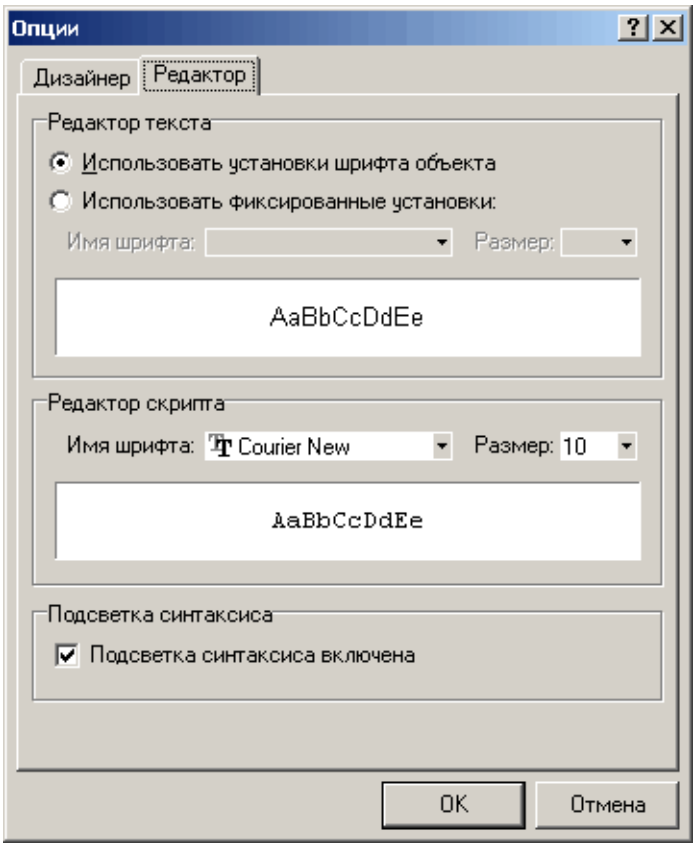


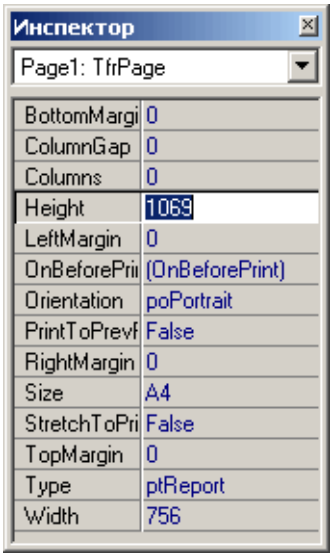
Рис. 1.34.



Здесь можно задать используемые единицы измерения (пикселы, миллиметры, дюймы), указать шаг сетки в точках (шаг сетки 18 точек точно соответствует 5мм. Это особо полезно в случаях, когда надо сделать бланк, размеры которого кратны 5мм.
Также можно указать способ прорисовки объектов при их перемещении (или изменения размеров).
Опция "Расположение страницы" позволяет выбрать расположение страницы. Это необходимо, если на экране присутствует инспектор объектов: таким образом можно задать взаимное расположение страницы и инспектора.
Если опция "Цветные кнопки" отключена, то все кнопки становятся черно-белыми; при наведении указателя мыши на кнопку она показывается в цвете.
Опция "Показывать заголовки бэндов" позволяет отключать заголовки у бэндов в целях экономии свободного места на странице.
Наконец, опция "Редактирование после вставки" управляет процессом вставки новых объектов. Если опция включена, каждый раз при вставке объекта будет показываться его редактор. При вставке большого количества пустых объектов опцию лучше отключать.
На закладке "Редактор" можно настроить шрифт для окна редактора текста. При включенной опции "Использовать установки шрифта объекта" шрифт в окне редактора текста будет соответствовать шрифту редактируемого объекта.

Инспектор объектов

Рис. 1.35.



Инспектор объектов позволяет манипулировать всеми свойствами выбранных объектов в окне дизайнера.

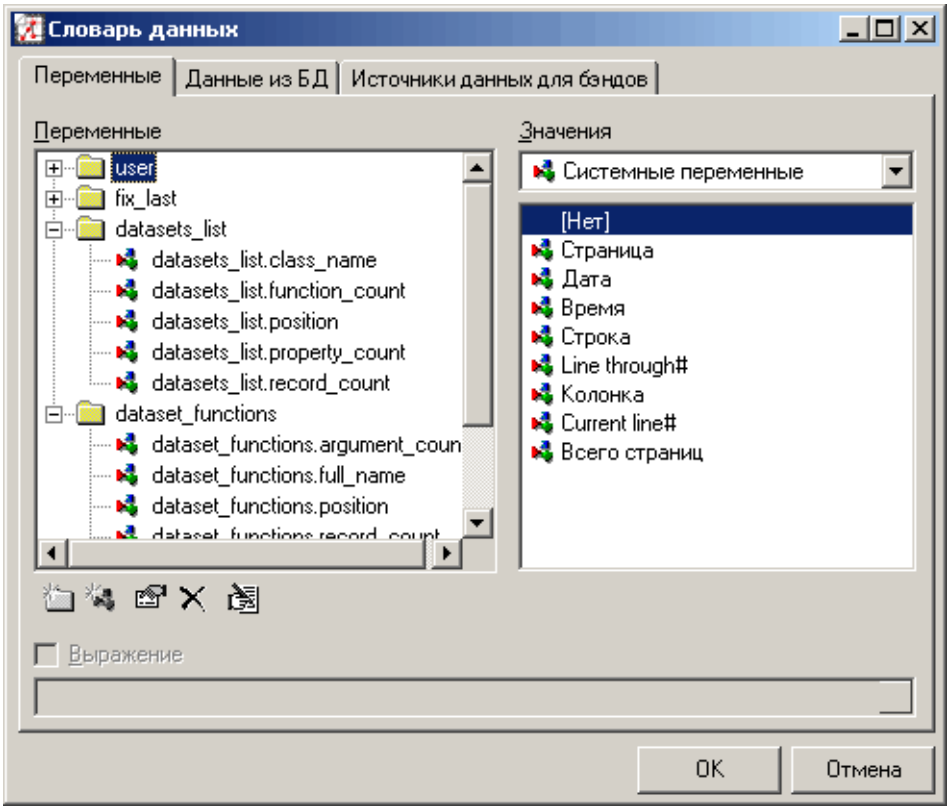
Так же, как и остальные панели инструментов, его можно показать или скрыть соответствующей командой меню ***Сервис>Панели инструментов***. Показать окно можно также при нажатии на клавишу ***F11***. Двойной щелчок на заголовке окна инспектора сворачивает окно в полоску, еще один двойной щелчок - восстанавливает исходные размеры.

Словарь данных

Окно словаря данных (Data dictionary) доступно из меню ***Файл>Словарь данных...***.
Окно представляет собой блокнот с тремя закладками: "Переменные", "Данные из БД" (в рамках работы с программой **VisiCAR™** не используется) и "Источники данных для бэндов".

Закладка "Переменные"

Рис. 1.36.



Эта закладка предназначена для работы со списком переменных (см. раздел Языковые средства на стр. 59.).
Список переменных расположен в левой части окна. Как видно из рисунка (рис. 1.36), структура списка двухуровневая: он состоит из источников данных, в каждом источнике данных может быть одна или несколько переменных. Такая структура нужны только для визуальной группировки переменных и в отчет не вставляются.
При создании нового отчета список пуст, и чтобы заполнить его, воспользуйтесь кнопками под списком.

Таблица 1.10: Работа с переменными

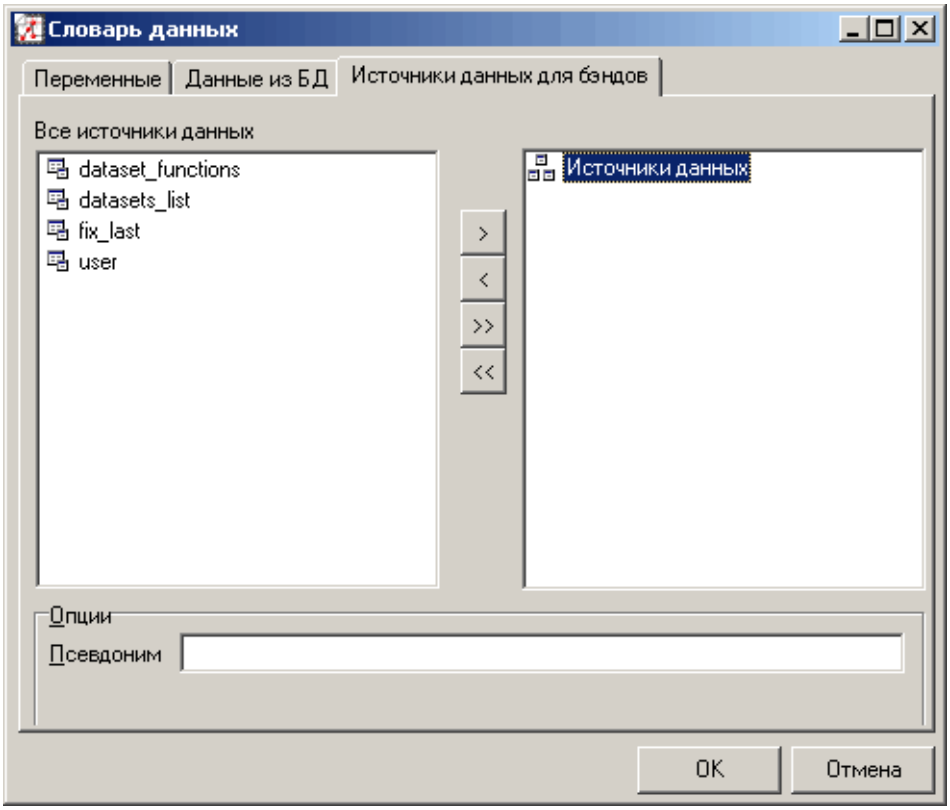
Иконка	*Описание*
	Добавляет новую категорию в перечень существующих (клавиатурный аналог - клавиша ***Ctrl+Insert***)
	Добавляет новую переменную к текущей категории (клавиатурный аналог - клавиша ***Insert***)
	Редактирует название переменной или категории (клавиатурный аналог - клавиша ***Enter***)
	Удаляет переменную или категорию (клавиатурный аналог - клавиша ***Delete***)
	Вызывает редактор списка переменных, где они представлены в виде списка строк

Здесь можно делать массовую вставку переменных из буфера обмена, переносить переменные из одной категории в другую.
После того как список переменных определен, каждой переменной необходимо назначить значение. Для этого нужно выбрать переменную в списке слева и выбрать соответствующее значение в списке справа, пользуясь мышью.
В списке справа показаны все источники данных, доступные в данный момент, и имена их полей. Кроме того, если выбрать из выпадающего списка справа значение «Системные переменные», то появится возможность сопоставить переменной одно из следующих значений:

- Страница - номер текущей страницы, соответствует функции Page# (см. раздел Описание встроенных процедур и функций, а также свойств и методов источников данных на стр. 64.);
- Дата - дата начала формирования отчета, соответствует Date;
- Время - время начала формирования отчета, соответствует Time;
- Строка, LineThrough#, Колонка, CurrentLine#, Всего страниц (см. раздел Описание встроенных процедур и функций, а также свойств и методов источников данных на стр. 64.).

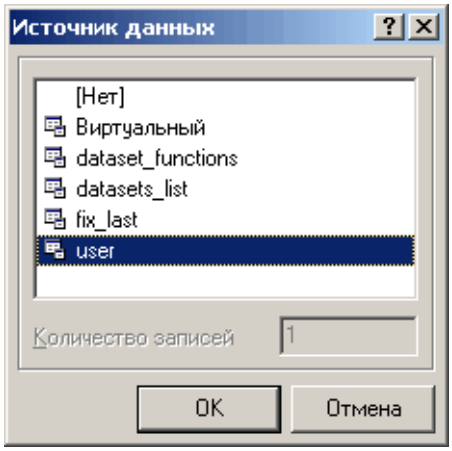
Переменной можно также сопоставить выражение (функцию) - для этого следует выбрать переменную и включить флажок "Выражение" в нижней части окна.
При этом строка ввода выражения становится активной. Для визуального построения выражения нажмите кнопку fx.
Окно вставки переменных в отчет для вышеприведенного примера будет выглядеть следующим образом (рис. 1.40).
Закладка "Источники данных для бэндов"

Рис. 1.37.



На этой закладке представлен список всех доступных источников данных для бэндов, имеющихся на всех формах проекта. Здесь можно давать элементам более осмысленные имена (псевдонимы). Результат можно увидеть в диалоге выбора источника данных для дата-бэнда (рис. 1.38).

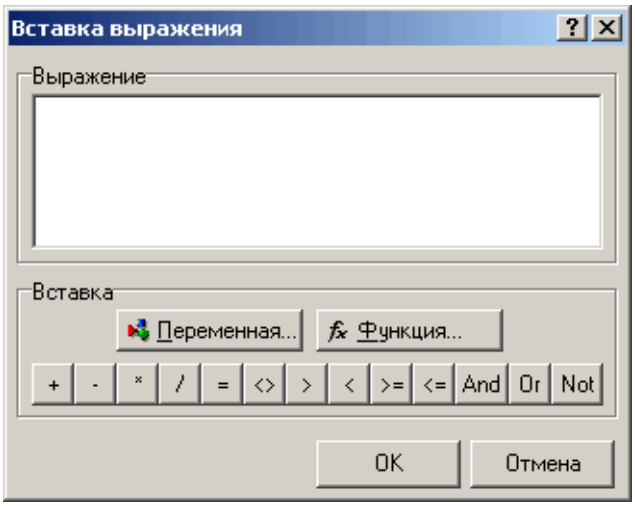
Рис. 1.38.



1.7 Конструктор выражений

Окно конструктора выражений можно вызвать из редактора текста, нажав кнопку  на панели инструментов. Оно также доступно из некоторых диалоговых окон, в которых требуется задать некоторое выражение (например, в редакторе условия группировки бэнда group header).

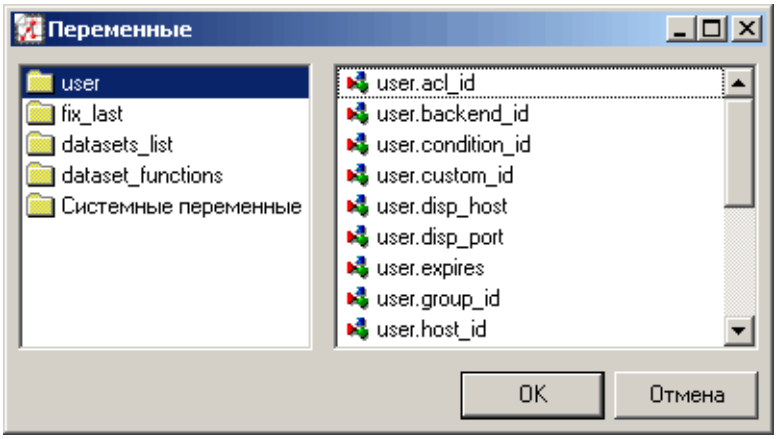
Рис. 1.39.



Окно содержит поле ввода выражения и кнопки вызова диалогов вставки переменных, полей БД (в рамках работы с программой **VisiCAR™** не используется), функций, а также кнопки быстрой вставки знаков арифметических и логических операций.

Диалог вставки переменной

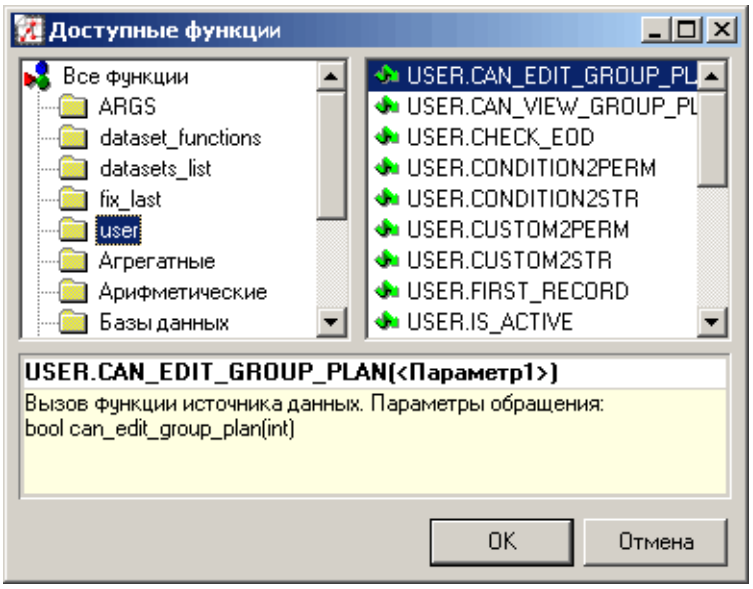
Рис. 1.40.



Диалог позволяет выбрать переменную для вставки в выражение. Слева размещен список источников данных, справа – список переменных в выбранном источнике. Для вставки переменной в выражение надо выбрать переменную и нажать кнопку **ОК**, либо сделать на ней двойной щелчок мышью.

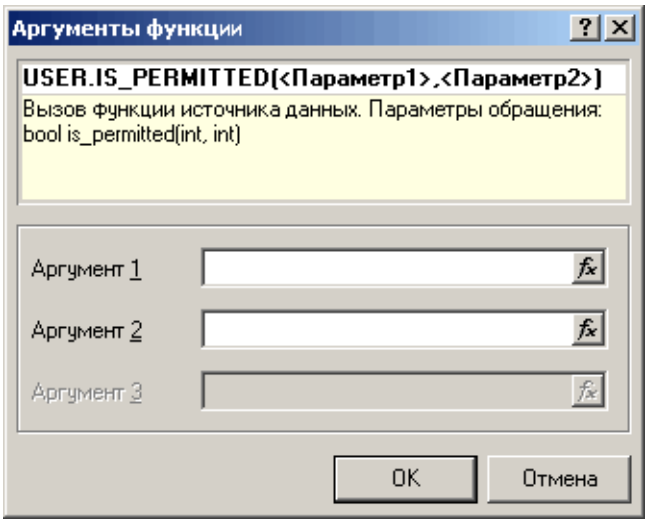
Диалог вставки функции

Рис. 1.41.



В этом диалоге можно выбрать функцию для вставки в выражение.
При выборе функции в нижней части окна отображается краткое описание функции и списка ее аргументов.
Если функция имеет аргументы, то при нажатии на кнопку **ОК** будет предложено заполнить их (рис. 1.42).

Рис. 1.42.



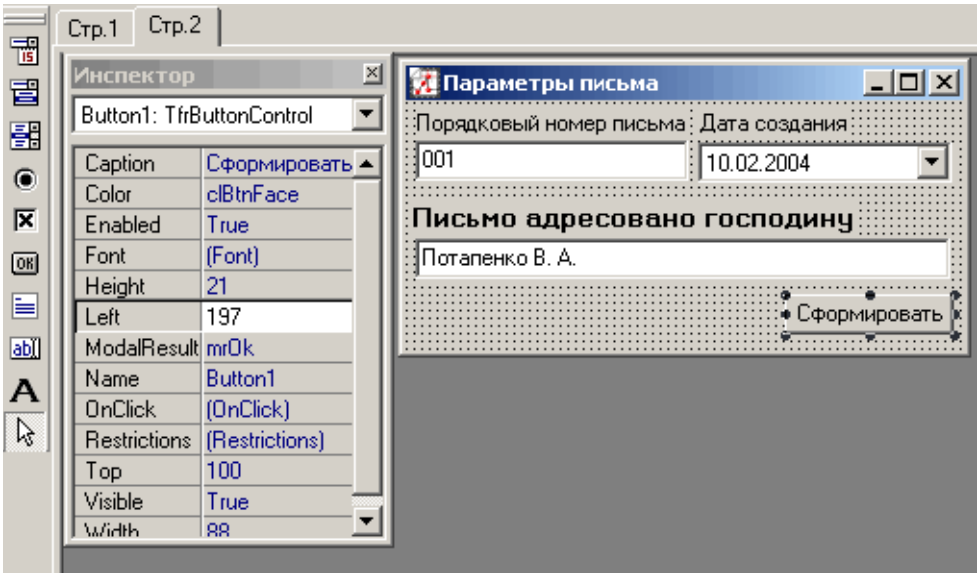
1.8 Диалоговые формы

На этапе построения отчета можно проектировать диалоговые формы. Эти формы будут появляться на экране в процессе непосредственной генерации отчета для указания параметров создаваемого отчета.
Для разработки диалоговых форм используется тот же дизайнер, что и для страниц отчета. Для создания новой формы служит кнопка панели инструментов дизайнера - она добавляет в отчет новую страницу, при переключении на которую меняется содержимое панели "Объекты":

Рис. 1.43.

При переключении на страницу с формой диалога рабочее поле дизайнера изменяется - теперь это форма, на которой можно размещать объекты - элементы управления:

Рис. 1.44.



В отчете может быть несколько диалоговых форм, они будут выполняться по порядку перед построением отчета.

Элементы управления (объекты)

Подключены следующие элементы управления:

Таблица 1.11: Элементы управления

Иконка	*Название*	*Описание*
	Label	Строка текста
	Edit	Предназначен для ввода строки текста с клавиатуры
	Memo	Предназначен для ввода нескольких строк текста
	Button	Кнопка
	CheckBox	Флажок
	RadioBut-ton	Круглая кнопка - аналог переключателя с зависимой фиксацией
	ListBox	Список строк предназначен для выбора одного из значений
	ComboBox	Выпадающий список строк
	DateEdit	Выпадающий список дат

Все объекты имеют ряд общих свойств и методов:

Общие свойства

* **Integer Color**
Цвет фона объекта
* **Boolean Enabled**
Разрешает или запрещает работу объекта
* **String Font.Name**
Имя шрифта, которым будет выведен текст объекта
* **Integer Font.Size**
Размер шрифта
* **Integer Font.Style**
Стиль шрифта. Набор констант fsBold, fsItalic, fsUnderline
* **Integer Font.Color**
Цвет шрифта
* **Integer Height**
Высота объекта
* **Integer Left**
Координата X левого верхнего угла объекта
* **String Name**
Имя объекта
* **Integer Top**
Координата Y левого верхнего угла объекта
* **Boolean Visible**
Видимость объекта
* **Integer Width**
Ширина объекта

Общие методы

* **Hide()**
Аналогично установке Visible := False
Параметры:
Не передаются.
Возвращаемые значения:
Нет.

---

* **SetFocus()**
Передает фокус ввода элементу управления
Параметры:
Не передаются.
Возвращаемые значения:
Нет.

---

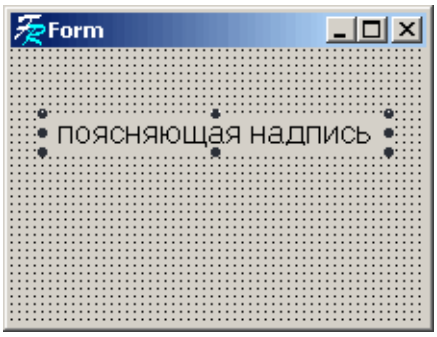
* **Show()**
Аналогично установке Visible := True
Параметры:
Не передаются.
Возвращаемые значения:
Нет.

Рассмотрим каждый объект в отдельности:

Label

Назначение: вывод поясняющей надписи

Рис. 1.45.



Свойства

- **Integer Alignment**
  Выравнивание строки текста внутри рамок объекта. Одно из значений taLeft-Justify, taRightJustify, taCenter.
- **Boolean AutoSize**
  Установка ширины объекта равной максимальной ширине имеющегося в нем текста.
- **String Caption**
  Собственно строка текста.
- **Boolean WordWrap**
  Перенос слов внутри объекта. При этом свойство AutoSize должно быть False.

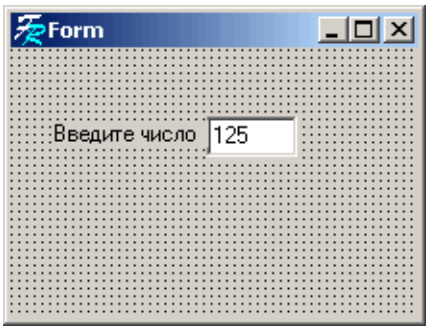
Методы.

Нет.

Edit

Назначение: ввод строки текста с клавиатуры

Рис. 1.46.



Свойства

- **Boolean ReadOnly**
  Запрещает модифицировать строку текста.

- **String Text**
  Текст, который будет показан в элементе управления по умолчанию.

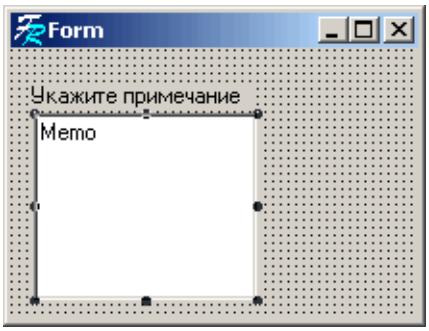
Методы

Нет.

Memo

Назначение: ввод нескольких строк текста с клавиатуры.

Рис. 1.47.



Свойства

- **String Lines**
  Строки текста, который будет выведен по умолчанию. К свойству можно обращаться по индексу: Memo1.Lines[0].
- **Integer Lines Count**
  Количество строк текста.
- **Boolean ReadOnly**
  Запрещает модифицировать текст.
- **String Text**
  Строки текста, представленные в виде одной строки (с символами CR+LF).

Методы

- **Lines.Add(String)**
  Добавляет строку.
  Параметры:
  String - добавляемая строка.
  Возвращаемые значения:
  Нет.

- **Lines.Clear()**
  Очищает строки.
  Параметры:
  String - добавляемая строка.
  Возвращаемые значения:
  Нет.

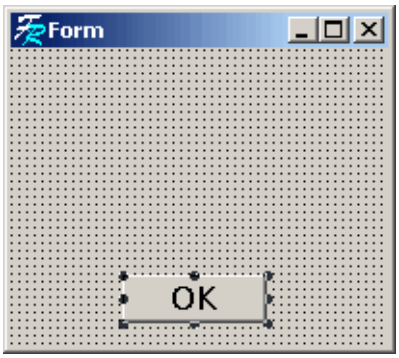
- **Lines.Delete(Integer)**
  Удаляет строку с данным индексом.
  Параметры:
  Integer - индекс удаляемой строки.

Возвращаемые значения:
Нет.

Button

Назначение:

Рис. 1.48.



Свойства

- **String Caption**
  Заголовок кнопки.
- **Integer ModalResult**
  Определяет, надо ли завершать работу диалогового окна при нажатии на кнопку и какой результат при этом возвращать. Предназначено для автоматического закрытия окна диалога при нажатии на кнопку. Обычно кнопка "OK" имеет ModalResult = mrOk, кнопка "Отмена" - mrCancel. Кроме того, аналогично свойство имеется у самой формы диалога. Если присвоить ему что-либо, отличное от mrNone, диалог завершит работу.

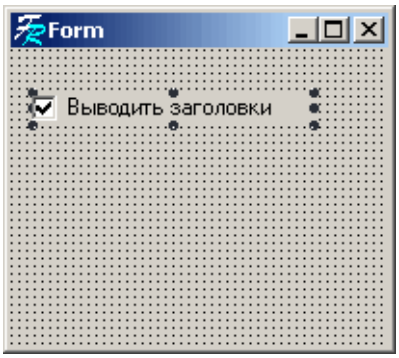
Методы

Нет.

CheckBox

Назначение: представляет собой флажок, который может быть в двух состояниях: включенном и выключенном. Около флажка выводится поясняющая надпись.

Рис. 1.49.



Свойства

- **Integer Alignment**
  Позиция надписи по отношению к флажку. Одна из констант taLeftJustify, taRightJustify.
- **String Caption**
  Поясняющая надпись.
- **Boolean Checked**
  Состояние флажка (включен/выключен).

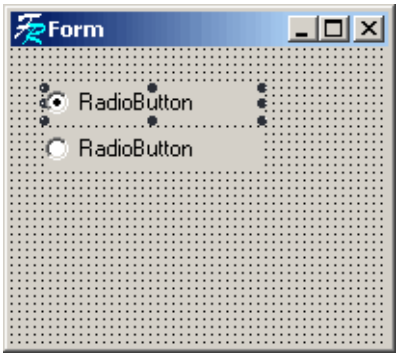
Методы

Нет.

RadioButton

Назначение: представляет собой аналог переключателя с зависимой фиксацией. По этой причине в одиночку не применяется.

Рис. 1.50.



Свойства

- **Integer Alignment**
  Позиция надписи по отношению к переключателю. Одна из констант taLeftJustify, taRightJustify.
- **String Caption**
  Поясняющая надпись.

* **Boolean Checked**
Состояние переключателя (включен/выключен).

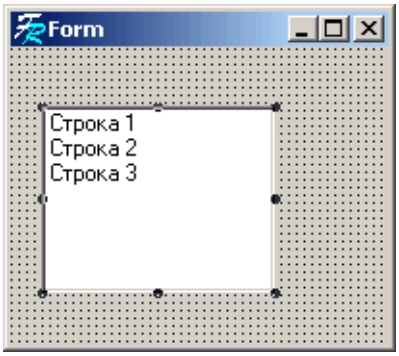
Методы

Нет.

ListBox

Назначение: представляет собой список строк с возможностью выбора одной из них.

Рис. 1.51.



Свойства

* **String Items**
Строки текста. К свойству можно обращаться по индексу: ListBox1.Items[0].
* **Integer ItemIndex**
Номер выбранной строки.
* **Integer Items.Count**
Количество строк.

Методы

* **Items.Add(String)**
Добавляет строку.
Параметры:
String - добавляемая строка.
Возвращаемые значения:
Нет.

---

* **Items.Clear()**
Очищает строки.
Параметры:
Не паредаются.
Возвращаемые значения:
Нет.

---

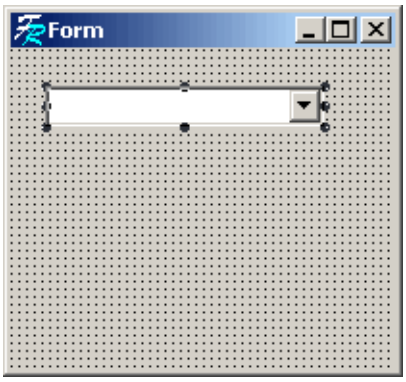
* **Items.Delete(Integer)**
Удаляет строку с данным индексом.
Параметры:
Integer - индекс удаляемой строки.
Возвращаемые значения:

Нет.

ComboBox

Назначение: представляет собой выпадающий список строк с возможностью выбора одной из них.

Рис. 1.52.



Свойства

- **String Items**
  Строки текста. К свойству можно обращаться по индексу: ComboBox1.Items[0].
- **Integer ItemIndex**
  Номер выбранной строки.
- **Integer Items.Count**
  Количество строк.
- **Integer Style**
  Тип элемента. Одно из значений csDropDown, csDropDownList, csLookup.
- **String Text**
  Выбранное значение.

Методы

- **Items.Add(String)**
  Добавляет строку.
  Параметры:
  String - добавляемая строка.
  Возвращаемые значения:
  Нет.

- **Items.Clear()**
  Очищает строки.
  Параметры:
  Не паредаются.
  Возвращаемые значения:
  Нет.

- **Items.Delete(Integer)**
  Удаляет строку с данным индексом.
  Параметры:

Integer - индекс удаляемой строки.

Возвращаемые значения:

Нет.

Диалоговая форма

Сама диалоговая форма также имеет набор свойств. Для того, чтобы увидеть свойства формы в инспекторе объектов, нужно щелкнуть мышкой на пустом месте формы, не занятом элементами управления.

Таблица 1.12: Свойства формы

Свойство	*Свойство по умолчанию*	*Описание*
Border-Style	bsDialog	Тип диалоговой формы - с фиксированными размерами либо с возможностью менять размеры окна с помощью мыши.
Caption	—	Заголовок окна
Color	clBtnFace	Цвет окна
Height	—	Высота окна
Left	—	Позиция X окна
OnActi-vate	—	Обработчик события, вызывающийся после инициализации всех элементов окна
Position	poScreen-Center	По центру экрана либо по заранее заданным координатам Left, Top
Top	—	Позиция Y окна
Type	ptDialog	Тип страницы - диалоговая форма либо страница отчета
Width	—	Ширина окна

Обработчик OnActivate вызывается, когда все элементы управления уже размещены на форме диалога и форма готова к выводу на экран. Этот обработчик удобно использовать для занесения каких-либо начальных значений в элементы управления, например, заполнения списка ListBox и пр.

Передача информации в отчет

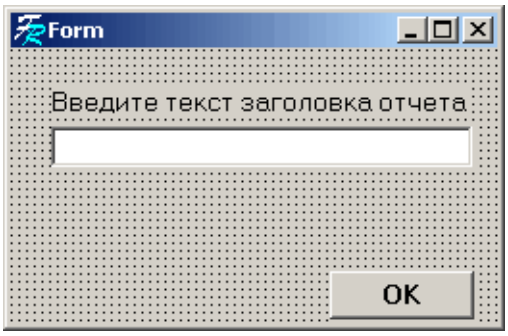
Значения, введенные в элементы управления, часто требуется тем или иным образом передать в отчет. Существует два способа это сделать.

С использованием переменных

Для передачи информации от элемента управления в отчет используются переменные.

Например, необходимо текст из элемента управления Edit отобразить в заголовке отчета.

Рис. 1.53.

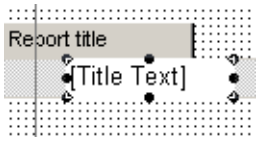


Это можно сделать следующим образом: в обработчике события OnClick кнопки OK написать:

```
begin
  TitleText := Edit1.Text;
end
```

В заголовок отчета при этом надо поместить объект "Текст" с содержимым [TitleText]:

Рис. 1.54.



Прямое обращение

Во многих случаях проще использовать для передачи введенного значения в отчет прямое обращение к элементу управления. При этом промежуточных переменных заводить не требуется:

Рис. 1.55.

1.9 Языковые средства

В состав программы входит интерпретатор Pascal-подобного языка. Данный интерпретатор является мощным средством, позволяющим писать скрипты, выполняющиеся при построении отчетов.

В качестве языка используется диалект Pascal, со следующими возможностями:

- операторы: присваивания; условные операторы, операторы циклов и безусловного перехода: if...then...else, while...do, repeat...until, for..to..do, goto;
- операторные скобки begin...end;
- безтиповые переменные, массивы;
- обращение к свойствам и методам объектов программы через точечную нотацию.

Язык по сравнению с Object Pascal сильно упрощен. Так, используются следующие упрощения:

- все переменные имеют тип Variant;
- отсутствует описание переменных;
- все переменные являются глобальными, нет локальных переменных;
- отсутствуют такие типы данных, как "класс", "запись", перечисляемый тип и т.п.;
- нет возможности написания собственных процедур или функций;
- нет операторов прерывания цикла (break, continue);
- количество параметров, передаваемых в процедуру или функцию, может быть не более 3 (для передачи боле чем 3 параметров необходимо использовать функцию *ARGS.Call(<function_name>,<array_name>)*. Передача параметров осуществляется через массив (см. Класс ARGS на стр. 68).);
- в связи с тем, что переменные не имеют типа, контроль типов данных невозможен — это следует учитывать при написании логических выражений;
- массивы могут быть только одномерными.

Несмотря на значительные упрощения, интерпретатор позволяет выполнять довольно сложную обработку информации. Из скрипта доступны все свойства и методы объектов отчета.
В скрипте можно создавать переменные и массивы, которые будут доступны во всем отчете. О возможностях встроенного языка говорит тот факт, что такая довольно сложная задача, как печать сумм группы в ее заголовке (сама сумма считается в подвале группы) средствами программы делается элементарно.
Соответствие типов данных мастера создания отчетов и Object Pascal рассмотрено ниже (табл. 1.13 на стр. 60):

Таблица 1.13: Соответствие типов данных

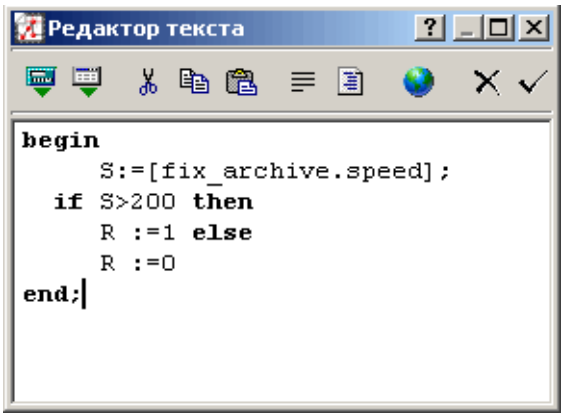
std::vector<...>	соответствует типизированному массиву (array[] of ...)
TPicture*	указатель, соответствует TPicture. Нельзя копировать как значение, только как указатель.
AnsiString	соответствует AnsiString
int	соответствует integer
double	соответствует double
void	метод такого типа ничего не возвращает, соответствует процедуре
TDataTime	соответствует TDataTime
bool	соответствует Boolean
unsigned int	соответствует Cardinal
Variant	соответствует Variant

Скрипты и объекты

В каждом объекте может быть размещен один или несколько (блок) операторов. Редактирование скрипта производится в окне редактора текста

(для того, чтобы увидеть скрипт, надо нажать кнопку в верхней части окна). Скрипт выполняется каждый раз перед печатью объекта.
По сути, скрипт является обработчиком события OnBeforePrint объекта.

Рис. 1.56.



Скрипт имеется не только у объектов. И бэнды, и страницы отчета также имеют свой скрипт. Вызвать скрипт бэнда можно, вызвав редактор свойства OnBeforePrint бэнда (из инспектора объектов, либо выделив бэнд и нажав Ctrl+Enter).
Вызвать скрипт страницы отчета можно, вызвав редактор свойства OnBeforePrint страницы (для этого надо щелкнуть на пустом месте страницы и вызвать редактор из инспектора объектов).
Диалоговая форма, как и страница отчета, имеет скрипт - свойство OnActivate.
Скрипт остальных объектов может быть вызван через их свойство Memo либо при нажатии Ctrl+Enter.

Написание кода

В скрипте можно использовать свойства и методы объектов отчета, различные константы. Также можно создавать переменные и массивы, которые будут доступны во всем отчете. Кроме того, можно использовать процедуры и функции.

Использование переменных

Переменную описывать не надо, она имеет тип Variant. В имени переменной можно использовать латинские буквы, цифры и знак подчеркивания.
Переменную из скрипта можно использовать в объектах, и наоборот, переменную из списка переменных можно использовать в скрипте. Можно обращаться к переменным, определенным в словаре данных, к системным переменным, а также пользовательским переменным. При этом переменные могут содержать в имени недопустимые с точки зрения синтаксиса символы (например, системная переменная Page#). Обращаться к таким переменным следует с использованием квадратных скобок:

```
begin
  a := [Page#];
end
```

Массивы

В скрипте, помимо переменных, можно создавать массивы. Массивы могут быть только одномерными, но можно организовать доступ к элементам

массива таким образом, что можно трактовать массив как двумерный или с большим количеством измерений.
Пример использования массива:

```
begin
  MyArr[0] := 'a'; MyArr[1] := 'b'; MyArr[3] := 'd';
  MyArr[2] := MyArr[0] + MyArr[1] + 'c' + MyArr[3];
end;
```

Т.е. в вышеприведенном примере мы получим:

```
Arr_MyArr_0 := 'a'
Arr_MyArr_1 := 'b'
Arr_MyArr_2 := 'abcd'
Arr_MyArr_3 := 'd'
```

Константы

В скрипте можно использовать константы. Простейший пример — использование числовых, строковых и логических констант:

```
begin
  a := 0;
  b := 'abcd';
  c := True;
  d := 'That''s all!';
end;
```

Обратите внимание на использование одинарной кавычки внутри строковой константы - как и в Pascal, ее надо дублировать: d := 'That''s all!'.
Кроме простых констант, в скрипте доступны константы - названия цветов, стилей шрифта и пр. Вот список констант, которые можно использовать:

* цвета: clWhite, clBlack и т.п. - все стандартные цвета + системные цвета;
* результат работы диалогового окна: mrNone, mrOk, mrCancel;
* служебные: CRLF, Null;
* стиль шрифта: fsBold, fsItalic, fsUnderline;
* рамка объекта: frftNone, frftRight, frftBottom, frftLeft, frftTop;
* выравнивание текста в объекте "Текст": frtaLeft, frtaRight, frtaCenter, frtaVertical, frtaMiddle, frtaDown;
* выравнивание по бэнду: baNone, baLeft, baRight, baCenter, baWidth, baBottom.

Кроме того, определены константы для подключаемых объектов, например, csCheck для объекта "CheckBoxObject". Все, что можно видеть в выпадающих списках свойств в инспекторе объектов, можно использовать как константу в скрипте.

Обращение к объектам

В скрипте можно обращаться к свойствам и методам объектов отчета. К объектам отчета относятся визуальные объекты, элементы управления, бэнды, страницы отчета, сам отчет. Для обращения к объекту применяется точечная нотация, например: Memo1.Text. Для обращения к собственным свойствам или методам точечная нотация необязательна.

Свойства, к которым возможно обращение, отображены в инспекторе объектов. К некоторым комплексным свойствам (например, Font) возможно обращение как к Font.Name, Font.Size и т.п.:

```
begin
  Memo1.Font.Name := 'Courier New';
  Memo1.Font.Size := 10;
  Memo1.Font.Color := clRed;
  Memo1.Font.Style := fsBold + fsItalic
end;
```

К свойствам типа TStrings (Memo, Items и пр.) возможен доступ по индексу:

```
if Memo1.Lines[1] = 'a' then
   Memo1.Lines[1] := 'b'
```

Также к таким свойствам могут применяться операции **Add**, **Delete**, **Clear** и **Count**:

```
if Memo1.Lines.Count > 10 then
   Memo1.Lines.Delete(10)
else
begin
  Memo1.Lines.Clear;
  Memo1.Lines.Add('a');
end;
```

Следует отметить, что обращение к несуществующему методу или свойству не приведет к сообщению об ошибке. Поэтому следует быть осторожным при написании кода.

Использование процедур и функций

В скрипте могут содержаться вызовы процедур и функций (см. раздел Описание встроенных процедур и функций, а также свойств и методов источников данных на стр. 64.).

При обращении к процедуре или функции нельзя ставить пробел между именем процедуры и открывающей скобкой.

Модификация объектов

В скрипте можно выполнить любые модификации объекта (либо объектов) - например, изменение размеров, цвета, содержимого и пр. Следует учесть, что объекты, которые уже обработаны, модифицированы быть не могут. То есть, попытка из объекта, лежащего на бэнде report summary, изменить содержимое объекта из report title, ни к чему не приведет. Однако, сделав отчет двухпроходным, мы сможем выполнить такую модификацию.
Аналогичным образом обстоят дела в случае многостраничного отчета. Мы можем обращаться к любому объекту по его имени (имя в пределах отчета уникально). Но модифицированы могут быть только объекты, которые еще не

обрабатывались. Если все же требуется модификация уже обработанного объекта, следует сделать отчет двухпроходным.

1.10 Описание встроенных процедур и функций, а также свойств и методов источников данных

На этапе проектирования отчетов есть возможность использовать различные процедуры , функции, методы, переменные и свойства для построения отчета. Переменные описывать не надо, они имеют тип Variant.
Переменную из скрипта можно использовать в объектах, и наоборот, переменную из списка переменных можно использовать в скрипте. Можно обращаться к переменным, определенным в словаре данных, к системным переменным, а также к свойствам (при использовании источников данных). В зависимости от выбранных источников данных будут доступны соответствующие свойства и методы. Переменные могут содержать в имени недопустимые с точки зрения синтаксиса символы (например, системная переменная Page#). Обращаться к таким переменным следует с использованием квадратных скобок:

Встроенные переменные и функции.

Агрегатные функции

Могут применяться в бэндах ReportSummary, PageFooter, MasterFooter, Detail-Footer, SubdetailFooter, GroupFooter, CrossFooter.

---

* **Sum(AnsiString [,AnsiString] [,1])**

Вычисление суммы.

Параметры:

AnsiString - выражение, сумму которого необходимо посчитать;
AnsiString - название бэнда (не обязательный параметр);

Возвращаемые значения:

Сумма выражения.

Комментарии:

Вычисляет сумму выражения для ряда строк данных. Если параметр [,AnsiString] не задан, то сумма считается по строке данных, соответствующей данному бэнду (по бэндам MasterData, DetailData, Sub-detailData); иначе сумма считается только по бэнду с именем [,AnsiString] . Если в расчет надо включить невидимые бэнды, надо указывать третий параметр (1).

Пример:

```
Sum([Part total], Band1);
Sum([[Part total] + [Part price]]);
Sum([Part total], Band1, 1).
```

---

* **Avg(AnsiString [,AnsiString] [,1])**

Вычисление среднего значения.

Параметры:

AnsiString - выражение для вычисления среднего значения;
AnsiString - название бэнда (не обязательный параметр);

Возвращаемые значения:

Среднее значение выражения.

Комментарии:

Работает аналогично функции Sum().

---

* **Min(AnsiString [,AnsiString] [,1])**

Вычисление минимума.

Параметры:

AnsiString - ряд для вычисления минимума;

AnsiString - название бэнда (не обязательный параметр);

Возвращаемые значения:

Минимум выражения.

Комментарии:

Работает аналогично функции Sum().

---

* **Max(AnsiString [,AnsiString] [,1])**

Вычисление максимума.

Параметры:

AnsiString - ряд для вычисления максимума;

AnsiString - название бэнда (не обязательный параметр);

Возвращаемые значения:

Минимум выражения.

Комментарии:

Работает аналогично функции Sum()

---

* **Count(AnsiString)**

Получить количество строк данных.

Параметры:

AnsiString - название бэнда;

Возвращаемые значения:

Количество строк данных..

Пример:

Count(Band1).

Функции работы со строками

* **Str(int)**

Преобразует число в строку.

Параметры:

int- число, которое необходимо преобразовать;

Возвращаемые значения:

Строка.

---

* **Copy(AnsiString, AnsiString, int)**

Возвращает подстроку .

Параметры:

AnsiString - строка, из которой необходимо выделить подстроку;

AnsiString - символ, с которого необходимо начать выделение подстроки (этот символ в подстроку не входит).

int- количество символов, которые необходимо выделить из строки.

Возвращаемые значения:

Выделенная подстрока.

Комментарии:

Из строки выделяется необходимая подстрока необходимой длинны.

---

* **If(AnsiString, AnsiString, AnsiString)**

Получить строку в зависимости от истинности или ложности условия.

Параметры:

AnsiString - строка условия, в зависимости от истинности или ложности которого будет получена соответствующая строка;
AnsiString - Строка1, которая будет получена, если условие истинно;
AnsiString - Строка2, которая будет получена, если условие ложно.

Возвращаемые значения:

Строка1 или Строка2.

---

* **FormatFloat(AnsiString, int)**

Преобразует числовое значение в строку.

Параметры:

AnsiString - строка форматирования.
int- число, которое необходимо отформатировать.

Возвращаемые значения:

Строка (отформатированное число).

Комментарии:

Преобразует числовое значение в строку, используя форматирование.

---

* **FormatDateTime(AnsiString , TDateTime)**

Преобразует дату/время.

Параметры:

AnsiString - строка форматирования.
TDateTime - значение даты и времени, которое необходимо отформатировать.

Возвращаемые значения:

Строка (отформатированное значение даты и времени).

Комментарии:

Преобразует дату/время в строку, используя форматирование.

---

* **StrToDate(AnsiString)**

Преобразует строку в дату.

Параметры:

AnsiString - строка, которую необходимо преобразовать.

Возвращаемые значения:

Дата.

Комментарии:

Преобразует строку в дату, используя стандартное форматирование.

---

* **StrToTime(AnsiString)**

Преобразует строку во время.

Параметры:

AnsiString - строка, которую необходимо преобразовать.

Возвращаемые значения:

Время.

Комментарии:

Преобразует строку во время, используя стандартное форматирование.

---

* **UpperCase(AnsiString)**

Преобразует символы строки в верхний регистр.

Параметры:

AnsiString - строка, которую необходимо преобразовать.

Возвращаемые значения:

Отформатированная строка.

---

* **LowerCase(AnsiString)**

Преобразует символы строки в нижний регистр.

Параметры:
AnsiString - строка, которую необходимо преобразовать.
Возвращаемые значения:
Отформатированная строка.

---

* **NameCase(AnsiString)**

Преобразует символы строки в нижний регистр, а первый символ - в верхний.
Параметры:
AnsiString - строка, которую необходимо преобразовать.
Возвращаемые значения:
Отформатированная строка.

---

* **Length(AnsiString)**

Возвращает длину строки.
Параметры:
AnsiString - строка, длину которой необходимо вычислить.
Возвращаемые значения:
Длинна строки.

---

* **Trim(AnsiString)**

Отбрасывает пробелы в начале и в конце строки и возвращает результат.
Параметры:
AnsiString - строка, которую необходимо преобразовать.
Возвращаемые значения:
Отформатированная строка.

---

* **Pos(AnsiString, AnsiString)**

Возвращает позицию подстроки в строке.
Параметры:
AnsiString - подстрока, позицию которой необходимо вычислить;
AnsiString - строка, позицию подстроки в которой необходимо вычислить.
Возвращаемые значения:
Позиция подстроки.

Арифметические функции

* **Int(Int)**

Возвращает целую часть числа value.
Параметры:
Int - число, целую часть которого необходимо вернуть;
Возвращаемые значения:
Целая часть числа.

---

* **Frac(Int)**

Возвращает дробную часть числа.
Параметры:
Int - число, дробную часть которого необходимо вернуть;
Возвращаемые значения:
Дробная часть числа.

---

* **Round(Int)**

Возвращает округленное значение.
Параметры:
Int - число, округленное значение которого необходимо вернуть;

Возвращаемые значения:

Округленное значение числа.

---

* **Int Mod Int**

Возвращает остаток от деления числа на число.

Параметры:

Int - делимое;

Int - делитель.

Возвращаемые значения:

Остаток от деления двух чисел.

---

* **MinNum(Int, Int)**

Возвращает меньшее из двух чисел.

Параметры:

Int - сравниваемое число1;

Int - сравниваемое число2.

Возвращаемые значения:

Меньше из двух чисел.

---

* **MaxNum(Int, Int)**

Возвращает большее из двух чисел.

Параметры:

Int - сравниваемое число1;

Int - сравниваемое число2.

Возвращаемые значения:

Большее из двух чисел.

Класс ARGS

В программе существует ограничение на количество параметров, передаваемых в процедуру или функцию — их не должно быть больше 3. Для передачи более 3 параметров (например, для метода **void crop(int,int,int,int)** источника данных **img_process**), необходимо использовать класс ***ARGS***.

Если в аргумент функции или метода можно вернуть какое-либо значение (на это указывает знак & после аргумента, например, метод **void world2pix(double &,double &)** источника данных) — параметры в такой метод также необходимо передавать через массив используя класс ***ARGS***.

Пример:

```
begin
 ARGS.clear('a'); //удаление данных массива
 ARGS.add('a',10); //формирование массива
 ARGS.add('a',20);
 ARGS.add('a',fix_archive.X);
 ARGS.add('a',fix_archive.Y);
 distance:=ARGS.call('MAP_PROJ.distance','a');
//вызываем метод измерения расстояния и передаем ей в качестве
параметров элементы массива “a” (4 параметра).
 end;
```

---

* **void add(<array_name>, <item_data>)**

Добавляет к массиву элемент данных.

Параметры:

array_name - имя массива, к которому необходимо добавить элемент;

item_data - элемент, который необходимо добавить.

Возвращаемые значения:

Нет.

---

* **Variant as_array(<array_name>)**

Возвращает массив с именем array_name.

Параметры:

array_name - имя массива.

Возвращаемые значения:

Массив с указанным именем.

---

* **Variant call(<function_name>,<array_name>)**

Вызывает функцию и передает ей в качестве аргумента массив.

Параметры:

function_name - имя функции;

array_name - имя массива.

Возвращаемые значения:

Массив с указанным именем.

Пример:

```
distance:=ARGS.call('MAP_PROJ.DISTANCE','a');
```

---

* **void clear(<array_name>)**

Удаляет все данные элементов массива.

Параметры:

array_name - имя массива.

Возвращаемые значения:

Нет.

---

* **unsigned int count(<array_name>)**

Возвращает количество элементов массива.

Параметры:

array_name - имя массива.

Возвращаемые значения:

Количество элементов массива.

---

* **void fill_Aray(<array_name>,<variant_array>)**

Создает массив с именем array_name и заполняет его элементами из массива variant_array.

Параметры:

array_name - имя создаваемого массива;

variant_array - имя массива, элементами которого будет заполнен новый массив.

Возвращаемые значения:

Нет.

---

* **Variant get_item(<array_name>, <item_index>)**

Возвращает значение элемента массива array_name с индексом item_index.

Параметры:

array_name - имя создаваемого массива;

item_index - индекс элемента (в массивах номерация начинается с нуля - 0, 1, 2 ...).

Возвращаемые значения:

Значение єлемента массива.

---

* **void put_item(<array_name>,<item_index>,<data>)**

Заменяет значение элемента массива array_name с индексом item_index на значение data.

Параметры:

array_name - имя массива;
item_index - индекс элемента (в массивах нумерация начинается с нуля - 0, 1, 2 ...);
data - значение.

Возвращаемые значения:

Нет.

Управление построением отчета

Переменные

* **int curY**
  Возвращает текущую позицию по Y (в пикселах), с которой будет выведен очередной бэнд. Этому свойству можно также присвоить нужное значение для смены позиции. Это позволяет строить довольно экзотические отчеты.
* **int freespace**
  Возвращает высоту оставшегося свободного места на странице в пикселах.
* **bool finalpass**
  Возвращает истину, если отчет двухпроходный и выполняется последний проход.
* **int pageheight**
  Возвращает высоту страницы в пикселах с учетом высоты бэнда Page footer.
* **int pagewidth**
  Возвращает ширину страницы в пикселах.

Функции

* **void stopreport()**
  Останавливает построение отчета.

---

* **void newpage()**
  Начинает формирование отчета с новой страницы.

---

* **void newcolumn()**
  Начинает формирование отчета с новой колонки, если отчет многоколоночный.

---

* **void showband(AnsiString)**

Показывает бэнд с указанным именем.

Параметры:

AnsiString - название бэнда.

Возвращаемые значения:

Нет.

---

* **void input(AnsiString [,AnsiString])**

Вывод на экран диалогового окна.

Параметры:

AnsiString - заголовок диалогового окна;
AnsiString - строка, появляющаяся в строке ввода (не обязательно).

Возвращаемые значения:

Нет.

Комментарии:

Выводит на экран диалоговое окно с указанным заголовком и строкой ввода. Если задан параметр [,AnsiString], то это значение появится в строке ввода. После ввода пользователя возвращает введенную строку.

---

* **TDataTime dayof(TDataTime)**

Возвращает день даты.

Параметры:

TDataTime - дата;

Возвращаемые значения:

День указанной даты.

---

* **TDataTime monthof(<date>)**

Возвращает месяц даты.

Параметры:

TDataTime - дата;

Возвращаемые значения:

Месяц указанной даты.

---

* **TDataTime yearof(<date>)**

Возвращает год даты.

Параметры:

TDataTime - дата;

Возвращаемые значения:

Год указанной даты.

---

* **int messagebox(AnsiString, AnsiString, <buttons_and_icons>)**

Выводит стандартное окно диалога с текстом и заголовком.

Параметры:

AnsiString - текст, который выводится в окне;

AnsiString - заголовок окна;

<buttons_and_icons> - набор кнопок и иконка окна

Возвращаемые значения:

Возвращает значение, соответствующее нажатой кнопке (mrOk, mrCancel, mrYes, mrNo)

Комментарии:

Набор кнопок и иконка окна задается в параметре <buttons_and_icons>. Возвращает значение, соответствующее нажатой кнопке (mrOk, mrCancel, mrYes, mrNo). В качестве <buttons_and_icons> можно использовать следующие значения:

Таблица 1.14: Значения <buttons_and_icons>

Кнопка	***Иконка***
mb_Ok	mb_IconError
mb_OkCancel	mb_IconQuestion
mb_YesNo	mb_IconInformation
mb_YesNoCancel	mb_IconWarning

Системные переменные

- **int line#**
  Возвращает номер строки данных; нумерация начинается с начала группы. Например:
  Master data
  1. Detail data
  2. Detail data
  3. Detail data
  Master data
  1. Detail data
  2. Detail data
- **int linethrough#**
  Возвращает номер строки данных; нумерация строк начинается с начала отчета, например:
  Master data
  1. Detail data
  2. Detail data
  3. Detail data
  Master data
  4. Detail data
  5. Detail data
- **int column#**
  Возвращает номер колонки в cross-tab отчете.
- **int current line#**
  Возвращает номер текущей строки отчета.
- **int Всего страниц**
  Возвращает общее количество страниц в сформированном отчете. Для использования этой переменной отчет должен быть двухпроходным.
- **TDataTime date (Дата)**
  Возвращает дату начала формирования отчета.
- **TDataTime time (Время)**
  Возвращает время начала формирования отчета.
- **int page# (Страница)**
  Возвращает номер текущей страницы.
- **int totalpages**
  Возвращает общее количество страниц в сформированном отчете. Для использования этой функции отчет должен быть двухпроходным (см. раздел Параметры отчета на стр. 39.).

Интерфейсы источников данных

Все методы и свойства определенных классов, описанные ниже, доступны для использования при добавлении в отчет соответствующего источника данных. Хранение информации ряда источников данных организовано в виде таблицы. Для получения информации о конкретном событии необходимо переместить “курсор” на нужную позицию, используя методы next_record(), first_record(), set_record. Источники данных:

- **current_fix_archive** (см. Источник данных: current_fix_archive на стр. 88);
- **dataset_functions** (см. Источник данных: dataset_functions на стр. 123);
- **dataset_props** (см. Источник данных: dataset_props на стр. 125);
- **datasets_list** (см. Источник данных: datasets_list на стр. 128);
- **events_archive** (см. Источник данных: events_archive на стр. 73);
- **fix_archive** (см. Источник данных: fix_archive на стр. 78);

- **fix_current** (см. Источник данных: fix_current на стр. 83);
- **fix_last** (см. Источник данных: fix_last на стр. 97);
- **poll_error** (см. Источник данных: poll_error на стр. 110);
- **virtual_dataset** (см. Источник данных: virtual_dataset на стр. 127);
- **table_<имя таблицы>** (см. Источник данных: table_<имя таблицы> на стр. 134);
- **user** (см. Источник данных: user на стр. 151);
- **binary_version** (см. Источник данных: binary_version на стр. 164);
- **plugin_version** (см. Источник данных: plugin_version на стр. 161);
- **objinf_window** (см. Источник данных: objinf_window на стр. 160);
- **distance_lines** (см. Источник данных: distance_lines на стр. 157)
- **map_zone** (см. Источник данных: map_zone на стр. 166)
- **any_table** (см. Источник данных: any_table на стр. 150).

Источник данных: car_pj

Источник данных, использующийся для пересчета координат мобильного объекта из одной проекции в другую. Это необходимо при перемещении мобильного объекта из карты в одной проекции на карту в другой проекции (например, из одной в области в другую - если карты этих областей соответствуют разным проекциям).

Свойства

* **MapProj* pj**
  Возвращает указатель на проекцию, в которой представлены мобильные объекты. Используется для пересчета координат положения мобильного объекта при изменении проекции карты.
  Параметры не передаются.

Методы

Нет

Источник данных: events_archive

Источник данных архива событий. Используя свойства и методы этого источника можно получить доступ к информации о событиях, произошедших в прошлом. Хранение данных источника организовано в виде таблицы.

Свойства

* **AnsiString add_info**
  Возвращает дополнительную информацию о произошедшем событии.
* **int cond_id**
  Возвращает идентификатор произошедшего события.
* **AnsiString condition**
  Возвращает название произошедшего события.
* **double double_time**
  Возвращает дату когда произошло событие.
* **int obj_id**
  Возвращает идентификатор мобильного объекта.
* **AnsiString obj_name**
  Возвращает имя мобильного объекта.
* **TPicture* picture**
  Возвращает пиктограмму, используемую для отображения мобильного объекта на карте.

* **TDateTime time**
  Возвращает дату когда произошло событие.
* **unsigned int position**
  Возвращает номер текущей записи.
* **unsigned int record_count**
  Возвращает общее количество записей в выборке.
* **std::vector<int> request_events**
  Возвращает массив идентификаторов событий, архив которых можно просмотреть.
* **std::vector<int> request_objects**
  Возвращает массив идентификаторов мобильных объектов, указанных в запросе (объекты, по которым формировалась выборка из архива).
* **TDateTime request_time_from**
  Возвращает время начала просмотра архива, указанное в запросе.
* **TDateTime request_time_to**
  Возвращает время окончания просмотра архива, указанное в запросе.

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет, достигнуто ли окончание потока данных и, в зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

---

* **void close()**

Закрыть соединения с сервером.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Прекратить получение данных по данному запросу от сервера. При этом в источнике данных будут доступны только те записи, которые были получены до вызова close(). Их количество можно получить функцией loaded(). Проверить получена ли текущая запись можно функцией is_data_present(). Нет необходимости вызывать close() после получения всех данных, т.к. в этом случае поток данных закрывается самостоятельно.

Смотри также:

is_data_present()
loaded()

---

* **AnsiString condition2str(int)**

Получить строку события

Параметры:

int - идентификатор события

Возвращаемые значения:

Событие в виде строки.

---

* **void default_param(TDateTime &,TDateTime &,std::vector<int> &,std::vector<int> &)**

Получение параметров источников данных

Параметры:

TDateTime - время “с” (начало периода просмотра). “0” означает, что время начала просмотра не ограничено;

TDateTime - время “до” (конец просмотра). “0” означает, что время окончания просмотра не ограничено;

std::vector<int> - массив идентификаторов мобильных объектов;

std::vector<int> - массив идентификаторов событий, архив которых необходимо просмотреть.

Полученный результат будет сохранен в этих переменных.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет получить параметры экземпляра источника данных events_archive, установленные по умолчанию. Используется для передачи этих параметров при инициализации экземпляра источника данных из программного кода.

Смотри также:

init()

---

* **void default_param(idataset_ctrl*,Variant &)**

Получение параметров по умолчанию источников данных.

Параметры:

*idataset_ctrl** - 0 (указатель на "интерфейс управления источником данных". Зарезервировано. Должен быть 0);

Variant & - ссылка на переменную, в которую будет записаны параметры по умолчанию источника данных.

Структура возвращаемого значение имеет вид массива: 0-ой элемент – время "с" (“0” означает, что время начала просмотра не ограничено), 1-ый – время "до" (“0” означает, что время окончания просмотра не ограничено), 2-й – массив идентификаторов мобильных объектов, 3-й – массив идентификаторов событий. Пустой массив означает все события.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет получить параметры экземпляра источника данных events_archive, установленные по умолчанию. Используется для передачи этих параметров при инициализации экземпляра источника данных из программного кода.

Смотри также:

init()

---

* **void first_record()**

Переместиться на первую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” в таблице источника данных на первую запись.

Смотри также:

next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.

Смотри также:

first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.

Параметры:

unsigned int - номер записи, на которую необходимо переместить “курсор”.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на необходимую запись в таблице источника данных.

Смотри также:

next_record()
first_record()

---

* **const AnsiString & get_error()**

Возвращает ошибку в виде строки.

Параметры:

Не передаются.

Возвращаемые значения:

Произошедшая ошибка в виде строки.

Комментарии:

Если в процессе выполнения отчета возникла ошибка – функция вернет эту ошибку в виде строки.

---

* **AnsiString get_object_name(int)**

Получить название мобильного объекта.

Параметры:

int - идентификатор мобильного объекта.

Возвращаемые значения:

Название мобильного объекта.

Комментарии:

Позволяет получить название мобильного объекта по его идентификатору.

---

* **void init(const TDateTime &,const TDateTime &,const std::vector<int> &,std::vector<int> &)**

Инициализация экземпляра источника данных.

Параметры:

TDateTime - время “с” (начало периода просмотра архива событий);
TDateTime - время “до” (конец просмотра архива событий);
std::vector<int> - массив идентификаторов мобильных объектов;
std::vector<int> - массив событий, архив которых необходимо просмотреть.

Возвращаемые значения:

Нет.

Комментарии:

Если при проектировании отчета некоторые источники данных не были инициализированы - это можно выполнить из программного кода, используя данный метод. Типовые значения параметров можно получить, вызвав метод default_param(). Для указания параметров пользователем используйте show_dialog().

Смотри также:

default_param()
show_dialog()

---

* **bool is_active()**

Проверка доступности источника данных.

Параметры:

Не передаются.

Возвращаемые значения:

True - если источник данных доступен (даже если он не инициализирован);
False - если источник данных не доступен.

Комментарии:

Проверяет доступен ли источник данных в данном отчете.

---

* **bool is_data_present()**

Проверка получения записи..

Параметры:

Не передаются.

Возвращаемые значения:

True - если необходимая запись получена;
False - если необходимая запись не получена.

Комментарии:

Процесс получения записей от сервера занимает какое-то время.
Если экземпляр источника данных инициализируется до выполнения отчёта (если он отмечен в списке источников которые подлежат инициализации), то к моменту выполнения отчёта все записи будут получены.
Если инициализировать источник данных самостоятельно необходимо следить за наличием записи с помощью функции is_data_present(). Чтобы дождаться получения всех записей используйте функцию synchronize().

Смотри также:

loaded()
synchronize()

---

* **unsigned int loaded()**

Возвращает количество загруженных записей.

Параметры:

Не передаются.

Возвращаемые значения:

Число загруженных записей.

Комментарии:

Возвращает количество полученных записей с сервера. Записи загружаются последовательно.

Смотри также:

is_data_present()

---

* **bool show_dialog(TDateTime &,TDateTime &,std::vector<int> &,std::vector<int> &)**

Отображает диалоговое окно инициализации источника данных.

Параметры:

TDateTime - время “с” (начало периода просмотра архива событий);
TDateTime - время “до” (конец просмотра архива событий);
std::vector<int> - массив идентификаторов мобильных объектов;
std::vector<int> - массив событий, архив которых просматривается.
В указанных параметрах передаются значения, отображаемые при создании диалогового окна. Полученный результат будет сохранен в этих переменных.

Возвращаемые значения:

True - если диалоговое окно создано;
False - если возникала ошибка.

Комментарии:

Этот метод используется для передачи значений, выбранных пользователем, в отчет. Как правило, используется для источников, не инициализированных при создании отчета (см. раздел Создание и удаление отчета на стр. 7.). Полученные параметры передаются в метод init().

Смотри также:

default_param()
init()

---

* **void synchronize(idataset_ctrl*)**

Синхронизация.

Параметры:

*idataset_ctrl** - 0.

Возвращаемые значения:

Нет.

Комментарии:

Используется для получения всех необходимых данных после инициализации источника данных из программного кода.

Смотри также:

init()

Источник данных: fix_archive

Позволяет получить доступ к архиву перемещений мобильных объектов. Хранение данных источника организовано в виде таблицы.

Свойства

- **double X**
  Возвращает координату X местоположения мобильного объекта.
- **double Y**
  Возвращает координату Y местоположения мобильного объекта.
- **double dX**
  Одна из составляющих для получения направления движения мобильного объекта (смещение вдоль оси X; $dX^2+dY^2=1$).
- **double dY**
  Одна из составляющих для получения направления движения мобильного объекта (смещение вдоль оси Y; $dX^2+dY^2=1$).
- **double double_time**
  Возвращает время фиксации координат местоположения мобильного объекта (в виде числа).
- **int obj_id**
  Возвращает идентификатор мобильного объекта.
- **AnsiString obj_name**
  Возвращает название мобильного объекта.
- **double speed**
  Возвращает скорость движения мобильного объекта.
- **TDateTime time**
  Возвращает дату фиксации координат местоположения мобильного объекта.
- **unsigned int position()**
  Возвращает номер текущей записи в выборке.
- **unsigned int record_count**
  Возвращает общее количество записей в выборке.
- **std::vector<int> request_objects**
  Возвращает массив идентификаторов мобильных объектов, указанных в запросе (объекты, по которым формировалась выборка из архива).
- **TDateTime request_time_from**
  Возвращает время начала просмотра архива, указанное в запросе.
- **TDateTime request_time_to**
  Возвращает время окончания просмотра архива, указанное в запросе.

Методы

- **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет, достигнуто ли окончание потока данных и, в зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

- **void close()**

Закрыть соединения с сервером.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Прекратить получение данных по данному запросу от сервера. При этом в источнике данных будут доступны только те записи, которые были получены до вызова close(). Их количество можно получить функцией loaded(). Проверить получена ли текущая запись можно функцией is_data_present(). Нет необходимости вызывать close() после получения всех данных, т.к. в этом случае поток данных закрывается самостоятельно.

Смотри также:

is_data_present()
loaded()

---

* **void default_param(TDateTime &,TDateTime &,std::vector<int> &)**

Получение параметров источника данных.

Параметры:

TDateTime - время “с” (начало просмотра архива). “0” означает, что время начала просмотра не ограничено;
TDateTime - время “до” (конец просмотра архива).“0” означает, что время окончания просмотра не ограничено;
std::vector<int> - массив идентификаторов мобильных объектов. Полученный результат будет сохранен в этих переменных.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет получить параметры, установленные по умолчанию для экземпляра источника данных.Эта функция используется для передачи параметров источнику данных при инициализации его из программного кода функцией init().

Смотри также:

init()

---

* **void default_param(idataset_ctrl*,Variant &)**

Получить параметры источника данных.

Параметры:

*idataset_ctrl** - 0 (указатель на "интерфейс управления источником данных". Зарезервировано. Должен быть 0.);
Variant & – ссылка на переменную, в которую будет записаны параметры по умолчанию источника данных. Структура возвращаемого значения имеет вид массива: 0 элемент – время "с", 1 – время "до", 2 – массив идентификаторов мобильных объектов.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет получить параметры, установленные по умолчанию для экземпляра источника данных.

Смотри также:

init()

---

* **void first_record()**

Переместиться на первую запись.

Параметры:

Не передаются.

Возвращаемые значения:
Нет.
Комментарии:
Позволяет переместить “курсор” на первую запись таблице источника данных.
Смотри также:
next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись.
Параметры:
Не передаются.
Возвращаемые значения:
Нет.
Комментарии:
Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.
Смотри также:
first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.
Параметры:
unsigned int - число, указывающее номер записи, на которую необходимо переместиться.
Возвращаемые значения:
Нет.
Комментарии:
Позволяет переместить “курсор” на необходимую запись в таблице источника данных.
Смотри также:
next_record()
first_record()

---

* **const AnsiString & get_error()**

Возвращает ошибку в виде строки.
Параметры:
Не передаются.
Возвращаемые значения:
Произошедшая ошибка в виде строки.
Комментарии:
Если в процессе выполнения отчета возникла ошибка – функция вернет эту ошибку в виде строки.

---

* **AnsiString get_object_name(int)**

Получить название мобильного объекта.
Параметры:
int - идентификатор мобильного объекта.
Возвращаемые значения:
Название мобильного объекта.
Комментарии:

Позволяет получить название мобильного объекта по его идентификатору.

---

* **void init(const TDateTime &,const TDateTime &,const std::vector<int> &)**

Инициализация экземпляра источника данных.

Параметры:

TDateTime - время “с” (начало периода просмотра);
TDateTime - время “до” (конец просмотра);
std::vector<int> - массив идентификаторов мобильных объектов.

Возвращаемые значения:

Нет.

Комментарии:

Если при проектировании отчета некоторые источники данных не были инициализированы - это можно выполнить из программного кода, используя данный метод. Типовые значения параметров можно получить, вызвав метод default_param(). Для указания параметров пользователем используйте show_dialog()

Смотри также:

default_param()
show_dialog()

---

* **bool is_active()**

Проверка доступности источника данных.

Параметры:

Не передаются.

Возвращаемые значения:

True - если источник данных доступен (даже если он не инициализирован);
False - если источник данных не доступен.

Комментарии:

Проверяет доступен ли источник данных в данном отчете.

---

* **bool is_data_present()**

Проверка получения записи.

Параметры:

Не передаются.

Возвращаемые значения:

True - если необходимая запись получена;
False - если необходимая запись не получена.

Комментарии:

Процесс получения записей от сервера занимает какое-то время.
Если экземпляр источника данных инициализируется до выполнения отчёта (если он отмечен в списке источников которые подлежат инициализации), то к моменту выполнения отчёта все записи будут получены.
Если инициализировать источник данных самостоятельно необходимо следить за наличием записи с помощью функции is_data_present().
Чтобы дождаться получения всех записей используйте функцию synchronize()

Смотри также:

loaded()
synchronize()

---

* **unsigned int loaded()**

Возвращает количество загруженных записей.

Параметры:

Не передаются.

Возвращаемые значения:

Число загруженных записей.

Комментарии:

Возвращает количество полученных записей с сервера. Записи загружаются последовательно.

Смотри также:

is_data_present()
synchronize()

---

* **bool show_dialog(TDateTime &,TDateTime &,std::vector<int> &)**

Отображает диалоговое окно инициализации источника данных.

Параметры:

TDateTime - время “с” (начало периода просмотра архива);
TDateTime - время “до” (конец просмотра архива);
std::vector<int> - массив идентификаторов мобильных объектов;
В указанных параметрах передаются значения, отображаемые при создании диалогового окна. Полученный результат будет сохранен в этих переменных.

Возвращаемые значения:

True - если диалоговое окно создано;
False - если возникала ошибка.

Комментарии:

Этот метод используется для передачи значений, выбранных пользователем, в отчет. Как правило, используется для источников, не инициализированных при создании отчета (см. раздел Создание и удаление отчета на стр. 7.). Полученные параметры передаются в метод init().

Смотри также:

default_param()
init()

---

* **void synchronize(idataset_ctrl*)**

Синхронизация.

Параметры:

*idataset_ctrl** - 0.

Возвращаемые значения:

Нет.

Комментарии:

Используется для получения всех необходимых данных после инициализации источника данных из программного кода.

Смотри также:

init()

Источник данных: fix_current

Источник данных текущего положения мобильного объекта на открытой карте (перемещение объекта, просмотр архива). Позволяет получить доступ к данным объектов, отображенных на карте. Хранение информации источника организовано в виде таблицы.

Свойства

- **double X**
  Возвращает координату X местоположения мобильного объекта.
- **double Y**
  Возвращает координату Y местоположения мобильного объекта.
- **double dX**
  Одна из составляющих для получения направления движения мобильного объекта (смещение вдоль оси X; $dX^2+dY^2=1$).
- **double dY**
  Одна из составляющих для получения направления движения мобильного объекта (смещение вдоль оси Y; $dX^2+dY^2=1$).
- **double double_time**
  Возвращает время фиксации координат местоположения мобильного объекта.
- **bool inside_map**
  Возвращает True если мобильный объект находится на открытой карте или False в другом случае.
- **bool inside_picture**
  Возвращает True если мобильный объект находится на участке карты, отображенном в окне программы или False в другом случае.
- **bool is_auto_change_map**
  Возвращает True если включен режим автозамены карты или False в другом случае.
- **bool is_fix_map**
  Возвращает True если включен режим перемещения карты за выделенными объектами или False в другом случае.
- **bool is_show_hint**
  Возвращает True если включен режим отображения подсказок для мобильных объектов или False в другом случае.
- **bool is_trace_mode**
  Возвращает True если включен режим отображения трассы перемещения мобильных объектов или False в другом случае.
- **int obj_id**
  Возвращает идентификатор мобильного объекта.
- **AnsiString obj_name**
  Возвращает название мобильного объекта.
- **double speed**
  Возвращает скорость движения мобильного объекта.
- **TDateTime time**
  Возвращает время фиксации координат местоположения мобильного объекта.
- **unsigned int position()**
  Возвращает номер текущей записи в выборке.
- **unsigned int record_count**
  Возвращает общее количество записей в выборке.
- **std::vector<int> request_objects**
  Возвращает массив идентификаторов мобильных объектов, указанных в запросе (объекты, по которым формировалась выборка из архива).
- **TDateTime request_time_from**
  Возвращает время начала просмотра архива, указанное в запросе.

* **TDateTime request_time_to**
  Возвращает время окончания просмотра архива, указанное в запросе.
* **int sym_request**
  Возращает идентификатор запроса к серверу мобильных объектов, ассоциированного с активным окном карты. (например "последнее местоположение","просмотр архива","наблюдение за объектами"). Список доступных запросов и их названия можно получить в источнике данных user.

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет, достигнуто ли окончание потока данных и, в зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

---

* **bool is_active()**

Проверка доступности источника данных.

Параметры:

Не передаются.

Возвращаемые значения:

True - если источник данных доступен (даже если он не инициализирован);
False - если источник данных не доступен.

Комментарии:

Проверяет доступен ли источник данных в данном отчете.

---

* **void first_record()**

Переместиться на первую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на первую запись таблицы источника данных.

Смотри также:

next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.

Смотри также:

first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.

Параметры:

unsigned int - число, указывающее номер записи, на которую необходимо переместиться.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на нужную запись в таблице источника данных.

Смотри также:

next_record()
first_record()

---

* **void fix_map(int)**

Режим перемещения карты за объектом

Параметры:

int - идентификатор мобильного объекта..

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить режим перемещения карты за мобильным объектом. Работает аналогично кнопке  [Д: перемещать карту за объектами]).

Смотри также:

is_fix_map

---

* **void fix_map(const std::vector<int> &)**

Режим перемещения карты за объектами.

Параметры:

const std::vector<int> & - массив идентификаторов мобильных объектов, за которыми необходимо перемещать карту.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить режим перемещения карты за несколькими мобильными объектами одновременно. Работает аналогично кнопке  [Д: перемещать карту за объектами])

Смотри также:

is_fix_map

---

* **void move_map(int)**

Отображение мобильного объекта в открытом окне.

Параметры:

int - идентификатор мобильного объекта.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет изменить масштаб карты, чтоб мобильный объект был отображен в открытом окне (на видимой области). Работает аналогично кнопке  [Д: показать выделенные объекты вместе]

Смотри также:

inside_map
inside_picture

---

* **void move_map(const std::vector<int> &)**

Отображение объектов на одной карте.

Параметры:

const std::vector<int> & - массив идентификаторов мобильных объектов.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет изменить масштаб карты, чтоб несколько мобильных объектов были одновременно отображены на одной карте (на видимой области в отрытом окне). Работает аналогично кнопке  [Д: показать выделенные объекты вместе]

Смотри также:

inside_map
inside_picture

---

* **void select_obj(int)**

Выбор мобильного объекта.

Параметры:

int - идентификатор мобильного объекта

Возвращаемые значения:

Нет.

Комментарии:

Позволяет выбрать (выделить) необходимый мобильный объект

---

* **void select_obj(const std::vector<int> &)**

Выбор нескольких объектов.

Параметры:

const std::vector<int> & - массив идентификаторов мобильных объектов.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет выбрать (выделить) одновременно несколько необходимых мобильных объектов.

---

* **void set_auto_change_map(bool)**

Автоматическая замена карты.

Параметры:

True - передается для установления режима замены карты;
False - передается для отмены режима замены карты.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить режим автоматического изменения карты ( [Д:автоматическая замена карты])для определения необходимой карты.

Смотри также:

is_auto_change_map

---

* **void set_show_hint(bool)**

Отображение подсказок.

Параметры:

True - передается для отображения подсказок на карте;

False - передается для запрета режима отображения подсказок на карте.

Возвращаемые значения:

Нет.

Комментарии:

Разрешает или запрещает отображение подсказок для мобильных объектов.

Смотри также:

is_show_hint

---

* **void set_trace_mode(bool)**

Отображение трассы.

Параметры:

True - передается для отображения трассы передвижений мобильного объетка на карте;

False - передается для запрета режима отображения трассы на карте.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет устанавливать или отменять режим отображения трассы передвижения мобильного объекта (аналогично кнопке  [Д: отображать трассу]).

Смотри также:

is_trace_mode

Источник данных: current_fix_archive

Источник данных, предоставляющий информацию о текущем положении мобильного объекта на открытой карте при просмотре архива. Используя переменные и методы этого источника можно получить доступ к архиву перемещений мобильных объектов. Хранение данных источника организовано в виде таблицы.

Свойства

* **double X**

Возвращает координату X местоположения мобильного объекта.

* **double Y**

Возвращает координату Y местоположения мобильного объекта.

* **double dX**

Одна из составляющих для получения направления движения мобильного объекта (смещение вдоль оси X; $dX^2+dY^2=1$).

* **double dY**

Одна из составляющих для получения направления движения мобильного объекта (смещение вдоль оси Y; $dX^2+dY^2=1$).

* **double double_time**

Возвращает время фиксации координат местоположения мобильного объекта.

- **bool inside_map**
  Возвращает True если мобильный объект находится на открытой карте или False в другом случае.
- **bool inside_picture**
  Возвращает True если мобильный объект находится на участке карты, отображенном в окне программы или False в другом случае.
- **bool is_auto_change_map**
  Возвращает True если включен режим автозамены карты или False в другом случае.
- **bool is_bof**
  Возвращает True если находимся в начале просмотра архива или False в другом случае.
- **bool is_eof**
  Возвращает True если находимся в конце просмотра архива или False в другом случае.
- **bool is_fix_map**
  Возвращает True если включен режим перемещения карты за выделенными объектами или False в другом случае.
- **bool is_show_hint**
  Возвращает True если включен режим отображения подсказок для мобильных объектов или False в другом случае.
- **bool is_trace_mode**
  Возвращает True если включен режим отображения трассы перемещения мобильных объектов или False в другом случае**.**
- **int obj_id**
  Возвращает идентификатор мобильного объекта.
- **AnsiString obj_name**
  Возвращает название мобильного объекта.
- **bool pause**
  Возвращает True если включен “режим паузы” или False в другом случае.
- **double speed**
  Возвращает скорость движения мобильного объекта.
- **TDateTime time**
  Возвращает дату фиксации координат местоположения мобильного объекта.
- **int total_count**
  Возвращает общее количество координат местоположений объектов в архиве.
- **unsigned int position()**
  Возвращает номер текущей записи в выборке.
- **unsigned int record_count**
  Возвращает общее количество записей в выборке.
- **std::vector<int> request_objects**
  Возвращает массив идентификаторов мобильных объектов, указанных в запросе (объекты, по которым формировалась выборка из архива).
- **TDateTime request_time_from**
  Возвращает время начала просмотра архива, указанное в запросе.
- **TDateTime request_time_to**
  Возвращает время окончания просмотра архива, указанное в запросе.
- **int sym_request**
  Возращает идентификатор запроса к серверу мобильных объектов,

ассоциированого с активным окном карты. Всегда соответствует запросу "просмотр архива". Список доступных запросов и их названия можно получить в источнике данных user.

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет достигнуто ли окончание потока данных или нет и, в зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

---

* **bool is_active()**

Проверка доступности источника данных.

Параметры:

Не передаются.

Возвращаемые значения:

True - если источник данных доступен (даже если он не инициализирован);
False - если источник данных не доступен.

Комментарии:

Проверяет доступен ли источник данных в данном отчете.

---

* **void first_record()**

Переместиться на первую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на первую запись таблицы источника данных.

Смотри также:

next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.

Смотри также:

first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.

Параметры:

unsigned int - число, указывающее номер записи, на которую необходимо переместиться.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на необходимую запись в таблице источника данных.

Смотри также:

first_record()
next_record()

---

* **void fix_map(int)**

Режим перемещения карты за объектом

Параметры:

int - идентификатор мобильного объекта..

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить режим перемещения карты за мобильным объектом. Работает аналогично кнопке  [Д: перемещать карту за объектами]).

Смотри также:

is_fix_map

---

* **void fix_map(const std::vector<int> &)**

Режим перемещения карты за объектами.

Параметры:

const std::vector<int> & - массив идентификаторов мобильных объектов, за которыми необходимо перемещать карту.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить режим перемещения карты за несколькими мобильными объектами одновременно. Работает аналогично кнопке  [Д: перемещать карту за объектами])

Смотри также:

is_fix_map

---

* **void go_end()**

Перейти в положение последней координаты объекта.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместиться в положение последней координаты мобильного объекта.

Перемещение в конец просмотра архива координат. Соответствует кнопке  в окне **Управление просмотром** (см. раздел Работа с

архивом на стр. 28.). Перемещение в конец осуществляется путём получения последнего блока данных с сервера мобильных объектов. Координаты объектов присутствующие в промежуточных блоков останутся неизвестными. Следствием этого может быть искажённая картина трасс и неверное определение последнего местоположения объектов. Координаты получается блоками по 512 записей.

Смотри также:

go_first()
go_next()
go_preview()
total_count

---

* **void go_first()**

Перейти в положение первой координаты объекта.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Перемещение в начало просмотра архива координат. Соответствует кнопке  в окне **Управление просмотром** (см. раздел Работа с архивом на стр. 28.).

Смотри также:

go_end()
go_next()
go_preview()
total_count

---

* **void go_next()**

Перейти в положение следующей координаты объекта.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Перемещение в следующую позицию. Соответствует кнопке  в окне **Управление просмотром** (см. раздел Работа с архивом на стр. 28.). В состоянии паузы перемещение осуществляется к следующей координате с большим временем. В состоянии просмотра перемещение осуществляется к следующей координате с временем большим на delta. Величина delta определяется в зависимости от скорости просмотра. Количество итераций go_next() может быть меньше величины current_fix_archive.total_count

Смотри также:

go_end()
go_first()
go_preview()
total_count
is_eof

---

* **void go_preview()**

Перейти в положение предыдущей координаты объекта.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Перемещение в предыдущую позицию. Соответствует кнопке  в окне **Управление просмотром** (см. раздел Работа с архивом на стр. 28.) В состоянии паузы перемещение осуществляется к предыдущей координате с меньшем временем. В состоянии просмотра перемещение осуществляется к предыдущей координате с временем, меньшим на delta. Величина delta определяется в зависимости от скорости просмотра. Количество итераций go_preview() может быть меньше величины current_fix_archive.total_count.

Смотри также:

go_end()
go_first()
go_next()
total_count
is_bof

---

* **void move_map(int)**

Отображение мобильного объекта в открытом окне.

Параметры:

int - идентификатор мобильного объекта.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет изменить масштаб карты, чтоб мобильный объект был отображен в открытом окне (на видимой области). Работает аналогично кнопке  [Д: показать выделенные объекты вместе]

Смотри также:

inside_map
inside_picture

---

* **void move_map(const std::vector<int> &)**

Отображение объектов на одной карте.

Параметры:

const std::vector<int> & - массив идентификаторов мобильных объектов.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет изменить масштаб карты, чтоб несколько мобильных объектов были одновременно отображены на одной карте (на видимой области в отрытом окне). Работает аналогично кнопке  [Д: показать выделенные объекты вместе]

Смотри также:

inside_map
inside_picture

---

* **void select_obj(int)**

Выбор мобильного объекта.

Параметры:

int - идентификатор мобильного объекта

Возвращаемые значения:

Нет.

Комментарии:

Позволяет выбрать (выделить) необходимый мобильный объект

---

* **void select_obj(const std::vector<int> &)**

Выбор нескольких объектов.

Параметры:

const std::vector<int> & - массив идентификаторов мобильных объектов.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет выбрать (выделить) одновременно несколько необходимых мобильных объектов.

---

* **void set_auto_change_map(bool)**

Автоматическая замена карты.

Параметры:

True - передается для установления режима замены карты;
False - передается для отмены режима замены карты.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить режим автоматического изменения карты ( [Д: автоматическая замена карты])для определения необходимой карты.

Смотри также:

is_auto_change_map

---

* **void set_pause(bool)**

Пауза в просмотре архива.

Параметры:

True - передается для установления режима “паузы”;
False - передается для отмены режима “паузы”.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить или отменить режим “паузы” при просмотре архива.

---

* **void set_show_hint(bool)**

Отображение подсказок.

Параметры:

True - передается для отображения подсказок на карте;
False - передается для запрета режима отображения подсказок на карте.

Возвращаемые значения:

Нет.

Комментарии:

Разрешает или запрещает отображение подсказок для мобильных объектов.

Смотри также:

is_show_hint

---

* **void set_trace_mode(bool)**

Отображение трассы.

Параметры:

True - передается для отображения трассы передвижений мобильного объекта на карте;
False - передается для запрета режима отображения трассы на карте.

Возвращаемые значения:
Нет.

Комментарии:
Позволяет устанавливать или отменять режим отображения трассы передвижения мобильного объекта (аналогично кнопке  [Д: отображать трассу]).

Смотри также:
is_trace_mode

---

* **void switch_current()**

Переключение источника информации на текущее положение всех объектов.

Параметры:
Не передаются.

Возвращаемые значения:
Нет.

Комментарии:
В каждый момент просмотра архива существует текущее положение объектов, которые участвуют в запросе. switch_current() позволяет представить источник данных таким образом, чтобы он представлял собой таблицу текущего положения объектов.

---

* **void switch_current_one()**

Переключение источник информации на текущее положение мобильного объекта в данный момент времени.

Параметры:
Не передаются.

Возвращаемые значения:
Нет.

Комментарии:
Каждое положение просмотра архива, определяет только одна координата, с определённым временем. switch_current_one() позволяет представить источник данных таким образом, чтобы он представлял собой таблицу, длинною в одну запись, которая представляет собой текущее положение просмотра. Переключение в этот режим позволяет “перебрать” все координаты в архиве в хронологическом порядке.

---

* **void switch_end()**

Переключение источника информации на конечные положения всех мобильных объектов.

Параметры:
Не передаются.

Возвращаемые значения:
Нет.

Комментарии:
Позволяет представить источник данных в виде таблицы с последним положением всех мобильных объектов, участвующих в запросе. Положения соответствует символам **F** (см. раздел Работа с архивом на стр. 28.)при просмотре трассы.

---

* **void switch_first()**

Переключает источник информации на начальные положения всех мобильных объектов.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет представить источник данных в виде таблицы с первым положением всех мобильных объектов, участвующих в запросе. Положения соответствует символам **S** (см. раздел Работа с архивом на стр. 28.)при просмотре трассы.

---

* **void switch_objects()**

Переключает источник информации на все координаты местоположения мобильного объекта, находящиеся в памяти в данный момент.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

При просмотре архива на карте, только часть координат храниться в памяти. Остальная часть подкачивается с сервера по мере необходимости. switch_objects() позволяет представить источник данных в виде таблицы с теми положением объектов, которые находятся в данный момент в памяти. При просмотре архива с небольшим количеством координат это позволяет перебрать все координаты более естественным для FastReport путём.

---

* **bool wait_data_load()**

Ожидание необходимых данных.

Параметры:

Не передаются.

Возвращаемые значения:

True - если все необходимые данные получены;
False - если необходимые данные не получены.

Комментарии:

Используется при выполнении запроса для проверки получены ли все необходимые данные или нет. При использовании этой команды программа будет ждать получения необходимой информации до тех пор, пока данные не будут получены.

---

* **bool wait_data_load(unsigned int)**

Ожидание необходимых данных.

Параметры:

unsigned int - значение промежутка времени в миллисекундах.

Возвращаемые значения:

True - если все необходимые данные получены;
False - если необходимые данные не получены.

Комментарии:

Используется при выполнении запроса для проверки получены ли все необходимые данные или нет. При использовании этой команды программа будет ждать получения необходимой информации в течении указанного промежутка времени.

Источник данных: fix_last

Источник информации предоставляет данные о последнем положении мобильного объекта. Однако, после открытия архива последнее местоположение одного или нескольких объектов может быть не известно и методы источника ***fix_last*** будут работать некорректно. Для правильной работы необходимо вначале пройти весь архив или переместиться на последнюю позицию в архиве, а затем использовать методы и переменные. Хранение данных источника организовано в виде таблицы.

Свойства

* **double X**
  Возвращает координату X местоположения мобильного объекта.
* **double Y**
  Возвращает координату Y местоположения мобильного объекта.
* **double dX**
  Одна из составляющих для получения направления движения мобильного объекта (смещение вдоль оси X; $dX^2+dY^2=1$).
* **double dY**
  Одна из составляющих для получения направления движения мобильного объекта (смещение вдоль оси Y; $dX^2+dY^2=1$).
* **double double_time**
  Возвращает время фиксации координат местоположения мобильного объекта (в формате **double**).
* **int obj_id**
  Возвращает идентификатор мобильного объекта.
* **AnsiString obj_name**
  Возвращает название мобильного объекта.
* **double speed**
  Возвращает скорость движения мобильного объекта.
* **TDateTime time**
  Возвращает дату фиксации координат местоположения мобильного объекта.
* **unsigned int position()**
  Возвращает номер текущей записи в выборке.
* **unsigned int record_count**
  Возвращает общее количество записей в выборке.
* **std::vector<int> request_objects**
  Возвращает массив идентификаторов мобильных объектов, указанных в запросе (объекты, по которым формировалась выборка из архива).

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет, достигнуто ли окончание потока данных и, в зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

---

* **void close()**

Закрыть соединения с сервером.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Прекратить получение данных по данному запросу от сервера. При этом в источнике данных будут доступны только те записи, которые были получены до вызова close(). Их количество можно получить функцией loaded(). Проверить получена ли текущая запись можно функцией is_data_present(). Нет необходимости вызывать close() после получения всех данных, т.к. в этом случае поток данных закрывается самостоятельно.

Смотри также:

is_data_present()
loaded()

---

* **void default_param(idataset_ctrl*,Variant &)**

Получить параметры источника данных.

Параметры:

*idataset_ctrl** - 0 (указатель на "интерфейс управления источником данных". Зарезервировано. Должен быть 0.);
Variant - ссылка на переменную, в которую будет записаны параметры по умолчанию источника данных.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет получить параметры экземпляра источника данных, установленные по умолчанию. Используется для передачи этих параметров при инициализации экземпляра источника данных из программного кода.

Смотри также:

init()

---

* **void default_param(std::vector<int> &)**

Получить параметры источника данных.

Параметры:

std::vector<int> & - массив идентификаторов мобильных объектов. Полученный результат будет сохранен в этом массиве.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет получить параметры экземпляра источника данных, установленные по умолчанию. Используется для передачи этих параметров при инициализации экземпляра источника данных из программного кода.

Смотри также:

init()

---

* **void first_record()**

Переместиться на первую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на первую запись таблицы источника данных.

Смотри также:

next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.

Смотри также:

first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.

Параметры:

unsigned int - число, указывающее номер записи, на которую необходимо переместиться.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на необходимую запись в таблице источника данных.

Смотри также:

first_record()
next_record()

---

* **const AnsiString & get_error()**

Возвращает ошибку в виде строки.

Параметры:

Не передаются.

Возвращаемые значения:

Произошедшая ошибка в виде строки.

Комментарии:

Если в процессе выполнения отчета возникла ошибка – функция вернет эту ошибку в виде строки.

---

* **AnsiString get_object_name(int)**

Получить название мобильного объекта.

Параметры:

int - идентификатор мобильного объекта.

Возвращаемые значения:

Название мобильного объекта.

Комментарии:

Позволяет получить название мобильного объекта по его идентификатору.

---

* **void init(const std::vector<int> &)**

Инициализация экземпляра источника данных.

Параметры:

std::vector<int> - массив идентификаторов мобильных объектов.

Возвращаемые значения:

Нет.

Комментарии:

Если при проектировании отчета некоторые источники данных не были инициализированы – это можно выполнить из программного кода, используя данную функцию. Параметры для инициализации передаются при помощи метода default_param().

Смотри также:

default_param()

---

* **bool is_active()**

Проверка доступности источника данных.

Параметры:

Не передаются.

Возвращаемые значения:

True - если источник данных доступен (даже если он не инициализирован);
False - если источник данных не доступен.

Комментарии:

Проверяет доступен ли источник данных в данном отчете.

---

* **bool is_data_present()**

Проверка получения записи.

Параметры:

Не передаются.

Возвращаемые значения:

True - если необходимая запись получена;
False - если необходимая запись не получена.

Комментарии:

Процесс получения записей от сервера занимает какое-то время.
Если экземпляр источника данных инициализируется до выполнения отчёта (если он отмечен в списке источников, которые подлежат инициализации), то к моменту выполнения отчёта все записи будут получены.
Если инициализировать источник данных самостоятельно — необходимо следить за наличием записи с помощью метода is_data_present().Чтобы дождаться получения всех записей используйте метод synchronize().

Смотри также:

loaded()
synchronize()

---

* **unsigned int loaded()**

Возвращает количество загруженных записей.

Параметры:

Не передаются.

Возвращаемые значения:

Число загруженных записей.

Комментарии:

Возвращает количество полученных записей с сервера. Записи загружаются последовательно.

Смотри также:

is_data_present()
synchronize()

---

* **bool show_dialog(std::vector<int> &)**

Отображает диалоговое окно инициализации источника данных.

Параметры:

std::vector<int> - массив идентификаторов мобильных объектов. В указанных параметрах передаются значения, отображаемые при создании диалогового окна. Полученный результат будет сохранен в этих переменных.

Возвращаемые значения:

True - если диалоговое окно создано;
False - если возникла ошибка.

Комментарии:

Этот метод используется для передачи значений, выбранных пользователем, в отчет. Как правило, используется для источников, не инициализированных при создании отчета (см. раздел Создание и удаление отчета на стр. 7.). Полученные параметры передаются в метод init().

Смотри также:

init()
default_param()

---

* **void synchronize(idataset_ctrl*)**

Синхронизация.

Параметры:

*idataset_ctrl** - 0.

Возвращаемые значения:

Нет.

Комментарии:

Используется для получения всех необходимых данных после инициализации источника данных из программного кода.

Смотри также:

init()

Источник данных: map_proj

Источник предоставляет доступ к данным проекции открытой карты. Используется для пересчета точек из одной проекции в другую, вычисления расстояния между двумя точками и т.п.

Свойства

* **int distance_unit**

Возвращает установленные единицы измерения карты (метры - 0, километры - 1, мили - 2).

* **int distance_unit_count**

Возвращает общее количество доступных единиц измерения расстояния.

* **MapProj* pj**
  Возвращает указатель на установленную проекцию карты.

Методы

* **double distance(double,double,double,double)**

Измерение расстояния

Параметры:

double - x координата первой точки;
double - y координата первой точки;
double - x координата второй точки;
double - y координата второй точки.

Возвращаемые значения:

Расстояние между двумя точками.

Комментарии:

Измеряет расстояние между двумя точками, координаты которых заданы, как (x1,y1) и (x2,y2).

---

* **double fwdx(double,double)**

Пересчёт координаты из градусов в установленную проекцию.

Параметры:

double - долгота в формате <градусы>.<доли градусов> местоположения точки в градусах;
double - широта в формате <градусы>.<доли градусов> местоположения точки в градусах.

Возвращаемые значения:

x координату точки в установленной проекции.

Комментарии:

Преобразует координаты точки в градусах в координату X картографической проекции, установленной функцией set_pj().

---

* **double fwdx(MapProj*,double,double)**

Пересчёт координаты из указанной в установленную проекцию.

Параметры:

*MapProj** - указатель на картографическую проекцию, в которой записаны координаты точки;
double - долгота местоположения точки (в указанной проекции);
double - широта местоположения точки (в указанной проекции).

Возвращаемые значения:

x координату точки в установленной проекции.

Комментарии:

Преобразует координаты точки в определенной проекции карты (передается в параметрах) в координату X в проекции, установленной функцией set_pj().

---

* **double fwdy(double,double)**

Пересчёт координаты из градусов в установленную проекцию.

Параметры:

double - долгота в формате <градусы>.<доли градусов> местоположения точки в градусах;
double - широта в формате <градусы>.<доли градусов> местоположения точки в градусах.

Возвращаемые значения:

Y координату точки в установленной проекции.

Комментарии:

Преобразует координаты точки в градусах в координату Y проекции, установленной функцией set_pj().

---

* **double fwdy(MapProj*,double,double)**

Пересчёт координаты из указанной в установленную проекцию.

Параметры:

*MapProj** - указатель на проекцию карты, в которой записаны координаты точки;
double - долгота местоположения точки (в указанной проекции);
double - широта местоположения точки (в указанной проекции).

Возвращаемые значения:

у координату точки в установленной проекции.

Комментарии:

Преобразует координаты точки в определенной проекции карты (передается в параметрах) в координату Y в проекции, установленной функцией set_pj().

---

* **double invx(double,double)**

Пересчёт координаты из установленной проекции в градусы.

Параметры:

double - долгота местоположения точки;
double - широта местоположения точки.

Возвращаемые значения:

координата X точки в градусах.

Комментарии:

Преобразует координаты точки X в градусы относительно установленной проекции.

---

* **double invx(MapProj*,double,double)**

Пересчёт координаты из указанной в параметрах проекции в градусы.

Параметры:

*MapProj** - указатель на проекцию карты, в которой записаны координаты точки;
double - долгота местоположения точки;
double - широта местоположения точки.

Возвращаемые значения:

координата X точки в градусах.

Комментарии:

Преобразует координаты точки X в градусы относительно указанной в параметрах проекции.

---

* **double invy(double,double)**

Пересчёт координаты из установленной проекции в градусы.

Параметры:

double - долгота местоположения точки;
double - широта местоположения точки.

Возвращаемые значения:

координата Y точки в градусах.

Комментарии:

Преобразует координаты точки Y в градусы относительно установленной проекции.

---

* **double invy(MapProj*,double,double)**

Пересчёт координаты из указанной в параметрах проекции в градусы.

Параметры:

*MapProj** - указатель на проекцию карты, в которой записаны координаты точки;
double - долгота местоположения точки;
double - широта местоположения точки.

Возвращаемые значения:

координата Y точки в градусах.

Комментарии:

Преобразует координаты точки Y в градусы относительно указанной в параметрах проекции.

---

* **bool is_equal(double,double)**

Сравнение двух чисел.

Параметры:

double - сравниваемое число;
double - сравниваемое число.

Возвращаемые значения:

True - если числа равны;
False - если числа не равны

Комментарии:

Сравнивает два числа, например координаты. Сравнение проводится в пределах одной проекции карты, т.е. числа должны быть записаны в координатах одной и той же проекции.

---

* **void set_pj(MapProj*)**

Установка проекции карты.

Параметры:

*MapProj** - указатель на проекцию карты.

Возвращаемые значения:

Нет.

Комментарии:

Устанавливает проекцию карты.

---

* **AnsiString distance2str(double)**

Преобразование расстояния в строку.

Параметры:

double - число (например, расстояние) в метрах.

Возвращаемые значения:

Число в отформатированном виде.

Комментарии:

Используется для перевода измеренного расстояния в строку с указанием едениц измерения, в данном случае - метров.

---

* **AnsiString distance2str(double,int)**

Преобразование расстояния в строку.

Параметры:

double - число (например, расстояние) в метрах;
int - идентификатор единиц измерения, в которых будет пересчитано указанное расстояние (0-метры; 1-километры, 2-мили).

Возвращаемые значения:

Строка, содержащая измеренное расстояние в отформатированном виде.

Комментарии:

Используется для пересчета измеренного расстояния из метров в “доступные единицы измерения”. В результате получим текстовую строку соответствующего формата.

Пример:

[MAP_PROJ.DISTANCE2STR(123.5, 1)]
В результате получим: 0.123 км

---

* **AnsiString distance_unit_name()**

Получить установленные единицы измерения.

Параметры:

Не передаются.

Возвращаемые значения:

Строка, содержащая название единиц измерения.

Комментарии:

Используется для получения единиц измерения, установленных в программе в данный момент (метры, километры, мили и т.п.).

---

* **AnsiString distance_unit_name(int)**

Получить название единиц измерения.

Параметры:

int - идентификатор “доступных единиц измерения” (0-метры; 1-километры, 2-мили).

Возвращаемые значения:

Строка, содержащая название единиц измерения.

Комментарии:

Используется для получения названия единиц измерения по их идентификатору.

---

* **double meters2unit(double)**

Пересчет числа из метров в “доступные единицы измерения”.

Параметры:

double - число (расстояние), которое необходимо пересчитать.

Возвращаемые значения:

Число (расстояние) в единицах измерения, установленных в программе в данный момент.

Комментарии:

Используется для пересчета расстояния из метров в единицы измерения, установленные в программе в данный момент.

---

* **double meters2unit(double,int)**

Пересчитывает число в метрах в “необходимые единицы измерения”.

Параметры:

double - число (расстояние), которое необходимо пересчитать;
int - идентификатор необходимых единиц измерения (0-метры; 1-километры, 2-мили).

Возвращаемые значения:

Число (расстояние) в единицах измерения, указанных в параметрах функции.

Комментарии:

Используется для пересчета расстояния из метров в единицы измерения, указанные в параметрах функции (например, километры, мили и т.п.).

---

* **double unit2meters(double)**

Пересчет расстояние из единиц измерения, установленных в программе в данный момент, в метры.

Параметры:

double - число (расстояние), которое необходимо пересчитать.

Возвращаемые значения:

Число (расстояние) в метрах.

Комментарии:

Используется для пересчета расстояния из установленных в программе еденицы измерения в данный момент в метры.

---

* **double unit2meters(double,int)**

Пересчитывает расстояние из каких-либо доступных единиц измерения в метры.

Параметры:

double - число, которое необходимо пересчитать;
int - идентификатор единиц измерения, в которых указано число(0-метры; 1-километры, 2-мили).

Возвращаемые значения:

Число (расстояние) в метрах.

Комментарии:

Используется для пересчета расстояния из указанных в параметрах функции единиц измерения в метры.

---

* **AnsiString degree_str(double)**

Преобразует число в градусах в отформатированную строку.

Параметры:

double - число в градусах.

Возвращаемые значения:

Отформатированная строка.

Комментарии:

Преобразует число в градусах в отформатированную строку стандартного вида.

Пример:

```
MAP_PROJ.INVY(fix_archive.X, fix_archive.Y)
//в результате получим: 50,4300383333229
MAP_PROJ.DEGREE_STR(MAP_PROJ.INVY(fix_archive.X,
fix_archive.Y))
//в результате получим: 50 ' 25 " 45
```

---

* **AnsiString degree_str(double,AnsiString)**

Преобразует число в градусах в отформатированную строку необходимого вида.

Параметры:

double - число в градусах.
AnsiString - маска в виде строки, которая будет использоваться для форматирования. Градусы обозначаются латинской буквой *d,* минуты - *m*, секунды - *s.* Для отображения числа с его дробной частью используется три подряд символа (например, *mmm* - минуты будут отображаться с дробной частью). При использовании двух символов подряд (например, *dd)* - 1 будет отображаться как 01. Разрешается запись, как 'dd*mmss*' , так и 'dd *mm ss*'

Возвращаемые значения:

Отформатированная строка.

Комментарии:

Преобразует число в градусах в отформатированную строку стандартного вида.

Пример:

```
MAP_PROJ.DEGREE_STR(MAP_PROJ.INVY(fix_archive.X,
fix_archive.Y),'dd mm ss')
//в результате получим градусы, минуты, секунды целыми числами:
```

```
50 25 46
MAP_PROJ.DEGREE_STR(MAP_PROJ.INVY(fix_archive.X,
fix_archive.Y),'ddd mmm sss')
//в результате получим градусы , минуты, секунды с дробной частью:
50,4296116666562 25,7766999993694 46,6019999621665
```

Источник данных: obj_inf

Источник информации о мобильных объектах (идентификаторы, названия, описания и т.п.). Предоставляет доступ к списку групп, их параметрам, правам доступа и перечню объектов, входящих в состав групп. Редактируется на сервере мобильных объектов.

Свойства

* **std::vector<int> active_group**
  Возвращает массив идентификаторов групп, доступных пользователю.

* **std::vector<int> active_obj**
  Возвращает массив идентификаторов мобильных объектов, доступных пользователю. Все объекты, входящие в состав “доступных групп” будут перечислены в возвращаемом списке.

Методы

* **TPicture* get_car_picture(int)**

Получить пиктограмму мобильного объекта.

Параметры:

int - идентификатор мобильного объекта.

Возвращаемые значения:

Указатель на пиктограмму мобильного объекта.

Комментарии:

Возвращает пиктограмму, используемую для отображения мобильного объекта на карте (по умолчанию).

---

* **TPicture* get_car_picture(int,double, double)**

Получить пиктограмму мобильного объекта.

Параметры:

int - идентификатор мобильного объекта;
double - смещение “по X” (dX);
double - смещение “по Y” (dY).

Возвращаемые значения:

Указатель на пиктограмму мобильного объекта.

Комментарии:

Используется для получения пиктограммы, используемой для отображения направления движения мобильного объекта (см. Руководство пользователя, раздел Диспетчеризация на стр. 76). Значения dX и dY соответствуют тригонометрическому кругу. Т.е. dX*dX+dY*dY=1. Вектор (0;0)-(dX,dY) определяет направление движения объекта. Свойства dX,dY можно получить у источников данных current_fix_archive, fix_archive, fix_current, fix_last.

Смотри также:

get_car_picture_height()
get_car_picture_width()

---

* **int get_car_picture_height(int)**

Получить высоту пиктограммы мобильного объекта.

Параметры:

int - идентификатор мобильного объекта.

Возвращаемые значения:

Возвращает высоту пиктограммы, используемой для отображения мобильного объекта на карте.

Смотри также:

get_car_picture()
get_car_picture_width()

---

* **int get_car_picture_width(int)**

Получить ширину пиктограммы мобильного объекта.

Параметры:

int - идентификатор мобильного объекта.

Возвращаемые значения:

Возвращает высоту пиктограммы, используемой для отображения мобильного объект на карте.

Смотри также:

get_car_picture()
get_car_picture_height()

---

* **AnsiString group_description(int)**

Получить описание группы мобильных объектов.

Параметры:

int - идентификатор группы мобильных объектов.

Возвращаемые значения:

Возвращает текстовое описание группы мобильных объектов.

Комментарии:

Текстовое описание группы мобильных объектов устанавливается администратором на сервере.

Смотри также:

active_group

---

* **AnsiString group_name(int)**

Получить название группы мобильных объектов.

Параметры:

int - идентификатор группы мобильных объектов.

Возвращаемые значения:

Название группы мобильных объектов.

Комментарии:

Название группы мобильных объектов редактируется администратором на сервере.

Смотри также:

active_group

---

* **int obj_car_type(int)**

Получить идентификатор типа мобильного объекта.

Параметры:

int - идентификатор объекта.

Возвращаемые значения:

Идентификатор типа мобильного объекта.

Комментарии:

Возвращаемый идентификатор соответствует типу, указанному пользователем при настройке клиентской программы (см. Руководство пользователя, раздел Диспетчеризация на стр. 76).

Смотри также:

type_description()
type_name()

---

* **AnsiString obj_description(int)**

Получить описание мобильного объекта.

Параметры:

int - идентификатор объекта.

Возвращаемые значения:

Текстовое описание мобильного объекта.

Комментарии:

Текстовое описание настраивается пользователем в клиентской программе (см. Руководство пользователя, раздел Диспетчеризация на стр. 76).

Смотри также:

active_ob

---

* **int obj_grp_id(int)**

Получить идентификатор группы мобильных объектов.

Параметры:

int - идентификатор мобильного объекта этой группы.

Возвращаемые значения:

Идентификатор группы мобильных объектов.

Смотри также:

group_name()

---

* **AnsiString obj_name(int)**

Получить название мобильного объекта.

Параметры:

int - идентификатор объекта.

Возвращаемые значения:

Название мобильного объекта.

Комментарии:

Возвращаемое значение указывается администратором сервера мобильных объектов при регистрации.

Смотри также:

active_ob

---

* **AnsiString type_description(int)**

Получить описание типа мобильного объекта.

Параметры:

int - идентификатор объекта.

Возвращаемые значения:

Текстовое описание типа мобильного объекта.

Комментарии:

Настраивается пользователем при создании/редактировании типа мобильных объектов в клиентской программе (см. Руководство пользователя, раздел Диспетчеризация на стр. 76).

Смотри также:

obj_car_type()

---

* **AnsiString type_name(int)**

Получить название типа мобильного объекта.

Параметры:

int - идентификатор объекта.

Возвращаемые значения:

Название типа мобильного объекта.

Комментарии:

Настраивается пользователем при создании/редактировании типа мобильных объектов в клиентской программе (см. Руководство пользователя, раздел Диспетчеризация на стр. 76).

Смотри также:

obj_car_type()

Источник данных: poll_error

Позволяет получить информацию об ошибках, связанных с получением или передачей данных и команд управления мобильным объектам. Хранение данных источника организовано в виде таблицы.

Свойства

* **bool close**
  Возвращает True если ошибка завершилась и False в другом случае. Ошибка считается завершенной, если определено время окончания получения сообщения об ошибках данного типа, т.е. получено сообщение, не содержащее ошибки.
* **double code**
  Возвращает числовой код ошибки.
* **double double_time**
  Возвращает дату возникновения ошибки (в формате **double)**.
* **AnsiString info**
  Возвращает дополнительную информацию об ошибке.
* **double interval**
  Возвращает интервал продолжительности ошибки (от начала ошибки до последнего запроса к мобильному объекту, когда ошибка еще была зафиксирована) в формате **double**.
* **int obj_id**
  Возвращает идентификатор мобильного объекта, для которого произошла ошибка.
* **AnsiString obj_name**
  Возвращает название мобильного объекта.
* **unsigned int position()**
  Возвращает номер текущей записи в выборке.
* **unsigned int record_count**
  Возвращает общее количество записей в выборке.
* **std::vector<int> request_objects**
  Возвращает массив идентификаторов мобильных объектов, указанных в запросе (объекты, по которым формировалась выборка из архива).
* **TDateTime request_time_from**
  Возвращает время начала просмотра архива, указанное в запросе.
* **TDateTime request_time_to**
  Возвращает время окончания просмотра архива, указанное в запросе.

* **AnsiString str_code**
  Возвращает строку ошибки.
* **TDateTime time**
  Возвращает дату возникновения ошибки в формате **TDateTime**.
* **TDateTime time_interval**
  Возвращает интервал от начала ошибки до последнего запроса к мобильному объекту, когда ошибка еще была зафиксирована (в формате **TDateTime**).

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет, достигнуто ли окончание потока данных и, в зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

---

* **void close()**

Закрыть соединения с сервером.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Прекратить получение данных по данному запросу от сервера. При этом в источнике данных будут доступны только те записи, которые были получены до вызова close(). Их количество можно получить функцией loaded(). Проверить получена ли текущая запись можно функцией is_data_present(). Нет необходимости вызывать close() после получения всех данных, т.к. в этом случае поток данных закрывается самостоятельно.

Смотри также:

is_data_present()
loaded()

---

* **void default_param(TDateTime &,TDateTime &,std::vector<int> &)**

Получение параметров источника данных.

Параметры:

TDateTime - время “с” (начало периода просмотра). “0” означает, что начало не ограничено;
TDateTime - время “до” (конец периода просмотра). “0” означает, что окончание не ограничено;
std::vector<int> - массив идентификаторов мобильных объектов. Полученный результат будет сохранен в этих переменных.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет получить параметры, установленные по умолчанию для экземпляра источника данных. Используется для передачи этих параметров при инициализации экземпляра источника данных из программного кода функцией init().

Смотри также:

init()

---

* **void default_param(idataset_ctrl*,Variant &)**

Получить параметры источника данных.

Параметры:

*idataset_ctrl** - 0 (указатель на "интерфейс управления источником данных". Зарезервировано. Должен быть 0.);
Variant - Variant& - ссылка на переменную, в которую будет записаны параметры по умолчанию источника данных. Структура возвращаемого значения имеет вид массива: 0-ой элемент – время "с", 1-ый - время "до", 2-ой - массив идентификаторов мобильных объектов..

Возвращаемые значения:

Нет.

Комментарии:

Позволяет получить параметры, установленные по умолчанию для экземпляра источника данных. Используется для передачи параметров при инициализации экземпляра источника данных из программного кода функцией init().

Смотри также:

init()

---

* **void first_record()**

Переместиться на первую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на первую запись таблицы источника данных.

Смотри также:

next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.

Смотри также:

first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.

Параметры:

unsigned int - число, указывающее номер записи, на которую необходимо переместиться.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на необходимую запись в таблице источника данных.

Смотри также:

first_record()
next_record()

---

* **const AnsiString & get_error()**

Возвращает ошибку в виде строки.

Параметры:

Не передаются.

Возвращаемые значения:

Произошедшая ошибка в виде строки.

Комментарии:

Если в процессе выполнения отчета возникла ошибка – функция вернет эту ошибку в виде строки.

---

* **AnsiString get_object_name(int)**

Получить название мобильного объекта.

Параметры:

int - идентификатор мобильного объекта.

Возвращаемые значения:

Название мобильного объекта.

Комментарии:

Возвращаемое значение указывается администратором сервера мобильных объектов при его регистрации.

Смотри также:

obj_id

---

* **void init(const TDateTime &,const TDateTime &,const std::vector<int> &)**

Инициализация экземпляра источника данных.

Параметры:

TDateTime - время “с” (начало периода просмотра архива событий);
TDateTime - время “до” (конец просмотра архива событий);
std::vector<int> - массив идентификаторов мобильных объектов.

Возвращаемые значения:

Нет.

Комментарии:

Если при проектировании отчета некоторые источники данных не были инициализированы - это можно выполнить из программного кода, используя даный метод. Типовые значения параметров можно получить, вызвав метод default_param(). Для указания параметров пользователем используйте show_dialog().

Смотри также:

show_dialog()
default_param()

---

* **bool is_active()**

Проверка доступности источника данных.

Параметры:

Не передаются.

Возвращаемые значения:

True - если источник данных доступен (даже если он не инициализирован);
False - если источник данных не доступен.

Комментарии:

Проверяет, доступен ли источник данных в данном отчете.

Смотри также:

init()

---

* **bool is_data_present()**

Проверка получения записи.

Параметры:

Не передаются.

Возвращаемые значения:

True - если необходимая запись получена;
False - если необходимая запись не получена.

Комментарии:

Процесс получения записей от сервера занимает какое-то время.
Если экземпляр источника данных инициализируется до выполнения отчёта (если он отмечен в списке источников, которые подлежат инициализации), то к моменту выполнения отчёта все записи будут получены.
Если инициализировать источник данных самостоятельно —необходимо следить за наличием записи с помощью функции is_data_present().Чтобы дождаться получения всех записей используйте функцию synchronize().

Смотри также:

loaded()
synchronize()

---

* **unsigned int loaded()**

Возвращает количество загруженных записей.

Параметры:

Не передаются.

Возвращаемые значения:

Число загруженных записей.

Комментарии:

Возвращает количество полученных записей с сервера. Записи загружаются последовательно.

Смотри также:

is_data_present()
synchronize()

---

* **unsigned int position()**

Возвращает номер текущей записи в выборке.

Параметры:

Не передаются.

Возвращаемые значения:

Номер текущей записи.

Комментарии:

Возвращает номер текущей записи, на которой сейчас находится “курсор”.

Смотри также:

set_record()

---

* **unsigned int record_count()**

Возвращает общее количество записей.

Параметры:

Не передаются.

Возвращаемые значения:

Количество всех записей.

Комментарии:

Возвращает общее количество записей в таблице источника данных.

Смотри также:

loaded()

---

* **bool show_dialog(TDateTime &,TDateTime &,std::vector<int> &)**

Вызывает диалог инициализации источника данных.

Параметры:

TDateTime - время “с” (начало периода просмотра архива событий);
TDateTime - время “до” (конец просмотра архива событий);
std::vector<int> - массив идентификаторов мобильных объектов.
В указанных параметрах передаются значения, отображаемые при создании диалогового окна. Полученный результат будет сохранен в этих переменных.

Возвращаемые значения:

True - если диалоговое окно создано;
False - если возникала ошибка.

Комментарии:

Этот метод используется для передачи значений, выбранных пользователем, в отчет. Как правило, используется для источников, не инициализированных при создании отчета(см. раздел Создание и удаление отчета на стр. 7.). Полученные параметры передаются в метод init().

Смотри также:

init()
default_param()

---

* **void synchronize(idataset_ctrl*)**

Синхронизация.

Параметры:

*idataset_ctrl** - 0.

Возвращаемые значения:

Нет.

Комментарии:

Используется для получения всех необходимых данных после инициализации источника данных из программного кода.

Смотри также:

loaded()

Источник данных: img_process

Источник данных, использующийся доя обработки изображений (изменить размер, повернуть и т.п.), а также для получения информации о рисунках.

Свойства

* **int height**
  Возвращает высоту рисунка.
* **bool transparent**
  Возвращает True если рисунок прозрачен или False в другом случае.
* **TPicture* val**
  Возвращает указатель на рисунок, присвоенный данному экземпляру источника данных функцией **set_val()**. Если ранее этому источнику данных ни один рисунок не был присвоен — возвращает рисунок с нулевой высотой, нулевой шириной.
* **int width**
  Возвращает ширину рисунка.

Методы

* **void crop(int,int)**

Обрезка рисунка.

Параметры:

int - ширина;
int - высота.

Возвращаемые значения:

Нет.

Комментарии:

Выполняет обрезку изображения. За точку отсчета принимается центральная точка изображения. От этой точки откладывается необходимые ширина и высота. Все остальное “обрезается”.

Смотри также:

width
height

---

* **void crop(int,int,int,int)**

Обрезка рисунка.

Параметры:

int - x-координата точки;
int - y-координата точки;
int - ширина;
int - высота.

Возвращаемые значения:

Нет.

Комментарии:

Эта функция выполняет обрезку изображения. За точку отсчета принимается точка, относительно которой будет “обрезан” рисунок. От этой точки в равных частях откладывается необходимые ширина и высота.

Смотри также:

width
height

---

* **void draw(int,int,TPicture*)**

Отобразить рисунок.

Параметры:

int - x-координата верхнего левого угла рисунка;
int - y-координата верхнего левого угла рисунка";
*TPicture** - указатель на рисунок.

Возвращаемые значения:

Нет.

Смотри также:

crop()
val

---

* **void rotate90()**

Поворот изображения.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Поворачивает изображение на 90 градусов против часовой стрелки.

---

* **void rotate_90()**

Поворот изображения.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Поворачивает изображение на 90 градусов по часовой стрелке.

---

* **void set_transparent(bool)**

Установить режим прозрачности изображения.

Параметры:

True - установить режим прозрачности;
False - отменить режим прозрачности.

Возвращаемые значения:

Нет.

---

* **void set_val(TPicture*)**

Загрузить рисунок в источник данных.

Параметры:

*TPicture** - указатель на рисунок.

Возвращаемые значения:

Нет.

Комментарии:

Используется для загрузки рисунка в класс источника данных. При этом создается копия рисунка для последующей работы — изменения размера, поворота и т.п.

Пример:

```
begin
  img_process.set_val(map_window.picture);
  img_process.crop(200,200);
  pic.Picture:=img_process.val;
  pic.Picture:= IMG_PROCESS.SET_TRANSPARENT(True);
end
```

Источник данных: map_window

Источник данных, позволяющий получить доступ к карте, открытой в данный момент в программе. Доступен только для активного окна открытой карты. Используется для получения характеристик карты, ее свойств, изображения видимого участка и т.п.

Свойства

* **int http_request_delay**
  Возвращает время задержки перед загрузкой карты в миллисекундах. Если значение 0 - задержки нет.
* **AnsiString map_name**
  Возвращает название карты (например, Киевская область).
* **double map_scale**
  Возвращает масштаб карты в котором она изготовлена (одна из характеристик карты).
* **AnsiString map_type**
  Возвращает тип карты (например: город, область, страна).
* **TPicture* picture**
  Возвращает рисунок видимой части карты, открытой в окне программы.
* **int picture_height**
  Возвращает высоту рисунка.
* **int picture_widt**
  Возвращает ширину рисунка.
* **MapProj* pj**
  Возвращает указатель на проекцию открытой карты.
* **double relative_scale**
  Возвращает относительный масштаб карты от 0 до 1 (Значение “0” соответствует наиболее детальному масштабу, а значение “1” — наименее детальному.)
* **double scale**
  Возвращает масштаб карты, в котором она отображена в момент формирования отчета.
* **AnsiString title**
  Возвращает заголовок карты (например: Карта - Киев - Просмотр журнала с 24.05.2003 12:41:28).

Методы

* **bool is_active()**

Проверка доступности источника данных.

Параметры:

Не передаются.

Возвращаемые значения:

True - если источник данных доступен (даже если он не инициализирован);
False - если источник данных не доступен.

Комментарии:

Проверяет доступен ли источник данных в данном отчете.

---

* **void get_extent(double &,double &,double &)**

Получить координаты участка карты, отображенного в окне программы.

Параметры:

double - x-координата центральной точки изображения;
double - y-координата центральной точки изображения;
double - масштаб карты.
Полученный результат будет сохранен в этих переменных.

Возвращаемые значения:

Нет.

Комментарии:

Используется для получения координат и масштаба участка карты, отображенного в видимой области открытого окна. Масштаб передается одним числом (например: 5000 =1:5000).

---

* **void set_extent(double ,double ,double)**

Установить координаты участка карты, отображенного в окне программы.

Параметры:

double - x-координата центральной точки изображения;
double - y-координата центральной точки изображения;
double - масштаб карты.

Возвращаемые значения:

Нет.

Комментарии:

Используется для установки координат и масштаба участка карты, отображенной в видимой области открытого окна программы. Масштаб передается одним числом (например, 5000 =1:5000).

---

* **void get_extent(double &,double &,double &,double &)**

Получить координаты участка карты, отображенного в открытом окне программы.

Параметры:

double - x-координата нижнего левого угла;
double - y-координата нижнего левого угла;
double - x-координата верхнего правого угла;
double - y-координата верхнего правого угла.
Полученный результат будет сохранен в этих переменных.

Возвращаемые значения:

Нет.

Комментарии:

Используется для получения координат участка карты, отображенного в открытом окне программы на видимой области в открытом окне программы.

---

* **void get_map_extent(double &,double &,double &,double &)**

Получить координаты карты.

Параметры:

double - x-координата нижнего левого угла;
double - y-координата нижнего левого угла;
double - x-координата верхнего правого угла;
double - y-координата верхнего правого угла.
Полученный результат будет сохранен в этих переменных.

Возвращаемые значения:

Нет.

Комментарии:

Используется для получения координат всей карты, открытой в данный момент в окне программы.

---

* **void set_extent_test_bound(double,double,double)**

Установить координаты участка карты, отображенного в окне программы.

Параметры:

double - x-координата центральной точки изображения;
double - y-координата центральной точки изображения;
double - масштаб карты.

Возвращаемые значения:

Нет.

Комментарии:

Используется для установки координат и масштаба участка карты, отображенной в видимой области открытого окна программы. Масштаб передается одним числом (например, 5000 =1:5000). Перед установкой координат вначале будет проверено, попадает ли указанная центральная точка в область карты. Если точка попадает – координаты участка карты будут установлены. Если точка не попадает – ничего не произойдет.

---

* **void pix2world(double &,double &)**

Переводит координаты точки в пикселах в мировые координаты.

Параметры:

double - x-координата точки в пикселах (относительно открытого окна, содержащего изображение карты);
double - y- координата точки в пикселах (относительно открытого окна, содержащего изображение карты).
Полученный результат будет сохранен в этих переменных (в мировых координатах).

Возвращаемые значения:

Нет.

Комментарии:

Используется для пересчета координат точки в мировые координаты.

---

* **double pix2world(double)**

Пересчитать длину отрезка (пикселы в мировые координаты).

Параметры:

double - длинна отрезка в пикселах.

Возвращаемые значения:

Длинна отрезка в мировых координатах.

Комментарии:

Используется для пересчета длины отрезка, рассчитанной с использованием пиксельных координат крайних точек отрезка, в длину, соответствующую мировым координатам указанных точек.

---

* **void set_http_request_delay(int)**

Время задержки перед загрузкой карты.

Параметры:

int - время задержки в миллисекундах.

Возвращаемые значения:

Нет.

Комментарии:

Метод устанавливает задержку перед загрузкой фрагментов карты. Задержка перед загрузкой изображения позволяет избежать создания лишних запросов к серверу. При быстром перемещении карты, некоторые участки карты не будут полностью загружены, но запрос поступит на сервер и начнется его загрузка. Если в функцию передать 0 - задержки не будет, что ускорит загрузку изображений.

Смотри также:
http_request_delay

---

* **void set_relative_scale(double)**

Устанавливает относительный масштаб карты.

Параметры:
double - значение от 0 до 1.

Возвращаемые значения:
Нет.

Комментарии:
Функция устанавливает относительный масштаб карты. Значеие “0” соответствует наиболее детальному масштабу, а значение “1” — наименее детальному.

Смотри также:
set_scale()

---

* **void set_scale(double)**

Устанавливает масштаб карты.

Параметры:
double - число, определяющее необходимый масштаб.

Возвращаемые значения:
Нет.

Комментарии:
Функция устанавливает необходимый масштаб для карты. Например, число 5000 соответствует масштабу 1:5000

Смотри также:
Приложение А на стр. 169

---

* **void set_size(int,int)**

Устанавливает размер рисунка.

Параметры:
int - необходимая ширина рисунка;
int - необходимая длина рисунка

Возвращаемые значения:
Нет.

Комментарии:
Функция устанавливает необходимые размеры изображения. Длинна и ширина в равных частях откладываются от центра изображения. К примеру, если для изображения 30*40 пикселей установить размер 40*50, то: существующее изображение будет расположено по центру, а по краям появятся полосы шириной в 10 пикселей.

---

* **bool wait_map_load()**

Ожидать получения всех данных.

Параметры:
Не передаются.

Возвращаемые значения:
True - если все необходимые данные получены;
False - если не все данные получены.

Комментарии:
При использовании этой команды программа будет ждать получения всех необходимых данных (полной загрузки рисунка). Если загрузка каких-либо участков не удалась – вернет *False.*

---

* **bool wait_map_load(unsigned int)**

Ожидать получения данных в течении указанного промежутка времени.

Параметры:

unsigned int - значение промежутка времени в миллисекундах.

Возвращаемые значения:

True - если все необходимые данные получены;
False - если не все данные получены.

Комментарии:

При использовании этой команды программа будет ждать получения всех необходимых данных (полной загрузки рисунка) в течении промежутка времени, указанного в параметрах функции. Если за указанное время удалось получить все необходимые данные – функция вернет *True.* Если получение данных не удалось – вернет *False*.

---

* **void world2pix(double &,double &)**

Пересчет координаты точки из мировых координат в пикселы.

Параметры:

double - x-координата точки в мировых координатах;
double - y- координата точки в мировых координатах.
Полученный результат будет сохранен в этих переменных (в пикселах).

Возвращаемые значения:

Нет.

Смотри также:

Приложение А на стр. 169

---

* **double world2pix(double)**

Пересчет длины отрезка из мировых координат в пикселы.

Параметры:

double - длина отрезка в мировых координатах.

Возвращаемые значения:

Длина отрезка в пикселах.

Комментарии:

Используется для пересчета длины отрезка, рассчитанной с использованием мировых координат крайних точек отрезка, в длину, соответствующую пиксельным координатам указанных точек.

---

* **void view_region(double,double,double,double)**

Отобразить указанный участок карты в открытом окне программы.

Параметры:

double - x-координата нижнего левого угла;
double - y-координата нижнего левого угла;
double - x-координата верхнего правого угла;
double - y-координата верхнего правого угла.

Возвращаемые значения:

Нет.

Комментарии:

Используется для отображения в открытом окне программы необходимого участка карты.

---

* **void view_pixel_region(double,double,double,double)**

Отобразить указанный участок карты в открытом окне программы (координаты передаются в пикселях).

Параметры:

double - x-координата нижнего левого угла в пикселях;
double - y-координата нижнего левого угла в пикселях;

double - x-координата верхнего правого угла в пикселях;
double - y-координата верхнего правого угла в пикселях.

Возвращаемые значения:

Нет.

Комментарии:

Используется для отображения в открытом окне программы необходимого участка карты. В качестве параметров передаются пиксельные координаты (точка с координатами (0, 0) соответствует левому верхнему углу окна). Все координаты передаются относительно текущего рисунка карты.

Источник данных: dataset_functions

Источник данных, применяемый для получения информации о классах, свойствах и их методах, использующихся в программе для формирования отчета. Хранение данных источника организовано в виде таблицы.

Свойства

* **unsigned int argument_count**
  Возвращает количество аргументов метода.
* **AnsiString full_name**
  Возвращает полное имя метода (например, *void world2pix(double &,double &)*)
* **AnsiString result_type**
  Возвращает тип результатов метода.
* **AnsiString short_name**
  Возвращает короткое имя метода (например, *world2pix*).
* **AnsiString val**
  Возвращает имя источника данных функции, установленного методом **set_val**.
* **unsigned int position()**
  Возвращает номер текущей записи в выборке.
* **unsigned int record_count**
  Возвращает общее количество записей в выборке.

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет, достигнуто ли окончание потока данных и, в зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

---

* **void first_record()**

Переместиться на первую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на первую запись таблицы источника данных.

Смотри также:

next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.

Смотри также:

first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.

Параметры:

unsigned int - число, указывающее номер записи, на которую необходимо переместиться.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор”на нужную запись в таблице источника данных. Эта функция доступна при прямом доступе к записям.

Смотри также:

next_record()
first_record()

---

* **AnsiString get_argument_type(unsigned int)**

Получить тип аргументов метода.

Параметры:

unsigned int - индекс аргумента метода, тип которого необходимо получить.

Возвращаемые значения:

Возвращает тип аргумента, используемого в методе. Индексация аргументов метода начинается с 0.

---

* **bool is_out_argument(unsigned int)**

Получить сведения о том, можно ли в аргументе вернуть какое-либо значение.

Параметры:

unsigned int - индекс аргумента метода.

Возвращаемые значения:

True - если в аргументе можно вернуть какое-либо значение;
False - если в аргументе нельзя вернуть никакого значения.

Комментарии:

На то, что в аргументе можно вернуть какое-либо значение, указывает знак “&”, расположенный в списке параметров метода после конкретного аргумента. Параметры метода, содержащие такие аргументы, можно передавать только используя функции ***ARGS***.

---

* **void set_val(AnsiString)**

Установить название источника данных.

Параметры:

AnsiString - название источника данных.

Возвращаемые значения:

Нет.

Комментарии:

Для получения информации о методах конкретного источника данных (используя dataset_function) необходимо указать его название. Для этого и используется данный метод.

Смотри также:

val

Источник данных: dataset_props

Источник данных о свойствах классов, использующихся в программе для получения информации из источников данных. Хранение данных источника организовано в виде таблицы.

Свойства

* **AnsiString prop_name**
  Возвращает имя свойства (например: *is_trace_mode*).
* **AnsiString prop_type**
  Возвращает тип свойства (например: для переменной *is_trace_mode* — тип bool).
* **AnsiString val**
  Возвращает название источника данных, установленного методом set_val().
* **unsigned int position()**
  Возвращает номер текущей записи в выборке.
* **unsigned int record_count**
  Возвращает общее количество записей в выборке.

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет, достигнуто ли окончание потока данных и, в зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

---

* **void first_record()**

Переместиться на первую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на первую запись таблицы источника данных.

Смотри также:

next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.

Смотри также:

first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.

Параметры:

unsigned int - число, указывающее номер записи, на которую необходимо переместиться.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на необходимую запись в таблице источника данных.

Смотри также:

first_record()
next_record()

---

* **void set_val(AnsiString)**

Установить название источника данных.

Параметры:

AnsiString - название источника данных.

Возвращаемые значения:

Нет.

Комментарии:

Для получения информации о методах конкретного источника данных — необходимо указать его название. Для этого и используется данный метод.

Смотри также:

val

Источник данных: virtual_dataset

Предоставляет доступ к вертуальной таблице данных о всеми неободимыми методами и свойствами. Используется для управления формированием отчета. Как правило, в качестве источника для “бэндов” (см. Объект “бэнд” на стр. 18). Хранение данных источника организовано в виде таблицы.

Свойства

* **bool eof**
  Возвращает True если достигнут конец таблицы или False в другом случае.
* **unsigned int position**
  Возвращает текущую позицию в таблице.
* **unsigned int record_count**
  Возвращает общее количество записей в выборке.

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет достигнуто ли окончание потока. В зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

---

* **void first_record()**

Переместиться на первую запись в таблице.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на первую запись таблицы источника данных.

Смотри также:

next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись в таблице.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись таблицы источника данных.

Смотри также:

first_record()
set_record()

* **void set_record(unsigned int)**

Переместиться на необходимую запись в таблице.

Параметры:

unsigned int - номер нужной записи.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на необходимую запись в источнике данных.

Смотри также:

next_record()
first_record()

---

* **void set_record_count(unsigned int)**

Указать количество записей.

Параметры:

unsigned int - необходимое количество записей.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить количество записей, “содержащихся” в виртуальном источнике данных.

Источник данных: datasets_list

Используется для получения общего списка источников данных, существующих в программе. Хранение данных источника организовано в виде таблицы.

Свойства

* **AnsiString class_name**
  Возвращает название источника данных.
* **int function_count**
  Возвращает количество методов источника данных.
* **int property_count**
  Возвращает количество свойств источника данных.
* **unsigned int position**
  Возвращает текущую позицию в таблице.
* **unsigned int record_count**
  Возвращает общее количество записей в выборке.

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет достигнуто ли окончание потока данных или нет и, в зависимости от результата, возвращает соответствующее значение.

Смотри также:
next_record()

---

* **void first_record()**

Переместиться на первую запись в таблице.

Параметры:
Не передаются.

Возвращаемые значения:
Нет.

Комментарии:
Позволяет переместить “курсор” на первую запись таблицы источника данных.

Смотри также:
next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись в таблице.

Параметры:
Не передаются.

Возвращаемые значения:
Нет.

Комментарии:
Позволяет переместить “курсор” на следующую за текущей запись таблицы источника данных.

Смотри также:
first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись в таблице.

Параметры:
unsigned int - необходимое количество записей.

Возвращаемые значения:
Нет.

Комментарии:
Позволяет переместить “курсор” на необходимую запись в таблице источника данных.

Смотри также:
next_record()
first_record()

Источник данных: dbb_storadge

Источник данных представляет собой таблицу, содержащую запись о каждой таблице, находящейся в “базе данных”. Он позволяет получить информацию о таблицах, содержащих данные пользователя (см. Руководсто пользователя, раздел Работа с таблицами на стр. 87). Используется для доступа к произвольным таблицам без подключения их в качестве “самостоятельных” источников данных (см. Интерфейсы источников данных на стр. 72). Если необходимо использовать при выборке данных фильтры, поиск и сортировку — подключайте таблицу, как источник данных.

Свойства

* **std::vector<AnsiString> columns_list**
  Возвращает перечень полей таблицы, запись о которой является активной в данный момент (т.е. на этой записи установлен “курсор”).
* **unsigned int position**
  Возвращает номер текущей позиции “курсора”.
* **unsigned int record_count**
  Возвращает общее количество записей в выборке (источнике данных).
* **AnsiString table_comment**
  Возвращает комментарии, указанные в редакторе таблиц для таблицы, запись которой является текущей в данный момент времени (т.е. на этой записи установлен “курсор”).
* **AnsiString table_name**
  Возвращет название таблицы, запись о которой является текущей.
* **unsigned int table_record_count**
  Возвращает общее количество записей таблицы, запись о которой является текущей.
* **std::vector<AnsiString> tables_list**
  Возвращает список всех таблиц, которые находяться в базе данных.
* **std::vector<AnsiString> types_list**
  Возвращает перечень типов, которые используются при создании таблиц.

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет достигнуто ли окончание потока данных или нет и, в зависимости от результата, возвращает соответствующее значение.

---

* **void create_table(const AnsiString &)**

Добавление новой таблицы в базу данных.

Параметры:

const AnsiString & - название создаваемой таблицы

Возвращаемые значения:

Нет.

Комментарии:

Создание новой таблицы с указанным именем.

---

* **void drop_table(const AnsiString &)**

Удаление таблицы.

Параметры:

const AnsiString & - название удаляемой таблицы

Возвращаемые значения:

Нет.

Комментарии:

Позволяет удалить необходимую таблицу из базы данных.

---

* **void first_record()**

Установить “курсор” в начало таблицы записей.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Установить “курсор” на первую запись в таблице.

---

* **AnsiString get_column_type(const AnsiString &)**

Получить тип колонки таблицы.

Параметры:

const AnsiString & - название колонки

Возвращаемые значения:

AnsiString - тип указанной колонки

Комментарии:

Позволяет получить тип поля таблицы, запись о которой является текущей (т.е. на этой записи установлен “курсор”).

---

* **void goto_table(const AnsiString &)**

Сделать текущей запись о необходимой таблице.

Параметры:

const AnsiString & - название таблицы

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на запись о таблице, название которой передается в параметрах метода.

---

* **bool is_column_autoinc(const AnsiString &)**

Позволяет узнать, является ли указанное поле таблицы автоинкрементным.

Параметры:

const AnsiString & - название колонки таблицы

Возвращаемые значения:

True - поле автоинкрементно;

False - поле не автоинкрементно.

Комментарии:

Позволяет узнать, установлено ли свойство автоинкрементности для необходимой колонки таблицы, запись о которой является текущей в данный момент.

---

* **bool is_column_indexed(const AnsiString &)**

Позволяет узнать, является ли указанное поле таблицы индексированным.

Параметры:

const AnsiString & - название колонки таблицы

Возвращаемые значения:

True - свойство индексации установлено;

False - свойство индексации не установлено.

Комментарии:

Позволяет узнать, установлено ли свойство индексации для необходимой колонки таблицы, запись о которой является текущей в данный момент.

---

* **bool is_column_linked(const AnsiString &)**

Позволяет узнать, содержит ли данная колонка ссылку.

Параметры:

const AnsiString & - название колонки таблицы

Возвращаемые значения:

True - поле содержит ссылку;
False - поле не содержит ссылку.

Комментарии:

Позволяет узнать, содержит ли поле ссылку, т.е. ограничиваются ли возможные значения данной колонки определенным диапазоном значений, представленых в другой таблице. Рассматривается поле таблицы, запись о которой является текущей на данный момент.

---

* **bool is_column_unique(const AnsiString &)**

Позволяет узнать, установлено ли свойство уникальности для указанного поля.

Параметры:

const AnsiString & - название колонки таблицы

Возвращаемые значения:

True - свойство уникальности установлено;
False - свойство уникальности не установлено.

Комментарии:

Позволяет узнать, установлено ли свойство уникальности для указанной колонки. Установка этого свойства гарантирует проверку значений поля таблиы на уникальность. Рассматривается поле таблицы, запись о которой является текущей в данный момент.

---

* **void next_record()**

Перейти на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Переводит “курсор” на следующую запись в таблице (после текущей).

---

* **void rename_table(const AnsiString &,const AnsiString &)**

Переименовать таблицу.

Параметры:

const AnsiString & - текущее название таблицы;
const AnsiString & - новок название таблицы.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет изменить название необходимой таблицы в базе данных.

---

* **void set_record(unsigned int)**

Установить позицию “курсора”.

Параметры:

unsigned int - локальный индекс записи в источнике данных.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить “курсор” на указанную запись в таблице, т.е. изменить текущую позицию “курсора”

---

* **bool type_can_autoinc(const AnsiString &)**

Позволяет узнать, может ли указанный тип быть автоинкрементным.

Параметры:

const AnsiString & - название типа данных.

Возвращаемые значения:

True - если для поля такого типа можно установить свойство автоинкрементности;
False - для поля указанного типа невозможно установить свойство автоинкрементности.

Комментарии:

При передаче параметров, тип следует указывать в формате, каким мы получаем его при использовании свойства types_list (т.е. *int*, а не *целый*; *TPicture **, а не *графика* и т.п.). Все доступные для использования типы можно получить, воспользовавшись свойством tipes_list.

Смотри также:

types_list

---

* **bool type_can_indexed(const AnsiString &)**

Позволяет узнать, может ли указанный тип быть индексированным.

Параметры:

const AnsiString & - название типа данных.

Возвращаемые значения:

True - если для поля такого типа можно установить свойство индексированности;
False - для поля указанного типа невозможно установить свойство индексированности.

Комментарии:

При передаче параметров, тип следует указывать в формате, каким мы получаем его при использовании свойства types_list (т.е. *int*, а не *целый*; *TPicture **, а не *графика* и т.п.). Все доступные для использования типы можно получить, воспользовавшись свойством tipes_list.

Смотри также:

types_list

---

* **bool type_can_link(const AnsiString &)**

Позволяет узнать, может ли поле указанного типа содержать ссылку.

Параметры:

const AnsiString & - название типа данных.

Возвращаемые значения:

True - если поле указанного тип может содержать ссылку;
False - поле указанного типа не может содержать ссылку.

Комментарии:

При передаче параметров, тип следует указывать в формате, каким мы получаем его при использовании свойства types_list (т.е. *int*, а не *целый*; *TPicture **, а не *графика* и т.п.). Все доступные для использования типы можно получить, воспользовавшись свойством tipes_list.

Смотри также:

types_list

---

* **bool type_can_unique(const AnsiString &)**

Позволят узнать, можна ли для поля указанного типа установить свойство уникальности.

Параметры:

const AnsiString & - название типа данных.

Возвращаемые значения:

True - если для поля такого типа можно установить свойство уникальности;
False - для поля указанного типа невозможно установить свойство уникальности.

Комментарии:

При передаче параметров, тип следует указывать в формате, каким мы получаем его при использовании свойства types_list (т.е. *int*, а не *целый*; *TPicture **, а не *графика* и т.п.). Все доступные для использования типы можно получить, воспользовавшись свойством tipes_list.

Смотри также:

types_list

---

* **AnsiString type_view_name(const AnsiString &)**

Получить условное название типа данных, используемое в редакторе таблиц .

Параметры:

const AnsiString & - название типа данных.

Возвращаемые значения:

AnsiString - условное название указанного типа данных.

Комментарии:

При передаче параметров, тип следует указывать в формате, каким мы получаем его при использовании свойства types_list (т.е. *int*, а не *целый*; *TPicture **, а не *графика* и т.п.). Все доступные для использования типы можно получить, воспользовавшись свойством tipes_list.

Смотри также:

types_list

Источник данных: table_<имя таблицы>

Каждая таблица, созданная пользователем (см. Руководсто пользователя, раздел Работа с таблицами на стр. 87), может служить в качестве источника данных при формировании отчетов (таблица инициализирована).

Свойства

* **unsigned int position**
  Возвращает номер текущей позиции “курсора”.
* **unsigned int record_count**
  Возвращает общее количество записей в выборке (источнике данных).
* **unsigned int oid**
  Возвращает номер записи источника данных, под которым она храниться в таблице (сквозная глобальная нумерация).
* **unsigned int global_record_count**
  Возвращает количество записей в таблице пользователя, не зависимо от выборки источника данных.
* **bool autosave**
  Возвращает *True* если автосохранение включено, и *False* - в другом случае.

У всех источников данных типа table_<имя таблицы> есть общие (описаны выше) и собственные свойства. Тип и количество таких свойств зависит от количества и типа полей в таблице.

Методы

* **void add_filter_set(const AnsiString &,const std::vector<Variant> &)**

Установить фильтр данных.

Параметры:

const AnsiString & - название колонки, данные которой необходимо отфильтровать;
const std::vector<Variant> & - массив значений, которые будут отображаться в источнике данных в результате фильтрации.

Возвращаемые значения:

Нет.

Комментарии:

Отфильтровать строки таблицы, по указанному полю.Строка будет присутствовать в выборке, если значение (в указанной колонке) присутствует в “фильтре”.

---

* **void add_filter_range(const AnsiString &,const Variant &,const Variant &)**

Установить фильтр данных.

Параметры:

const AnsiString & - название колонки, данные которой необходимо отфильтровать;
const Variant & - начальное значение диапазона значений для фильтрации;
const Variant & - конечное значение диапазона значений.

Возвращаемые значения:

Нет.

Комментарии:

В выборке присутствуют только строки , у которых значение (в соответствующей колонке) попадает в указанный диапазон (“с” - “по”).

---

* **void add_filter_regexp(const AnsiString &,const AnsiString &)**

Установить фильтр данных.

Параметры:

const AnsiString & - название колонки, данные которой необходимо отфильтровать;
const AnsiString & - правило фильтрации, согласно которому будут отфильтровываться данные указанного поля.

Возвращаемые значения:

Нет.

Комментарии:

В выборке присутствуют только строки , у которых значение (в соответствующей колонке) соответствует указанному правилу. Этот метод может применяться только к полям строкового типа.

Смотри также:

field_as_string()
add_filter_regexp()

---

* **void add_filter_regexp(const AnsiString &,const AnsiString &,int)**

Установить фильтр данных с указанным параметром.

Параметры:

const AnsiString & - название колонки, данные которой необходимо отфильтровать;
const AnsiString & - правило фильтрации, согласно которому будут отфильтровываться данные указанного поля;
int - параметр, учивающийся при фильтрации.

Возвращаемые значения:

Нет.

Комментарии:

В выборке присутствуют только строки , у которых значение (в соответствующей колонке) соответствует указанному правилу. Этот метод может применяться только к полям строкового типа. Перечень доступных параметров, с учетом которых будет производиться фильтрация: “1” - без учета регистра символов;

Смотри также:

field_as_string()
add_filter_regexp()

---

* **void add_filter_ethalon(const AnsiString &,const Variant &)**

Установить фильтр данных

Параметры:

const AnsiString & - название колонки, данные которой необходимо отфильтровать;
const Variant & - значение;

Возвращаемые значения:

Нет.

Комментарии:

Фильтрация данных таблицы по одному значнию, из указанной колонки.

Смотри также:

add_filter_set()

---

* **void add_sorter(const AnsiString &)**

Сортировка колонки таблицы по возрастанию.

Параметры:

const AnsiString & - название колонки, данные которой необходимо отсортировать;

Возвращаемые значения:

Нет.

Комментарии:

Значения указанной колонки таблицы будут отсортированы по возрастанию.

---

* **void add_sorter(const AnsiString &,bool)**

Сортировка по возрастанию или убыванию.

Параметры:

const AnsiString & - название колонки, данные которой необходимо отсортировать;
bool:
True - сортировка по возрастанию;
False - сортировка по убыванию.

Возвращаемые значения:

Нет.

Комментарии:

Значения указанной колонки таблицы будут отсортированы по возрастанию или убыванию, в зависимости от параметра *bool*.

---

* **bool begin_change_session()**

Монопольное использование таблицы.

Параметры:

Не передаются.

Возвращаемые значения:

True - если сессия была успешно открыта, т.е. установлен режим монопольного использования таблицы.
False - если сессию открыть не удалось.

Комментарии:

Установить режим монопольного использования таблицы (открытие сессии редактирования таблицы). При этом доступ к таблице другим пользователям будет запрещен. Только в монопольном режиме возможно редактирование данных таблицы.

Смотри также:

change_field()
set_change_session()
end_change_session()

---

* **void cancel_change_session()**

Отмена монопольного режима.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Закрывает сессию редактирования таблицы без сохранения внесенных изменений. При этом таблица загружается с диска.

---

* **void change_column_type(const AnsiString &,const AnsiString &)**

Изменить тип поля таблицы.

Параметры:

const AnsiString & - название колонки, тип которой необходимо изменить;
const AnsiString & - название нового типа колонки.

Возвращаемые значения:

Нет.

Комментарии:

При передаче параметров, тип поля указывется в формате, каким мы получаем его при использовании свойства types_list источника данных dbb_storeg. При невозможности изменить типа колонки (нельзя переконвертировать данные в указанный тип) будет сгенерировано сообщение об ошибке.

Смотри также:

types_list

---

* **void change_field(const AnsiString &,unsigned int,const Variant &)**

Изменение значение поля таблицы.

Параметры:

const AnsiString & - название колонки;
unsigned int - локальный индекс записи в источнике данных;
const Variant & - новое значение.

Возвращаемые значения:
Нет.
Комментарии:
Позволяет изменить значение поля таблицы. Доступно только в монопольном режиме доступа к таблице.
Смотри также:
begin_change_session()

---

* **bool check_eod()**

Проверка окончания потока данных.
Параметры:
Не передаются.
Возвращаемые значения:
True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.
Комментарии:
Определяет достигнуто ли окончание потока данных или нет и, в зависимости от результата, возвращает соответствующее значение.
Смотри также:
next_record()

---

* **void clear_filters()**

Отменить фильтрацию.
Параметры:
Не передаются.
Возвращаемые значения:
Нет.
Комментарии:
Позволяет отменить установленное ранее правило фильтрации.
Смотри также:
add_filter_set()
add_filter_range()
add_filter_ethalon()

---

* **void clear_sorters()**

Отменить сортировку.
Параметры:
Не передаются.
Возвращаемые значения:
Нет.
Комментарии:
Позволяет отменить установленное ранее правило и сортировки.
Смотри также:
add_sorter()

---

* **void create_column(const AnsiString &,const AnsiString &)**

Создать колонку.
Параметры:
const AnsiString & - название создаваемого поля;
const AnsiString & - тип создаваемого поля.
Возвращаемые значения:
Нет.
Комментарии:

Позволяет добавить новую колонку необходимого типа в таблицу. При передаче параметров, тип колонки необходимо передавать в формате, в каком мы получаем его при использовании свойства types_list.

Смотри также:

types_list

---

* **void create_record(unsigned int)**

Добавить запись.

Параметры:

unsigned int - глобальный индекс позиции, в которую будет добавлена новая запись.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет добавить запись в указанную позицию таблицы (доступно только при монопольном режиме доступа к таблице). В качестве параметра передается индекс позиции исходной таблицы (не источника данных).

---

* **void delete_record(unsigned int)**

Удалить запись.

Параметры:

unsigned int - глобальный индекс позиции в таблице, на которой находится удаляемая запись.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет удалить запись, находящуюся на указанной позиции в таблице (доступно только при монопольном режиме доступа к таблице). В качестве параметра передается индекс позиции в исходной таблице (не в источнике данных).

---

* **void end_change_session()**

Отмена монопольного режима доступа.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Если для указанной таблицы установлено свойствоautosave, то, при вызове данного метода, все внесенные изменения будут сохранены. Если свойствоautosave не установлено — для сохранения изменений перед закрытием сессии необходимо вызвать метод save_table().

Смотри также:

begin_change_session()
save_table()
autosave

---

* **void erase()**

Удалить запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет удалить запись, находящуюся на текущей позиции в источнике данных (доступно только при монопольном режиме доступа к таблице). В качестве параметра передается индекс позиции в источнике данных (не в исходной таблице).

---

* **void erase(unsigned int)**

Удалить запись.

Параметры:

unsigned int - локальный индекс позиции в источнике данных, на которой находится удаляемая запись.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет удалить запись, находящуюся на указанной позиции в таблице (доступно только при монопольном режиме доступа к таблице). В качестве параметра передается индекс позиции в исходной таблице (не в источнике данных).

---

* **void erase_all()**

Удалить всю таблицу.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет удалить всю таблицу из базы данных, не зависимо от выборки.

---

* **void erase_selection()**

Удалить данные в таблице.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет удалить данные из таблицы, котрые попадают под выборку (ранее примененное правило фильтрации, сортировки и т.п.).

---

* **AnsiString field_as_string(const AnsiString &)**

Получить значение текущей записи указанного поля в виде строки.

Параметры:

const AnsiString & - название поля таблицы.

Возвращаемые значения:

AnsiString - значение поля в виде строки.

Комментарии:

Позволяет получить значение текущей записи указанной колонки. Рекомендуется использовать этот метод до применения методов фильтрации add_filter_regexp().

Смотри также:

add_filter_regexp()
add_filter_regexp()

---

* **Variant field_val(const AnsiString &)**

Получить значение колонки текущей записи.

Параметры:
const AnsiString & - название поля таблицы.
Возвращаемые значения:
Variant - значение указанного поля текущей записи.
Комментарии:
Позволяет получить значение текущей записи указанной колонки.

---

* **bool find_first(const AnsiString &,const Variant &)**

Поиск данных (с начала).
Параметры:
const AnsiString & - название колонки для поиска;
const Variant & - искомое значение.
Возвращаемые значения:
True - если найдены необходимые данные;
False - если необходимые данные не найдены.
Комментарии:
Поиск данных в указанной колонке выполняется с начала таблицы. Используйте эту команду для ускорения поиска, если заведомо известно, что нужная информация находится ближе к началу таблицы.
Смотри также:
find_last()
find_next()

---

* **bool find_last(const AnsiString &,const Variant &)**

Поиск данных (с конца).
Параметры:
const AnsiString & - название колонки для поиска;
const Variant & - искомое значение.
Возвращаемые значения:
True - если найдены необходимые данные;
False - если необходимые данные не найдены.
Комментарии:
Поиск данных в указанной колонке выполняется с конца таблицы. Используйте эту команду для ускорения поиска, если заведомо известно, что нужная информация находится ближе к концу таблицы.
Смотри также:
find_first()
find_next()

---

* **bool find_next(const AnsiString &,const Variant &)**

Поиск данных (с текущей позиции до последней).
Параметры:
const AnsiString & - название колонки для поиска;
const Variant & - искомое значение.
Возвращаемые значения:
True - если найдены необходимые данные;
False - если необходимые данные не найдены.
Комментарии:
Используется для поиска данных с текущей позиции (“курсора” в таблице). Направление поиска — от текущей строки таблицы до последней.
Смотри также:
find_first()
find_last()

---

* **bool find_prev(const AnsiString &,const Variant &)**

Поиск данных (с текущей позиции до первой).

Параметры:

const AnsiString & - название колонки для поиска;
const Variant & - искомое значение.

Возвращаемые значения:

True - если найдены необходимые данные;
False - если необходимые данные не найдены.

Комментарии:

Используется для поиска данных с текущей позиции (“курсора” в таблице). Направление поиска — от текущей строки таблицы до первой.

Смотри также:

find_first()
find_last()

---

* **void first_record()**

Установить “курсор” в начало таблицы.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Установить “курсор” на первую запись в таблице.

Смотри также:

set_record()
position()

---

* **Variant get_autoinc(const AnsiString &)**

Получить автоинкремент.

Параметры:

const AnsiString & - название колонки таблицы

Возвращаемые значения:

Variant - Значение автоинкремента указанной колонки.

Комментарии:

Позволяет получить последнее присвоенное значение автоинкремента указанной колонки.

---

* **std::vector<AnsiString> get_columns_list()**

Получить список полей таблицы.

Параметры:

Не передаются.

Возвращаемые значения:

std::vector<AnsiString> - массив названий колонок таблицы.

Комментарии:

Позволяет получить список отображаемых в редакторе полей таблицы, не включая поля глобальных индексов, ссылочные поля, получаемые из других таблиц и т.п.

Смотри также:

get_fields_list()

---

* **std::vector<AnsiString> get_fields_list()**

Получить список всех полей таблицы.

Параметры:

Не передаются.

Возвращаемые значения:

std::vector<AnsiString> - массив названий всех колонок таблицы.

Комментарии:

Позволяет получить список всех полей таблицы, включая поля глобальных индексов, ссылочные поля, получаемые из других таблиц и т.п.

Смотри также:

get_columns_list()

---

* **AnsiString get_field_type(const AnsiString &)**

Получить тип поля таблицы.

Параметры:

const AnsiString & - название колонки таблицы.

Возвращаемые значения:

AnsiString - тип поля указанной колонки.

Комментарии:

Позволяет получить тип указанного поля таблицы. Тип получим в формате, в котором мы его получаем при использовании свойства types_list. Можно пулучить тип не только тех полей, которые отображаютс в редакторе таблицы, но также тип ссылочных полей, получаемых из других таблиц.

Смотри также:

get_fields_list()
types_list
type_view_name()

---

* **Variant get_default(const AnsiString &)**

Получить значение колонки, указываемое по умолчанию.

Параметры:

const AnsiString & - название колонки таблицы

Возвращаемые значения:

Значение, устанавливаемое по умолчанию для указанной колонки.

Комментарии:

Позволяет получить последнее присвоенное значение автоинкремента указанной колонки.

---

* **std::vector<Variant> get_linked_ids_list(const AnsiString &)**

Получить список допустимых значений ключевого поля.

Параметры:

const AnsiString & - название колонки таблицы

Возвращаемые значения:

std::vector<Variant> - массив допустимых значений ключевого поля, по которому строится ссылка.

Комментарии:

Позволяет получить список всех возможных значений ключевого поля.

---

* **std::vector<Variant> get_linked_sub_list(const AnsiString &)**

Получить список допустимых значений, подставляемых из ссылочной таблицы.

Параметры:

const AnsiString & - название колонки таблицы

Возвращаемые значения:

std::vector<Variant> - массив значений, которые отображаются в редакторе таблиц.

Комментарии:

Позволяет получить список всех значений, которые могут быть выбраны (подставленны) в указанное поле.

---

* **AnsiString get_table_name()**

Получить название таблицы.

Параметры:

Не передаются.

Возвращаемые значения:

AnsiString - название таблицы.

Комментарии:

Позволяет получить название талицы данных.

---

* **unsigned int global2local(unsigned int)**

Преобразовывает глобальный индекс записи в локальный.

Параметры:

unsigned int - глобальный индекс записи в таблице.

Возвращаемые значения:

unsigned int - локальный индекс записи в источнике данных.
Если преобразование невозможно — возвращает “-1”.

Комментарии:

Позволяет получить значение локального индекса записи источника данных, указав значение глобального индекса этой записи в таблице.

---

* **void insert(unsigned int,const std::vector<AnsiString> &,const std::vector<Variant> &)**

Вставка записи.

Параметры:

unsigned int - глобальный индекс создаваемой записи в таблице.
const std::vector<AnsiString> & - массив названий полей, значений которых необходимо указать при создании записи,
const std::vector<Variant> & - массив значений указанных полей.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет создавать запись с необходимыми параметрами в указанных полях. Все остальные поля заполняются либо автоинкрементным значением, либо значением по умолчанию.

---

* **bool is_autoinc(const AnsiString &)**

Позволяет узнать, является ли поле автоинкрементным.

Параметры:

const AnsiString & - название колонки.

Возвращаемые значения:

True - свойство автоинкремента установлено;
False - свойство автоинкремента не установлено.

Комментарии:

Позволяет узнать устанволено ли свойство автоинкремента для указанной колонкию.

---

* **bool is_column_linked(const AnsiString &)**

Позволяет узнать, является ли колонка ссылкой.

Параметры:

const AnsiString & - название колонки.

Возвращаемые значения:

True - колонка является ссылкой;
False - колонка не является ссылкой.

Комментарии:

Позволяет узнать, является поле ссылкой или нет, т.е. подставляются в это поле значение из ссылочных таблиц или нет.

---

* **bool is_indexed(const AnsiString &)**

Позволяет узнать,является ли колонка индексированоой.

Параметры:

const AnsiString & - название колонки.

Возвращаемые значения:

True - колонка индексированная;
False - колонка не индексированная.

Комментарии:

Позволяет узнать устанволено ли свойство индексации для указанной колонкию.

---

* **bool is_unique(const AnsiString &)**

Позволяет узнать, установлено ли свойство уникальности для указанной колонки.

Параметры:

const AnsiString & - название колонки.

Возвращаемые значения:

True - свойство уникальности установлено;
False - свойство уникальности не установлено.

Комментарии:

При устанвовленном свойстве уникальности все значения указанной колонки будут проверяться на уникальность.

---

* **void load_table()**

Перегрузить таблицу.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет загрузить таблицу с диска в оперативную память. Все сделанные изменения будут потеряны (доступно только в монопольном режиме использования таблицы).

---

* **unsigned int local2global(unsigned int)**

Преобразовывает локальный индекс записи в глобальный.

Параметры:

unsigned int - локальный индекс записи в источнике данных.

Возвращаемые значения:

unsigned int - глобальный индекс записи в таблице.
Если преобразование невозможно — возвращает “-1”.

Комментарии:

Позволяет получить значение глобального индекса записи таблицы, указав значение локального индекса этой записи в источнике данных.

---

* **void move_global_record(unsigned int,unsigned int)**

Переместить запись в таблице.

Параметры:

unsigned int - глобальный индекс записи в таблице;
unsigned int - глобальный новый индекс записи в таблице, куда эта запись будет перемещена.

Возвращаемые значения:

Нет.

Комментарии:

При использовании этого метода записи с глобальными индексами “старый индекс” и “новый индекс” поменяются местами. Если указан несуществующий глобальный индекс – будет сгенерировано сообщение об ошибке.

---

* **Variant next_autoinc(const AnsiString &)**

Увеличение значения автоинкремента.

Параметры:

const AnsiString & - название поля.

Возвращаемые значения:

Variant - значение автоинкремента.

Комментарии:

Позволяет увеличить значение автоинкремента. Для поля типа “целый” автоинкремент увеличивается на “1”, для поля типа “время” — на одну секунду и т.п.

---

* **void next_record()**

Перейти на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Переводит “курсор” на следующую запись в таблице (после текущей).

Смотри также:

check_eod()

---

* **unsigned int position()**

Получить номер текущей позиции “курсора”.

Параметры:

Не передаются.

Возвращаемые значения:

unsigned int - номер текущей позиции.

---

* **unsigned int push_back(const std::vector<AnsiString> &,const std::vector<Variant> &)**

Вставка записи в конец таблицы.

Параметры:

const std::vector<AnsiString> & - массив названий полей;
const std::vector<Variant> & - массив значений указанных полей.

Возвращаемые значения:

unsigned int - глобальный индекс созданной записи.

Комментарии:

Позволяет вставить запись в конец таблицы. При этом в указанных полях будут отображены указанные параметры. Во всех остальных полях будут вставлены значения автоинкремента либо значения по умолчанию.

---

* **unsigned int pushback_record()**

Вставка записи в конец таблицы.

Параметры:

Не передаются.

Возвращаемые значения:

Глобальный индекс созданной записи.

Комментарии:

Позволяет вставить запись в конец таблицы. При этом во всех полях созданной записи будут указаны значения автоинкремента или значения по умолчанию.

---

* **unsigned int record_count()**

Получить количество записей в источнике данных.

Параметры:

Не передаются.

Возвращаемые значения:

Общее количестов записей в источнике данных (не в таблице).

Смотри также:

global_record_count

---

* **void remove_column(const AnsiString &)**

Удалить колонку.

Параметры:

const AnsiString & - название необходимой колонки.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет удалить колонку из таблицы.

---

* **void rename_column(const AnsiString &,const AnsiString &)**

Переименовать колонку таблицы.

Параметры:

const AnsiString & - старое название поля;
const AnsiString & - новое название поля.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переименовать нужную колонку в таблице.

---

* **void reserve(unsigned int)**

Резервирование памяти.

Параметры:

unsigned int - количество записей.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет зарезервировать блок памяти при необходимости добавления в таблицу n записей (n - достаточно велико). При добавлении большого количества записей, не использовав предварительно этот метод, записи будут сохраняться в различных участках памяти. При использовании этого метода будет выделен необходимый блок памяти, в которм записи будут сохраняться “подряд”. Этот метод способствует избежанию фрагментации памяти.

---

* **void save_table()**

Сохранение таблицы.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет сохранить все изменения, сделанные в таблице. Доступно только в монопольном режиме использования таблицы. Используется перед закрытием сессии редактирования для сохранения изменений, если не установлено свойство autosave.

Смотри также:

end_change_session()
autosave

---

* **void set_autoinc(const AnsiString &,const Variant &)**

Установить значение автоинкремента.

Параметры:

const AnsiString & - название поля;
const Variant & - необходимое значение автоинкремента.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет изменить значение автоинкремента выбранного поля на необходимое значение.

Смотри также:

get_autoinc()
next_autoinc()

---

* **void set_autoincrement(const AnsiString &,bool)**

Установить/блокировать свойство автоинкремента.

Параметры:

const AnsiString & - название колонки таблицы;
bool:
True - установить автоинкремет;
False - отменить автоинкремент.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить или отменить установлку свойства автоинкремента для указанного поля таблицы.

---

* **void set_autosave(bool)**

Установить/блокировать автосохранение.

Параметры:

bool:
True - установить автосохранение;
False - отменить автосохранение.

Возвращаемые значения:

Нет.

Комментарии:

Если автосохранение не установлено, после изменения данных, перед закрытием сессии редактирования необходимо вызывать метод save_table().

Смотри также:

autosave

---

* **void set_change_session()**

Начать сессию редактирования таблицы (монопольное использование).

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Установить режим монопольного использования таблицы (открытие сессии редактирования таблицы). При этом доступ к таблице другим пользователям будет запрещен. Только в монопольном режиме возможно редактирование данных таблицы. При невозможности начать сессию редактирования будет сгенерировано соответствующее сообщение.

Смотри также:

change_field()
begin_change_session()
end_change_session()

---

* **void set_default(const AnsiString &,const Variant &)**

Установить значение по умолчанию.

Параметры:

const AnsiString & - название поля;
const Variant & - значение.

Возвращаемые значения:

Нет.

Комментарии:

Это значение будет автоматически указываться в поле при создании новой записи.

Смотри также:

get_default()

---

* **void set_indexed(const AnsiString &,bool)**

Установить/блокировать свойство индексированности.

Параметры:

const AnsiString & - название колонки;
bool:
True - установить свойство индексированности;
False - отменить свойство индексированности.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить свойство индексированности для указанной колонки таблицы.

---

* **void set_record(unsigned int)**

Установить позицию “курсора”.

Параметры:

unsigned int - локальный индекс записи в источнике данных.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить “курсор” на указанную запись в таблице, т.е. изменить текущую позицию “курсора”

* **void set_unique(const AnsiString &,bool)**

Установить/блокировать свойство уникальности.

Параметры:

const AnsiString & - название колонки;
bool:
True - установить свойство уникальности;
False - отменить свойство уникальности.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет установить свойство уникальности для указанной колонки таблицы. Каждая запись поля будет проверяться на уникальность.

* **void update(unsigned int,const std::vector<AnsiString> &,const std::vector<Variant> &)**

Редактирование записи.

Параметры:

unsigned int - локальный индекс записи в источнике данных;
const std::vector<AnsiString> & - массив названий редактируемых полей;
const std::vector<Variant> & - массив новых значений указанных полей источника данных.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет изменить необходимые поля записи.

* **void update(const std::vector<AnsiString> &,const std::vector<Variant> &)**

Редактирование текущей записи.

Параметры:

const std::vector<AnsiString> & - массив названий редактируемых полей;
const std::vector<Variant> & - массив новых значений указанных полей источника данных.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет изменить необходимые поля текущей записи.

Источник данных: any_table

Класс, который позволяет инициализировать любую доступную таблицу базы данных в качестве источника данных на этапе формирования отчета, что позволяет создавать универсальные отчеты, не привязанные к конкретной таблице.

Свойства

Все свойства источника данных **table_<имя таблицы>** (см. Источник данных: table_<имя таблицы> на стр. 134).

Методы

Все методы источника данных **table_<имя таблицы>** (см. Источник данных: table_<имя таблицы> на стр. 134).

* **void init(const AnsiString &)**

Инициализация таблицы.

Параметры:

const AnsiString & - название таблицы.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет инициализировать любую доступную таблицу базы данных для использования ее в качестве источника данных в ходе формирования отчета.

---

* **bool show_dialog(AnsiString &)**

Отображает диалоговое окно инициализации источника данных (таблицы).

Параметры:

AnsiString & - название таблицы

Возвращаемые значения:

True - если диалоговое окно создано;
False - при возникновении ошибки.

Комментарии:

Этот метод используется для передачи значений, выбранных пользователем, в отчет. Используется для источников данных (таблиц), не инициализированных при создании отчета (см. раздел Создание и удаление отчета на стр. 7.). Полученные параметры передаются в метод init().

Смотри также:

init()

Источник данных: user

Источник данных, предоставляющий информацию о пользователе, его лицензии, правах на выполнение операций и т.п. Хранение данных источника организовано в виде таблицы.

Свойства

* **int acl_id**
  Возвращает идентификатор “прав пользователя” в списке всех доступных “прав”.
* **int backend_id**
  Возвращает номер соединения с сервером.
* **AnsiString disp_host**
  Возвращает адрес диспетчерского сервера.
* **int disp_port**
  Возвращает номер порта диспетчерского сервера.
* **TDateTime expires**
  Возвращает дату окончания действия лицензии на доступ к серверу.
* **int group_id**
  Возвращает идентификатор группы мобильных объектов пользователя.
* **AnsiString host_id**
  Возвращает регистрационный номер пользователя (на сервере).
* **AnsiString login**
  Возвращает логин пользователя.
* **AnsiString org_name**
  Возвращает название организации, использующей данное программное

обеспечение. Это название было указано пользователем при регистрации программного обеспечния на сервере мобильных объектов.

* **AnsiString perm**
  Возвращает, в виде, строки разрешение на выполнение операции.
* **int perm_id**
  Возвращает идентификатор разрешения на выполнение операции.
* **int sym_id**
  Возвращает константу **sym_id**, которая ставится в соответствие каждому идентификатору разрешения **perm_id**.
* **int user_id**
  Возварщает идентификатор пользователя.
* **AnsiString user_name.**
  Возвращает имя пользователя, использующего данное програмное обеспечение (указывалось при регистрации программного обеспечения на сервере мобильных объектов).
* **unsigned int position**
  Возвращает номер текущей позиции.
* **unsigned int record_count**
  Возвращает общее количество записей в выборке.
* **int condition_id**
  Возвращает идентификатор события.
* **int custom_id**
  Возвращает идентификатор запроса

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет достигнуто ли окончание потока данных или нет и, в зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

---

* **bool is_active()**

Проверка доступности источника данных.

Параметры:

Не передаются.

Возвращаемые значения:

True - если источник данных доступен (даже если он не инициализирован);
False - если источник данных не доступен.

Комментарии:

Проверяет доступен ли источник данных в данном отчете.

---

* **void first_record()**

Переместиться на первую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на первую запись таблицы источника данных.

Смотри также:

next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.

Смотри также:

first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.

Параметры:

unsigned int - число, указывающее номер записи, на которую необходимо переместиться.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на необходимую запись в таблице источника данных.

Смотри также:

next_record()
first_record()

---

* **bool can_edit_group_plan(int)**

Проверка разрешения на редактирование плана.

Параметры:

int - идентификатор группы мобильных объектов.

Возвращаемые значения:

True - если пользователь может редактировать план для указанной группы мобильных объектов;
False - если редактироване не доступно.

Смотри также:

group_id

---

* **bool can_view_group_plan(int)**

Проверка разрешения на просмотр плана.

Параметры:

int - идентификатор группы мобильных объектов.

Возвращаемые значения:

True - если пользователь может просматривать план для указанной группы мобильных объектов;
False - если просмотр не доступен.

Смотри также:
group_id

---

* **bool is_permitted(int,int)**

Проверка разрешения на выполнение запроса.

Параметры:
int - идентификатор мобильного объекта;
int - идентификатор разрешения на выполнение операции (perm_id)

Возвращаемые значения:
True - если пользователь может выполнить запрос для данного мобильного бъекта;
False - если выполнение запроса не доступно.

Смотри также:
perm_id

---

* **int perm2sym(int)**

Получить системное значение (sym_id) разрешения на операцию.

Параметры:
int - идентификатор разрешения на выполнение операции (perm_id).

Возвращаемые значения:
Число (sym_id), соответствующее указанному идентификатору разрешения на операцию (perm_id).

Комментарии:
Позволяет получитть системное значение (sym_id), соответствующее указанному идентификатору разрешения на операцию.

Смотри также:
sym_id
sym2perm()

---

* **int condition2perm(int)**

Получить идентификатор разршения на операцию (perm_id).

Параметры:
int - идентификатор события.

Возвращаемые значения:
Идентификатор разрешения на операцию.

Комментарии:
Позволяет получить идентификатор разрешения на операцию в соответствии с указанной ситуацией.

Смотри также:
condition_id

---

* **AnsiString condition2str(int)**

Получить строку события.

Параметры:
int - идентификатор события.

Возвращаемые значения:
Текстовая строка события.

Смотри также:
condition_id

---

* **int custom2perm(int)**

Получить идентификатор разрешения на операцию.

Параметры:

int - идентификатор запроса.

Возвращаемые значения:

Идентификатор разрешения на операцию.

Комментарии:

Позволяет получить идентификатор разрешения на операцию в соответствии с указанным запросом.

Смотри также:

custom_id
custom2str()

---

* **AnsiString custom2str(int)**

Получить строку запроса.

Параметры:

int - идентификатор запроса.

Возвращаемые значения:

Текстовая строка запроса.

Смотри также:

custom_id
custom2perm()

---

* **int perm2condition(int)**

Получить идентификатор события.

Параметры:

int - идентификатор разрешения на операцию.

Возвращаемые значения:

Идентификатор события.

Комментарии:

Позволяет получить идентификатор события в соответствии с идентификатором разрешения на операцию.

Смотри также:

perm_id

---

* **int perm2custom(int)**

Получить идентификатор запроса.

Параметры:

int - идентификатор разрешения на операцию.

Возвращаемые значения:

Идентификатор запроса.

Комментарии:

Позволяет получить идентификатор запроса в соответствии с идентификатором разрешения на операцию.

Смотри также:

perm_id

---

* **bool perm_filter(const std::vector<int> &,int)**

Проверка разрешения выполнения запроса.

Параметры:

const std::vector<int> - массив идентификаторов мобильных объектов;
int - идентификатор разрешения на операцию (perm_id).

Возвращаемые значения:

True - если пользователь может выполнить запрос для указанных мобильных объектов;
False - если выполнение запроса не доступно.

Смотри также:
perm_id
is_permitted()

---

* **int sym2perm(int)**

Получить идентификатор разрещения на операцию (perm_id).

Параметры:
int - системное значение разрешения на операцию (sym_id).

Возвращаемые значения:
Идентификатор разрешения на операцию (perm_id), соответствующий указанному символьному значению (sym_id).

Комментарии:
Позволяет получитть идентификатор разрешения на операцию (perm_id), соответствующий указанному символьному значению.

Смотри также:
sym_id
perm2sym()

---

* **AnsiString sym2str(int)**

Получить разрешение на операцию в виде строки.

Параметры:
int - системное значение разрешения на операцию (sym_id).

Возвращаемые значения:
Разрешение на операцию в виде строки.

Комментарии:
Позволяет получитть разрешение на операцию, соответствующее указанному системному значению (sym_id).

Смотри также:
sym_id
sym2perm()

---

* **int sym_filter(std::vector<int> &,std::vector<int> &,int)**

Получить перечень мобильных объектов, для которых допустимо выполнение указанной операции.

Параметры:
std::vector<int> & - массив проверяемых мобильных объектов. В этом массиве будут сохранены идентификаторы мобильных объектов, для которых допустимо выпольнить указанную операцию;
std::vector<int> & - произвольный массив. В этом массиве будут сохранены идентификаторы мобильных объектов, для которых выполнение операции не допустимо;
int - системное значение разрешения на операцию (sym_id).

Возвращаемые значения:
-1 — если в перечне проверяемых мобильных объектов нет ни одного, для которого можно выполнить операцию;
0 — если для всех проверяемых мобильных объектов можно выполнить указанную операцию;
1 — если для некоторых мобильных объектов (в перечне проверяемых) можно выполнить указанную операцию, а для некоторых – нет.

Комментарии:

Позволяет определить для каких объектов возможно выполнить указанную операцию, а для каких – нет.

Смотри также:

sym_filter()
sym_id
perm2sym()

---

* **bool sym_filter(const std::vector<int>,int)**

Проверка, возможно ли выполнить указанную операцию хотя бы для одного мобильного объекта из указанных.

Параметры:

std::vector<int> & - массив идентификаторов проверяемых мобильных объектов.

int - системное значение разрешения на операцию (sym_id).

Возвращаемые значения:

True - если хотя бы для одного мобильного объекта, из массива, можно выполнить запрос;

False - для перечисленных объектов выполнение запроса не возможно.

Комментарии:

Позволяет определить для каких объектов возможно выполнить указанную операцию, а для каких – нет.

Смотри также:

sym_filter()
sym_id
perm2sym()

---

* **bool sym_is_permitted(int,int)**

Возможно ли выполнение указанного действия для мобильного объектаПроверка разрешения на выполнение действия для данного мобильного объекта.

Параметры:

int - идентификатор мобильного объекта;

int - системное значение идентификатора разрешения на выполнение операции (sym_id).

Возвращаемые значения:

True - если выполнение запроса для данного мобильного объекта допустимо;

False - если выполнение запроса не допустимо.

Комментарии:

Позволяет определить для каких объектов возможно выполнить указанную операцию, а для каких – нет.

Источник данных: distance_lines

Доступен только во время измерения расстояния на карте (см. Руководсто пользователя, раздел Измерение расстояния. на стр. 34). Позволяет получить доступ к различным данным, относящимся к определению расстояния. Хранение данных источника организовано в виде таблицы.

При измерении расстояния в окне ***Расстоянием*** *все сегменты записываются в порядке, обратном их появлению на карте, а в отчете - в порядке, согласно которому они появлялись на карте.*

Свойства

- **double distance**
  Возвращает длинну сегмента в текущей позиции таблицы данных.
- **bool is_selected**
  Возвращает *True* - если сегмент текущей записи таблицы выделен в окне **Расстояни**, или *False* в другом случае.
- **unsigned int position.**
  Возвращает номер текущей позиции “курсора”.
- **unsigned int record_count.**
  Возвращает общее количество записей в выборке.
- **double selected_distance**
  Возвращает длинну выделенных сегментов.
- **double total_distance**
  Возвращает общую длинну всех сегментов.
- **double x1**
  Возвращает x-координату начальной точки сегмента текущей записи таблицы.
- **double x2**
  Возвращает x-координату конечной точки сегмента текущей записи таблицы.
- **double y1**
  Возвращает y-координату начальной точки сегмента текущей записи таблицы.
- **double y2**
  Возвращает y-координату конечной точки сегмента текущей записи таблицы.

Методы

- **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет достигнуто ли окончание потока данных. В зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

- **void first_record()**

Переместиться на первую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” в таблице источника данных на первую запись.

Смотри также:

set_record()
next_record()

---

* **bool is_active()**

Проверка доступности источника данных.

Параметры:

Не передаются.

Возвращаемые значения:

True - если источник данных доступен (даже если он не инициализирован);
False - если источник данных не доступен.

Комментарии:

Проверяет доступен ли источник данных в данном отчете.

---

* **void next_record()**

Переместиться на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.

Смотри также:

set_record()
first_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.

Параметры:

unsigned int - число, указывающее номер записи, на которую необходимо переместиться.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на нужную запись в талице источника данных.

Смотри также:

first_record()
next_record()

---

* **void view_selected()**

Показать выделенные сегменты.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Используется для отображения на карте выделенных (в окне **Расстояние**) сегментов. При этом участок карты с выделеными

сегментами масштабируется и размещается таким образом, чтоб были видны все выделенные сегменты.

---

* **void view_total()**

Показать все сегменты.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Используется для отображения на карте всех сегментов. При этом участок карты масштабируется и размещается таким образом, чтоб были видны все сегменты

Источник данных: objinf_window

Источник информации, позволяющий получить доступ к данным окна доступных мобильных объектов (см. Руководсто пользователя, Глава 2 на стр. 25). Хранение данных источники организовано в виде таблицы.

Свойства

* **int first_selected**

  Возвращает идентификатор первого выделенного объекта в окне **Досупные мобильные объекты**.

* **bool is_selected**

  Возвращает *True* - если мобильный объект текущей записи таблицы выделен в окне **Доступные мобильные объекты**, или *False* в другом случае.

* **int obj_id**

  Возвращает идентификатор мобильного объекта текущей записи таблицы.

* **unsigned int position.**

  Возвращает номер текущей позиции “курсора”.

* **unsigned int record_count.**

  Возвращает общее количество записей в выборке.

* **int selected_count**

  Возвращает количество объектов, выделенных в окне **Доступные мобильные объекты**.

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет достигнуто ли окончание потока данных. В зависимости от результата, возвращает соответствующее значение.

Смотри также:

next_record()

---

* **void first_record()**

Переместиться на первую запись.

Параметры:
Не передаются.
Возвращаемые значения:
Нет.
Комментарии:
Позволяет переместить “курсор” в таблице источника данных на первую запись.
Смотри также:
next_record()
set_record()

---

* **bool is_active()**

Проверка доступности источника данных.
Параметры:
Не передаются.
Возвращаемые значения:
True - если источник данных доступен (даже если он не инициализирован);
False - если источник данных не доступен.
Комментарии:
Проверяет доступен ли источник данных в данном отчете.

---

* **void next_record()**

Переместиться на следующую запись.
Параметры:
Не передаются.
Возвращаемые значения:
Нет.
Комментарии:
Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.
Смотри также:
first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.
Параметры:
unsigned int - число, указывающее номер записи, на которую необходимо переместиться.
Возвращаемые значения:
Нет.
Комментарии:
Позволяет переместить “курсор” на нужную запись в таблице источника данных.
Смотри также:
first_record()
next_record()

Источник данных: plugin_version

Источник данных, позволяющий получить доступ к информации о плагинах для различных устройст сбора и передачи информации (см. Приложение на стр. 93). Хранение даных источника организовано в виде таблицы.

.

Свойства

* **AnsiString binary_file**
  Возвращает путь к загруженному файлу плагина текущей записи таблицы.
* **AnsiString binary_name**
  Возвращает имя файла плагина текущей записи таблицы.
* **int charset**
  Возвращает кодировку языка, при которой был “собран” плагин.
* **int commands_count**
  Возвращает количество команд управления, коотрые поддерживает устройство.
* **int dev_id**
  Возвращает идентификатор устройства сбора и передачи информации.
* **int internal_ver**
  Возвращает “внутреннюю” версию плагина.
* **int lang**
  Возвращает идентификатор языка, при котором был “собран” плагин.
* **AnsiString plugin_name**
  Возвращает название плагина в виде строки.
* **unsigned int position.**
  Возвращает номер текущей позиции “курсора”.
* **unsigned int record_count.**
  Возвращает общее количество записей в выборке.
* **int requests_count**
  Возвращает количество общих запросов, которые поддерживает устройство.
* **AnsiString version**
  Возвращает версии плагина в виде строки.
* **bool version_info_present**
  Возвращает *True* — если есть информация о плагине или *False* в другом случае.

Методы

* **bool check_eod()**

Проверка окончания потока данных.

Параметры:

Не передаются.

Возвращаемые значения:

True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.

Комментарии:

Определяет достигнуто ли окончание потока данных. В зависимости от результата, возвращает соответствующее значение.

---

* **void first_record()**

Переместиться на первую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” в таблице источника данных на первую запись.

Смотри также:

next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.

Смотри также:

first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.

Параметры:

unsigned int - число, указывающее номер записи, на которую необходимо переместиться.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на нужную запись в талице источника данных.

Смотри также:

next_record()
first_record()

---

* **int commands_id(unsigned int)**

Номер команды.

Параметры:

unsigned int - идентификато управляющей команды.

Возвращаемые значения:

Порядковый номер команды.

Комментарии:

Получить порядковый номер команды в перечне всех команд устройства сбора и передачи информации.

---

* **AnsiString commands_name(unsigned int)**

Получить название команды.

Параметры:

unsigned int - идентификатор управляющей команды.

Возвращаемые значения:

Строка названия команды управления.

Комментарии:

Плучить название управляющей команды устройства сбора и передачи информаци по ее идентификатору.

---

* **AnsiString get_version_parameter(AnsiString)**

Получить параметры версии плагина.

Параметры:

AnsiString - название параметра (в виде строки). Существует ряд стандартных параметров, таких как: CompanyName, FileDescription, FileVersion, ProductVersion, InternalName, ProductName и т.д.

Возвращаемые значения:

Значение указанного параметра в виде строки.

---

* **int requests_id(unsigned int)**

Получить номер запроса.

Параметры:

unsigned int - идентификатор запроса.

Возвращаемые значения:

Порядковый номер запроса.

Комментарии:

Получить порядковый номер запроса в перечне всех запросов устройства сбора и передачи информации.

---

* **AnsiString requests_name(unsigned int)**

Поучить название запроса.

Параметры:

unsigned int - идентификатор запроса.

Возвращаемые значения:

Название запроса в виде строки.

Комментарии:

Получить название запроса устройства сбора и передачи информации по его идентификатору.

Источник данных: binary_version

Свойства

* **AnsiString binary_file**
  Возвращает путь к загруженному выполняющемуся файлу (exe, dll, bpl) текущей записи таблицы, находящемуся в директории главного запускаещего файла программы.
* **AnsiString binary_name**
  Возвращает название загруженного выполняющегося файла (exe, dll, bpl) текущей записи программы, находящегося в директории главного запускаещего файла программы.
* **int charset**
  Возвращает кодировку, при которой был “собран” файл текущей записи таблицы.
* **AnsiString exe_file**
  Возвращает путь к запускающему файлу программы
* **AnsiString exe_name**
  Возвращает название запускающегося файла программы.
* **int lang**
  Возвращает язык, при котором был “собран” файл текущей записи таблицы.
* **unsigned int position.**
  Возвращает номер текущей позиции “курсора”.
* **unsigned int record_count.**
  Возвращает общее количество записей в выборке.**AnsiString version**

* **bool version_info_present**
  Возвращает *True* — если есть информация о плагине или *False* в другом случае.

Методы

* **int bin_charset(AnsiString)**

Параметры:
AnsiString - название (**binary_name**) или путь (**binary_file**) к файлу.
Возвращаемые значения:
Идентификатор кодировки, при которой был “собран” указанный файл.

---

* **int bin_lang(AnsiString)**

Параметры:
AnsiString - название (**binary_name**) или путь (**binary_file**) к файлу.
Возвращаемые значения:
Идентификатор языка, при котором был “собран” указанный файл.

---

* **AnsiString bin_version(AnsiString)**

Параметры:
AnsiString - название (**binary_name**) или путь (**binary_file**) к файлу.
Возвращаемые значения:
Версии указанного файла в виде строки.

---

* **AnsiString bin_version_parameter(AnsiString,AnsiString)**

Получить параметр версии файла.
Параметры:
AnsiString - название (**binary_name**) или путь (**binary_file**) к файлу.
AnsiString - название параметра (в виде строки). Существует ряд стандартных параметров, таких как: CompanyName, FileDescription, FileVersion, ProductVersion, InternalName, ProductName и т.д.
Возвращаемые значения:
Значение указанного параметра в виде строки.

---

* **bool check_eod()**

Проверка окончания потока данных.
Параметры:
Не передаются.
Возвращаемые значения:
True — если обнаружено окончание потока данных;
False — если окончание потока данных не обнаружено.
Комментарии:
Определяет достигнуто ли окончание потока данных. В зависимости от результата, возвращает соответствующее значение.

---

* **void first_record()**

Переместиться на первую запись.
Параметры:
Не передаются.
Возвращаемые значения:
Нет.
Комментарии:
Позволяет переместить “курсор” в таблице источника данных на первую запись.
Смотри также:

next_record()
set_record()

---

* **void next_record()**

Переместиться на следующую запись.

Параметры:

Не передаются.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на следующую за текущей запись в таблице источника данных.

Смотри также:

first_record()
set_record()

---

* **void set_record(unsigned int)**

Переместиться на необходимую запись.

Параметры:

unsigned int - число, указывающее номер записи, на которую необходимо переместиться.

Возвращаемые значения:

Нет.

Комментарии:

Позволяет переместить “курсор” на нужную запись в талице источника данных.

Смотри также:

next_record()
first_record()

---

* **AnsiString get_version_parameter(AnsiString)**

Получить параметры версии файла текущей записию.

Параметры:

AnsiString - название параметра (в виде строки). Существует ряд стандартных параметров, таких как: CompanyName, FileDescription, File-Version, ProductVersion, InternalName, ProductName и т.д.

Возвращаемые значения:

Значение указанного параметра в виде строки.

---

* **bool version_info_present(AnsiString)**

Проверка наличия информации о файле.

Параметры:

AnsiString - название (**binary_name**) или путь (**binary_file**) к файлу.

Возвращаемые значения:

True — если информация о файле присутствует;
False — если информации о файле нет.

Источник данных: map_zone

Свойства

* **bool empty**

Возвращает *True* если разрешенная зона не установлена и *False* в другом случае.

* **double latitude**
Возвращает широту установленной зоны.
* **double longitude**
Возвращает долготу установленной зоны.
* **double radius**
Возвращает радиус установленной зоны.

Методы

* **void set_empty(bool)**
Установить или отменить установку зоны.
Параметры:
True — разрешить режим установки зоны;
False — запретить режим установки зоны.
Возвращаемые значения:
Нет.
Комментарии:
Позволяет перейти в режим установки или отмены установки зоны. После использования этого метода можно использовать соответствующий метод установки координат радиуса, долготы и широты разрешенной зоны.
Смотри также:
set_values()

---

* **void set_values(double,double,double)**
Установить параметры зоны.
Параметры:
double — широта центра разрешенной зоны;
double — долгота центра разрешенной зоны;
double — радиус центра разрешенной зоны (в метрах).
Возвращаемые значения:
Нет.
Комментарии:
Позволяет указать координаты (долготу и широту центра, а также радиус разрешенной зоны). Доступно после использования метода void set_empty().
Смотри также:
set_empty()

1.11 Применение

Вот некоторые простые примеры использования интерпретатора:
1.Необходимо вывести сумму заказа: белым фоном, если сумма меньше 2000; зеленым, если сумма от 2000 до 10000; красным, если сумма больше 10000. Для этого в объекте с суммой необходимо набрать следующий скрипт:

```
if [Сумма] < 2000 then
  FillColor := clTransparent
else if [Сумма] < 10000 then
  FillColor := clGreen
else
  FillColor := clRed
```

Цвет можно задать числом, например:

FillColor := 128 + 128*256 + 128*65536 //(серый цвет)

2.Необходимо вывести те записи, для которых сумма заказа больше 2000. Для этого в бэнде необходимо набрать следующий скрипт (редактор можно вызвать из инспектора объектов):

if [Сумма] > 2000 **then**
Visible := 1 else
Visible := 0

3. Пользуясь функцией FreeSpace, которая возвращает размер свободного места на листе, и процедурой NewPage, которая начинает печать с новой страницы (или NewColumn – с новой колонки), можно управлять процессом формирования отчета.
Например: если места осталось меньше 30мм, начать печать со следующего листа. Для этого в скрипте нужного бэнда необходимо набрать следующее:

if FreeSpace * 5/18 < 30 **then** NewPage

4. Для того, чтобы вывести бэнд Report summary внизу страницы, используйте следующий код в скрипте этого бэнда:

CurY := PageHeight - Height

Приложение А

Масштаб, проекция, система координат.

В определении карты сказано, что одним из важнейших ее свойств является математический закон построения. Он проявляется в наличии масштаба и картографической проекциия.

Масштабом называется отношение длины линии на карте к длине соответствующей линии на земном шаре. Масштаб показывает, во сколько раз уменьшено картографическое изображение, сколько сантиметров на местности содержится в 1 см на карте. Например, масштаб 1:1 000 000 означает, что 1 см на карте соответствует 1 000 000 см на местности, т.е. I0 км. На картах дают не только числовой, но и линейный масштаб - отрезок масштабной линейки, удобный для выполнения измерений.

Картографическая проекция - это способ перехода от реальной, геометрикекси сложной земной поверхности к плоскости карты. Для этого вначале переходят к математически правильной фигуре элипсоида или шара, а затем проектируют изображение на плоскость, опять-таки с помощью строгих математических зависимостей. В результате каждой точке на земной шаре с широтой l и долготой f соответствует одна и только одна точка на карте с прямоугольными координатами x и y. Поэтому общее уравнение картографических проекций имеет вид: x=fI(f,l); y=f2(f,l).

Сферическую поверхность невозможно развернуть на плоскость без деформаций - сжатий и растяжений. Значит, всякая карта имеет те или иные искажения. Различают искажения длин, площадей, углов и форм. На крупномасштабных картах искажения могут быть практически неощутимы, но на мелкомасштабных они бывают очень велики. Картографические проекции обладают разными свойствами в зависимости от характнра и размера искажений.

Равноугольные проекции сохраняют без искажений углы и формы малых объектов, зато в них резко деформируются длины и площади объектов. По картам в равноугольных проекциях удобно, например, прокладывать маршруты судов и самолетов, но невозможно измерять площади.

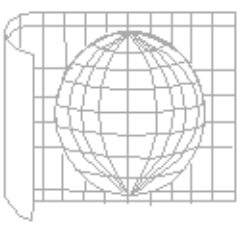
Равновеликие проекции не искажают площадей, но углы и формы объектов в них сильно искажены. Эти проекции хорошо приспособлены для определения площадей (например, размер государств, земельных угодий и др.).

Произвольные проекции имеют искажения длин, площадей и углов, но они распределяются по карте наиболее выгодным образом. Например выбирают проекции с минимальными искажениями в центральной чатси, зато они резко возрастают по краям карты. Среди произвольных проекций выделяются равнопромежуточные, в которых искажения длин отсутствуют по одному из направлений: либо вдоль меридиана, либо вдоль параллели.

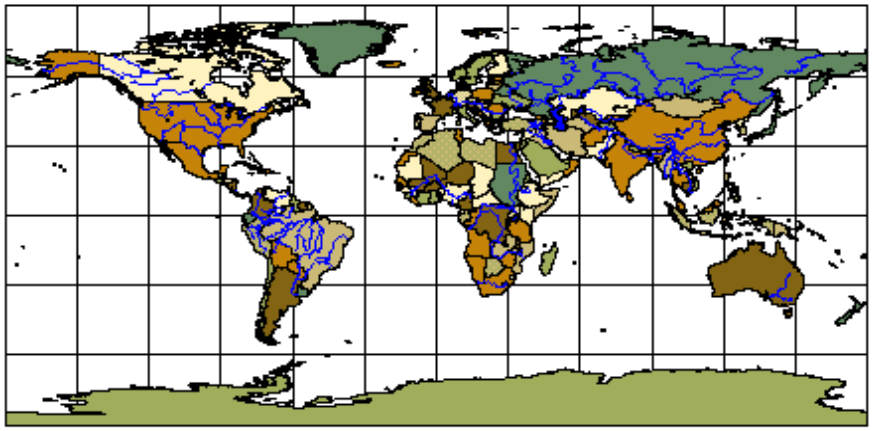
Проекции различают и по виду вспомогательной поверхности, используемой при переходе от эллипсоида или шара к плоскости карты.

Наиболее распространены проекции цилиндрические, когда проектирование с шара ведется как бы на поверхность цилиндра, конические и поликонические, когда вспомогательными поверхностями служат один или несколько конусов, и азимутальные, когда проектирование ведется на плоскость.

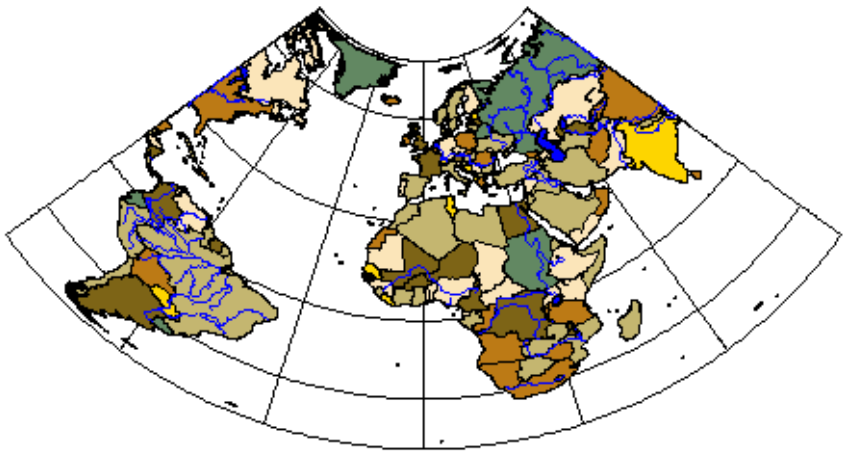
Ниже изображена цилиндрическая и коническая проекция.



Вот так выгляди земля в прямоугольной проекции.



А вот так в конической.

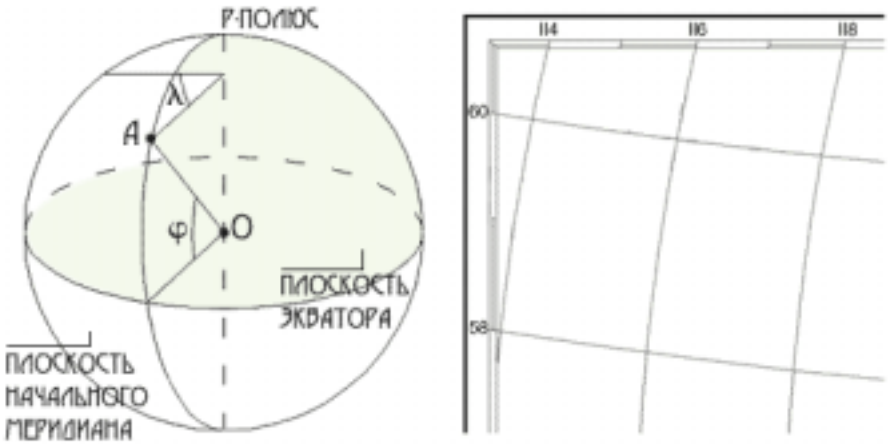


Для карт мира чаще всего применяют цилиндрические и различные условные проекции, обладающие наименьшими искажениями в области экватора и средних широт. Для карт Украины, России (и стран СНГ) обычно выбирают

конические проекции, в которых воображаемый конус сечет земной шар по параллелям 47° и 62° с.ш., - это так называемые линии нулевых искажений. Вблизи них искажения невелики. Это удобно, поскольку между указанными параллелями размещаются основные хозяйственные зоны стран и здесь сосредоточена максимальная нагрузка карт. Зато в конических проекциях сильно искажены районы, лежащие в высоких широтах и акватории Северного Ледовитого океана.

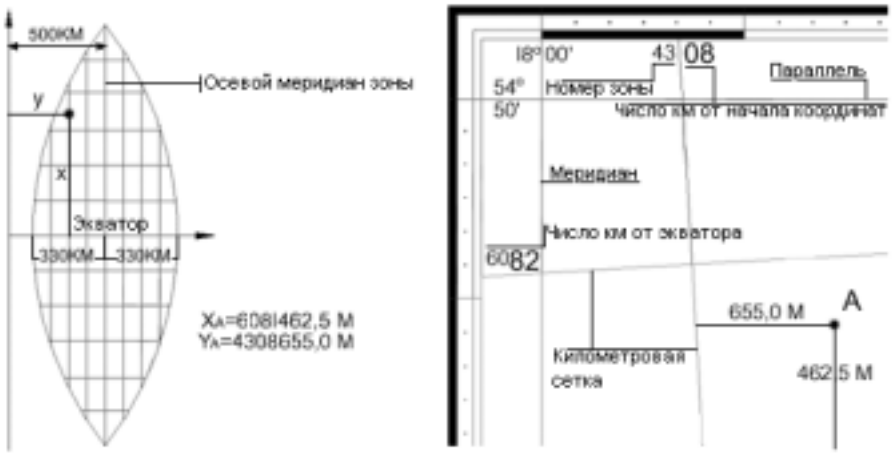
На мелкомасштабных картах главный масштаб (тот, что указан на карте) сохраняется только на линиях и в точках нулевых искажений, там, где вспомогательная поверхность касается (или сечет) шар. Во всех остальных частях карты масштаб несколько преувеличен или преуменьшен. Для определения частного масштаба в любом месте карты проводят специальные расчеты. При этом нужно знать длину какой-либо линии на шаре (например, расстояние между двумя городами, длину отрезка меридиана или параллели) и ту же длину на карте.

Географические координаты - угловые величины: широта и долгота, определяющие положение любой точки относительно экватора и начального (Гринвичского) меридиана. Широтой точки называется угол между плоскостью экватора и отвесной линией в данной точке. Долготой называется линейный угол двугранного угла, образованного плоскостью начального меридиана и плоскостью меридиана, проходящего через данную точку. На карту наносятся линии параллелей и меридианов. Параллель - это любая линия, все точки которой имеют одну и ту же географическую широту. Счет параллелей идет от экватора к северу и югу (от 0° до 90° с. или ю.ш.). Меридиан - это линия, все точки которой имеют одинаковую географическую долготу. По отношению к Гринвичскому меридиану различаются западные и восточные долготы (з.д. и в.д.), отсчитываемые от 0° и 180°.



Линии меридианов и параллелей образуют картографическую сетку. Обычно на рамках карты подписывают значения меридианов и параллелей и дают более дробные деления (например, через 5° или через I°) для удобства отсчета координат.

Прямоуголные координаты - система координат, в которой за ось абсцисс (X) принят осевой меридиан 6-градусной геодезической зоны, а за ось ординат (Y) - экватор.



Точка пересечения осевого меридиана и экватора (начало координат) имеет значения: X = 0 км; Y = 500 км. Это сделано для того, чтобы в пределах каждой зоны не было отрицательных значений Y. Величины X всегда положительны, так как территория Украины, России (и стран СНГ) расположена к северу от экватора.

На топографические карты нанесены линии, проведенные через 1 км и параллельные осям X и Y. Они образуют квадратную километровую сетку, которая служит для снятия с карты прямоугольных координат объектов или нанесения объектов по их координатам. У рамок карты подписаны значения линий километровой сетки в целых км. Для координат Y указан также номер зоны. Например, координата X = 6 08I 462,5 означает, что точка удалена от экватора на 6 08I км и 462,5 м, а Y = 4 308 655,0 показывает, что точка находится в 4-ой зоне и имеет координату 308 км и 655 м, т.е. удалена к западу от осевого мередиана на I9I км 345 м.

Сетка прямоугольных координат наносится только на топографические карты и используется при выполнении различных геодезических работ, вычмслении по картам расстояний, направлений, площадей объектов.

Приложение Б

Встроенная база данных

Смотри также:
Руководсто пользователя, раздел Работа с таблицами на стр. 87;
Источник данных: table_<имя таблицы> на стр. 134

Целостность таблиц

Изменение данных таблицы (источника данных) можно осуществлять как из редактора таблиц (см. Руководсто пользователя, раздел Работа с таблицами на стр. 87) так и программно (см. Источник данных: table_<имя таблицы> на стр. 134).
Изменение записи таблицы доступно только при монопольном использовании таблицы, что гарантирует целостность таблиц.
Монопольным режимом использования таблицы будем называть режим, при котором таблицу может редактировать одновременно только один пользователь. Остальные пользователи имеют возможность в данный момент времени просматривать эту таблицу.
Сохранение внесенных изменений в таблице (источнике данных) автоматически изменяет данные во всех источниках (таблице).

Редактирование

Изменение данных таблицы програмно происходит после открытия сессии редактирования (метод begin_change_session()). Открыть сессию можно только в том случае, если таблица не находиться в монопольном использовании. После сохранения изменений необходимо закрыть сессию редактирования (end_change_session()).
При вызове функций изменения данных таблицы (например, вставка записи) программа попытается открыть сессию редактирования автоматически. Если это не удается (таблица уже редактируется другим пользователем) – генерируется исключительная ситуация.
При внесении небольших корректив в данные таблицы/источника данных — сессия редактирования специально можно не открывать. При необходимости внесения больших изменений рекомендуется открыть сессию редактирования. Это уменьшит количество запросов программы для открытия/закрытия сессии редактирования.

Атомарность вызова функции

Вызов функции является атомарным, т.е. если не все операции, определенные вызовом, были завершены успешно, осуществляется откат всех внесенных изменений в таблице до состояния “перед выполнением вызова”.

Нумерация

Пользовательской таблице данных присуща сквозная (глобальная) нумерация. Все записи нумеруются от 0 до n-1 (n – количество записей в таблице).

Каждому источнику данных таблицы присуща локальная нумерация от 0 до n-1 (n – количество записей в источнике данных). В каждом источнике данных также храниться сквозная нумерация таблиц, т.е. каждая запись источника данных имеет локальный номер, а также номер, под которым она храниться в таблице. Для получения глобального номера записи необходимо использовать свойство oid.

Свойства фильтрации

После редактирования записи в выборке, к которой было применено определенное правило фильтрации, осуществляется проверка на соответствие данных измененной записи примененному фильтру. При несоответствии данных — запись изменяется в исходной таблице данных и удаляется из источника данных. Если “курсор” находился на удаленной записи – он переместиться в конец таблицы, т.е. за последнюю запись.
Фильтры накладываются друг на друга. При применении последовательно нескольких правил фильтрации к выборке — при каждой следующей фильтрации в качестве источника данных будет рассматриваться ранее отфильтрованная выборка.

Свойства сортировки

Сортировки накладываются друг на друга:
исходная выборка:

celii	celii1
1	2
2	6
1	8
1	7
2	5

выборка после сортировки “по возрастанию” поля *celii*:

celii	celii1
1	2
1	8
1	7
2	6
2	5

выборка после сортировки “по возрастанию” поля *celii1*:

celii	celii1
1	2
1	7
1	8
2	5
2	6

Свойства полей таблиц

Автоинкремент
Автоинкрементное поле увеличивает свое значение на единицу при добавлении новой записи. Не могут быть автоинкрементными поля типа *текст*, *булеан*, *цвет*, *графика*. Значение поля *целый* и *действительный* увеличивается на единицу, поле типа *время* — на 1 секунду, поля

Уникальность
Гарантирует уникальность данных каждой записи поля. При добавлении (изменении) записи будет осуществляться проверка на уникальность данных поля. При несоответствии — генерируется исключительная ситуация. Нельзя гарантировать уникальность полей типа *цвет* и *графика*.

Ссылочная целостность

При удалении (изменении) какого либо элемента ссылки – промежуточной таблицы, поля и т.п. – ссылка не сможет раскрыться, Генерируется исключительная ситуация.

Типы полей таблицы

целый
действительный
текст
булеан
цвет
время
дата
дата/время
графика

Алфавитный указатель терминов

Б

бэнд 18

В

виды отчетов
- Master-Detail-Detail 38
- без бэндов 38
- вложенный 37
- динамическиq 35
- многоколоночный 37
- разрываемые бэнды 36
- с группами 38
- с двумя уровнями 34
- с диаграммами 39
- с одним уровнем (список) 34
- с титульным листом 37
- с тремя уровнями 34
- cross-tab 35

выражение
- вставка 47

Д

диалоговые формы 49
- объекты 49
  - Button 50, 54
    - методы 54
    - свойства 54
  - CheckBox 50, 54
    - методы 55
    - свойства 55
  - ComboBox 50, 57
    - методы 57
    - свойства 57
  - DateEdit 50
  - Edit 50, 52
    - методы 53
    - свойства 52
  - Label 50, 52
    - методы 52
    - свойства 52
  - ListBox 50
    - методы 56
    - свойства 56
  - Memo 50, 53
    - методы 53
    - свойства 53
  - RadioButton 50, 55
    - методы 56
    - свойства 55
  - общие методы объектов 51
  - общие свойства объектов 51
- свойства диалоговых форм 58
- элементы управления 50

дизайнер отчетов 12
- интерфейс 12
- компоненты окна 12
- объекты 12
  - OLE объект 13, 27
  - RichText 12, 22
    - методы 23
    - свойства 23
  - бэнд 12, 18
    - источник данных 19
    - методы 21
    - свойства 20
  - вложенный отчет (SubReport) 12, 27
  - выбор объекта 12
  - диаграмма 13, 24
  - кросс-таб объект 13, 28
  - линия 12, 22
  - прямоугольник с тенью 23
    - методы 24
    - свойства 24
  - рисунок 12, 21
    - методы 22
    - свойства 22
  - текст 12, 14
    - методы 18
    - свойства 17
    - условное выделение 16
    - формат переменной 15
  - текст с тенью 13
  - фигура 12, 23
    - методы 23
    - свойства 23
  - флажок
    - методы 24
    - свойства 24
  - флажок (CheckBox) 13, 24

штрихкод 13, 26
методы 27
свойства 27
опции 42

И

инспектор объектов 44
источник данных 8, 46, 72
any_table 8, 73
binary_version 73, 164
методы 165
свойства 164
car_pj 8, 73
методы 73
свойства 73
current_fix_archive 8, 72, 88
методы 90
свойства 88
dataset_functions 8, 72, 123
методы 123
свойства 123
dataset_props 8, 72, 125
методы 125
свойства 125
datasets_list 8, 72, 128
методы 128
свойства 128
DB_Storadge 8, 129
методы 130
свойства 130
distance_lines 73, 157
методы 158
свойств 158
events_archive 8, 72, 73
методы 74
свойства 73
fix_archive 8, 72, 78
методы 79
свойства 79
fix_current 8, 73, 83
методы 85
свойства 84
fix_last 8, 73, 97
методы 97
свойства 97
img_process 8, 115
методы 116
свойства 116
map_proj 101
методы 102
свойства 101
map_window 8, 118
методы 118
свойства 118
map_zone 73, 166
методы 167
свойства 166
obj_inf 8, 107
методы 107
свойства 107
objinf_window 73, 160
методы 160
свойства 160
plugin_version 73, 161
методы 162
свойства 162
poll_error 8, 73, 110
методы 111
свойства 110
table_ 8
table_<имя таблицы> 73, 134
методы 135
свойства 134, 150
user 8, 73, 151
методы 152
свойства 151
virtual_dataset 8, 73, 127
методы 127
свойства 127
общие методы 64
общие свойства 64
свойства 64
экземпляр 9
параметры 9, 10
источник информации 8

К

категория 7
клавиши управления 32
класс
ARGS 68
as_array() 69
call() 69
clear() 69
count() 69
fill_Aray() 69
get_item() 69
put_item() 69
add() 68
конструктор выражений 47

О

отчет
виды см. виды отчетов 34
категория 7
параметры 39
делать два прохода 40
параметры страницы 40
передача информации 58
печать 33
просмотр 33
редактирование 11
редактор 12
создание 7
создание по шаблону 7
удаление 7, 11
управление построением
переменные 69, 70
системные переменные 72
функции 70

П

панель инструментов
выравнивание 32
прямоугольник 31
стандартная 28
текст 30
переменная
вставка 47
переменные 45, 61
процедура 63, 64

Р

редактор отчетов 12

С

словарь данных 44
стандартные объекты дизайнера отчетов
методы 14
свойства 13

У

управление мышью 33

Ф

функции
агрегатные 64
Avg() 64
Count() 65
Max() 65
Min() 65
Sum() 64
арифметические 67
Frac() 67
Int() 67
MaxNum() 68
MinNum() 68
Mod 68
Round() 67
работы со строками 65
Copy() 65
FormatDateTime() 66
FormatFloat() 66
If() 65
Length() 67
LowerCase() 66
NameCase() 67
Pos() 67
Str() 65
StrToDate() 66
StrToTime() 66
Trim() 67
UpperCase() 66
функция 63, 64
аргументы 48
вставка 48

Х-Я

языковые средства 59
использование процедур и функций 63
константы 62
модификация объектов 63
написание кода 61
обращение к объектам 62
переменные
использование 61
скрипты 60
соответствие типов данных 60