Інструкція користувача

Розділ 23

КОНСТРУКТОР

Версія 7.12.004

Сторінок 26

2020

ЗМІСТ

23. Конструктор 3
23.1. Створення користувальницького вікна 3
23.2. Дизайнер вікон 3
23.2.1. Компонент TISPROGraph 4
23.2.2. Компонент VirtualTree 5
23.2.3. Компонент XMLGrid 8
23.2.4. Компонент TISPROVirtualTree 13
23.2.5. Компонент TISPROXML 17
23.2.6. Компонент TISPROPopupMenu 18
23.2.7. Компонент TISPROMainMenu 18
23.2.8. Компонент TISPROBARCODE (штрих-код) 19
23.2.9. Компонент TISPRO Q RCODE (штрих-код) 20
23.2.10. Компонент TISPROMAIL 20
23.3. Користувальницькі реєстри 21
23.3.1. Опис структури XML-файлу 21
23.3.2. Додавання користувальницького реєстру 23
23.3.3. Редактор XML-файлу 23
23.4. Користувальницькі таблиці 24
23.5. Експорт та імпорт призначених для користувача вікон, таблиць та реєстрів 25
23.6. Резервне копіювання 25
23.7. Функції для роботи з користувальницькими вікнами і реєстрами 25

23. Конструктор

Модуль **Конструктор** призначений для розробки користувачами **Комплексу** реєстрів, модулів, таблиць зі своєю, особливою, логікою роботи. Модуль знаходиться в системі **Адміністратор.**

У модулі є три вкладки:

- **Вікна** - створення дизайну свого вікна, написання до нього логіки відображення і обробки даних;
- **Реєстри** - створення призначених для користувача реєстрів, списків, написання логіки для обробки таких дій як **Створити, Редагувати, Видалити, Копіювати, Групова операція, Подія користувача;**
- **Таблиці** - створення власних таблиць, полів, індексів.

23.1. Створення користувальницького вікна

Для створення нового користувальницького вікна необхідно в меню модуля **Конструктор** у вкладці **Вікна** вибрати пункт **Створення** (клавіша **Ins**). У вікні в полі **Номер** ввести унікальний номер вікна, а в поле **Найменування** - назву призначеного для користувача вікна. За кнопці **ОК** створюється новий запис в реєстрі. По кнопці **Дизайнер** відкриється дизайнер, в якому можна редагувати вигляд вікна, логіку обробки подій, які відображаються користувачу дані.

23.2. Дизайнер вікон

Це інструмент для створення користувацьких вікон. Дизайнер за своїм виглядом дуже нагадує середу розробки звітів Fast Report. Робочий простір розділено на три **області: Код, Ресурси, Вікно.**

В дизайнері вікон по кнопці **"Закоментувати / Розкоментувати текст"** (комбінація клавіш **Ctrl+К**) можна залишити / зняти коментарі. При створенні призначених для користувача форм по клавіші **F3** викликаються елементи програм.

На вкладці **Код** можна писати свою логіку відображення, обробки, збереження даних. Дизайнер підтримує мови PascalScript, Jscript, BasicScript, C ++ Script. Тут так само підтримується всі функції, поля, реєстри і довідники **Комплексу,** які доступні в звітах.

На вкладці **Ресурси** проводиться робота з запитами, читання даних з **Комплексу,** робота з тимчасовими таблицями, файлами DBF. На робочу область програми можна виносити такі елементи програми як ImageList (використовується для відображення іконок в компоненті ToolBar), запит ODBC , Тимчасова таблиця, Таблиця DBF .

На вкладці **Вікно** можна створювати свій дизайн вікна, виводити на форму такі стандартні елементи як Label, Edit, Button, Memo, RadioButton, ListView і інші. На за кладці **Дані** виведені спеціальні компоненти, у властивостях яких, можна вказати Dataset і поле, з якого отримувати дані. На вкладці **Розширені** містяться елементи для роботи з довідниками **Комплексу.** Вони аналогічні по своїй роботі з довідниками, які вбудовані в FastReport.

У дизайнера користувальницьких вікон доступний компонент **Граф** (TISPROGraph). Компонент може відображати схеми бізнес-процесів у вигляді документів підсистем і їх напрямок руху або довільні схеми роботи. Компонент має функції, за допомогою яких користувачі можуть самостійно, програмно, створювати графи, наповнювати та керувати ними.

23.2.1. Компонент TISPROGraph

Розташування: модуль **Конструктор,** вкладка **Розширені.**

Як приклад на малюнку вище показаний реалізований в підсистемі **Конструктор** модуль - **Зведений реєстр документів збуту і стану відвантаження,** призначений для контролю ходу прийому замовлень і відвантаження продукції покупцям, в тому числі і контролю за процесом руху первинних документів від підприємства покупцеві і повернення документів від покупця на підприємство.

1. Подвійний клік відкриває редактор графа.

У редакторі реалізовані можливості:

- завантаження з файлу
- об'єднання з файлу
- збереження в файл
- додавання вузла і зв'язку.

Додані константи атрибутів файлів. Реалізовані функції:

- **FileSearch** - Здійснює пошук файлу в одній або більше папках;
- **FindInDir** - Повертає список файлів в каталозі по шляху, включаючи маску файлів, із зазначенням атрибутів файлів.
- **FileGetAttr** - Повертає атрибути файлу;
- **FileSetAttr** - Встановлює атрибути файлу.

2. У редакторі подвійний клік на елементі відкриває вікно властивостей вузла або зв'язку.

Вікно властивостей вузла:

- Тема, його розташування і шрифт.
- Вид (Квадрат, коло, трикутник ...)
- Розміри.
- Кольори фону і кордонів
- Стилі заливки і рамки
- Розширені можливості пошуку (код та ідентифікатор документа)
- Вибирати зображення (іконку) з файлу
- Налаштовувати розташування іконки.
- Вибирати зображення з бібліотеки **Комплексу**.

Вікно властивостей зв'язку:

- Підпис і її розташування
- Види і розміри країв зв'язків.
- Стиль зв'язку
- Кольори ліній і заливки країв.

Функції для програмного доступу:

Завантажити з файлу: function LoadFromFile (F ileName: String): Integer
Об'єднати з файлом: function MergeFromFile (FileName: String): Integer
Зберегти у файл: function SaveToFile (FileName: String): Integer

Отримати ID натиснутого об'єкта: function GetClickedObject: int eger
Отримати ID натиснутого вузла: function GetClickedNode: integer
Отримати ID натиснутою зв'язку: function GetClickedLink: integer
Додати вузол: function AddNode (X, Y: Integer; Cap: String; ImageIndex: Integer): Integer
Присвоїти реквізити документа: procedure SetNodeDoc (NodeID, DocCd, DocRcd: Integer; Cap: String; ImageIndex: Integer): Integer
Отримати ID по реквізитами документа: function GetDocID (DocCd, DocRcd: Integer): Integer
Додати зв'язок: function AddLink (SourceID, TargetID: Integer): Integer
Отримати ID зв'язку з ID пов'язаних вузлів: function GetLinkID (SourceID, TargetID: Integer): Integer
Видалити елемент графа: procedure DeleteGraphObj (ID: Integer)
Отримати код документа по ID вузла: function GetDocCd (ID: Integer): Integer
Отримати ID документа по ID вузла: function GetDocRcd (ID: Integer): Integer
Встановити розмір шрифту вузла: procedure NodeFontSize (ID, Size: Integer)
Встановити колір шрифту вузла: procedure NodeFontColor (ID: Integer; color: TColor)
Встановити стиль вузла: procedure NodeBodyStyle (ID: Integer; style: TBrushStyle
Встановити колір вузла: procedure NodeBodyColor (ID: Integer; color: TColor);
Встановити стиль рамки вузла: procedure NodeBorderStyle (ID: Integer; style: TPenStyle)
Встановити колір рамки вузла: procedure NodeBorderColor (ID: Integer; color: TColor)
Встановити товщину рамки вузла: procedure NodeBorderWidth (ID: I nteger; value: Integer)
Встановити ширину вузла: procedure NodeWidth (ID: Integer; value: Integer)
Встановити висоту вузла: procedure NodeHeight (ID: Integer; value: Integer)
Встановити форму вузла: procedure NodeShape (ID: Integ er; value: Integer)
Встановити розмір шрифту зв'язку: procedure LinkFontSize (ID, Size: Integer)
Встановити колір шрифту зв'язку: procedure LinkFontColor (ID: Integer; color: TColor)
Встановити стиль початку зв'язку: procedure LinkBeg inStyle (ID, style: Integer)
Встановити розмір початку зв'язку: procedure LinkBeginSize (ID, Size: Integer)
Встановити стиль закінчення зв'язку: procedure LinkEndStyle (ID, style: Integer)
Встановити розмір закінчення зв'язку: procedure L inkEndSize (ID, Size: Integer)
Встановити товщину зв'язку: procedure LinkWidth (ID, Size: Integer)
Встановити стиль зв'язку: procedure LinkStyle (ID: Integer; style: TPenStyle)
Встановити текст зв'язку: procedure LinkText (ID: Integer; Text: String)
Встановити колір зв'язку: procedure LinkLineColor (ID: Integer; color: TColor)
Встановити колір заливки зв'язку: procedure LinkFillColor (ID: Integer; color: TColor)
Встановити колір шрифту колонки по імені поля колонки: Se tFieldFontColor
Встановити колір фону колонки по імені поля колонки: SetFieldBgColor
Встановити колір шрифту і фону для рядка: SetRowBgFntColor

Події:

Подвійне натискання на вузлі: OnNodeDblClick
Подвійне натискання на зв'язку: OnLinkDblClick
Натискання на вузлі: OnNodeClick
Натискання на зв'язку: OnLinkClick

23.2.2. Компонент VirtualTree

Компонент **VirtualTree** призначений для відображення деревовидних даних на користувальницької формі. У дизайнера вікон розміщений на вкладці **За умовчанням.**

Функції

Функції створення структури дерева

- **procedure AddNode (ID: Int64; ParentID: Int64; Text: String)**

Додає гілку з ім'ям Text в дерево, ID якій повинно бути унікальним.
ParentID - ID гілки, в яку буде додаватися нова гілка.

- **procedure AddColumn (Name: String)**

Додає у дерево нову колонку з ім'ям Name. Текст в гілках нової колонки встановлюється з першої колонки відповідної гілки.

Функції заповнення дерева

- **procedure SetText (NodeID: Int64; ColumnIndex: Int64; Text: String)**

Встановлює текст Text в осередку гілки з номером NodeID і колонки з номером ColumnIndex (нумерація починається з 0).

- **procedure SetColumnName (ColumnIndex: Integer; Name: String)**

Встановлює ім'я Name колонки з номером ColumnIndex.

- **procedure SetNodeCheck (ID: Int 64; Checked: Boolean)**
- Встановлює або знімає позначку з гілки з номером ID.

Функції навігації по дереву

- **procedure SetFocusedNode (ID: Int64)**

Встановлює курсор на гілці з номером ID, при цьому фокус колонки не змінюється.

- **procedure SetFocusedColumn (ColumnIndex: Integer)**

Встановлює курсор на колонці з номером ColumnIndex, при цьому фокус гілки не змінюється.

- **procedure ExpandNode (ID: Int64)**
- Розкриває гілку з номером ID.
- **procedure CollapsNode (ID: Int64)**
- Згортає гілку з номером ID.
- **procedure ExpandAll ()**
- Розкриває всі гілки.
- **procedure CollapsAll ()**
- Згортає всі гілки.

Функції зворотного зв'язку з деревом

- **function GetText (NodeID: Int 64; ColumnIndex: Int 64): String**
- Повертає текст в осередку, яка розміщена в гілці з номером NodeID і колонці з номером ColumnIndex.
- **functi on GetFocusedText (): String**
- Повертає текст у виділеній комірці.
- **function GetSelectedNodeID (): Int64**
- Повертає номер гілки, на якій знаходиться курсор.
- **function GetSelectedColumnIndex (): Int64**
- Повертає номер колонки, на якій знаходиться курсор.
- **fu nction GetNodeCheck (ID: Int64): Boolean**
- Повертає true або false, що відображають відзначена чи гілка дерева.
- **function GetNodeParentID (ID: Int 64): Int 64**

- Повертає ID батька гілки з номером ID.

Функції циклічного перебору структури дерева

- **function GetF irstNodeID (): Int64**
- Повертає ID першої гілки.
- **function GetNextNodeID (ID: Int 64): Int 64**
- Повертає ID наступної за переданої в параметрі функції гілки.
- **function GetNodeCount (): Int64**
- Повертає кількість гілок в дереві.
- **function GetColumnCount (): In t64**
- Повертає кількість колонок в дереві.

Код програми:

- *Заповнення дерева:*

```
procedure Button1OnClick (Sender: TISPROComponent);
var
 i: Int64;
begin
 VirtualTree1.AddNode (1,0, '1');
 Vi rtualTree1.AddNode (2,1, '11 ');
 VirtualTree1.AddNode (11,0, '2');
 VirtualTree1.AddNode (12,2, '111');
 VirtualTree1.AddNode (17,2, '112');
 VirtualTree1.AddNode (18,11, '21 ');
 VirtualTree1.AddColumn ( 'col');
end;
```

- тест в певній клітинці:

```
procedure Button2OnClick (Sender: TISPROComponent);
begin
VirtualTree1.SetText (2,1, 'dcdvdv');
end;
```

- Встановити фокус в вибраній комірці:

```
procedure Button5OnClick(Sender: TISPROComponent); procedure Button5OnClick (Sender: TISPROComponent);
begin begin
 VirtualTree1.SetFocusedColumn(1); VirtualTree1.SetFocusedColumn (1);
 VirtualTree1.SetFocusedNode (11);
 end;
```

- Отримати текст комірки:

```
procedure Button6OnClick (Sender: TISPROComponent);
begin
ShowMessage (VirtualTree1.GetText (2, 1));
end;
```

23.2.3. Компонент XMLGrid

Компонент **«TISPROGrid XML»** призначений для відображення даних в табличній формі з використанням XML - реєстру на користувальницької формі. В дизайнері вікон розміщений на вкладці **Розширені.**

Параметри компонента:

- **XMLFileDescription** - ім'я файлу XML -реестра:
  - MdID - код модуля
  - RstID - код реєстру
  - QueryID - код запиту

  Значення з імені файлу XML -реестра.

  Наприклад: U055_1_001.xml - MdID = 55, RstID = 1, QueryID = 1
- **GridID** - ідентифікатор реєстру.
- **XMLBaseDescription** - опис таблиці:
  - **BaseSynonym** - Ім'я таблиці. Синонім для таблиці в реєстрі.
  - **UniqueKeys** - Список полів, що становлять унікальний ключ записи. Обязательн ий параметр для позиціонування на реєстрі.
  - **MarkAbsField** - Ім'я поля, значення якого буде використовуватися для оцінки записів. Обязательн ий параметр для роботи із зазначеними записами.
  - **Mrk** - Код маркера реєстру (повинен бути більше 20 000).

Події:

- **OnClick** - приходить при будь-якій зміні фокуса в реєстрі, тобто при зміні осередки, в т.ч. в межах одного рядка;
- **OnShowRecord** - викликається при відображенні записи.
- **OnModFieldBegin** - викликається на початку редагування поля.
- **OnModifyField** - викликається після завершення редагування поля.
- **OnNewRecord** - викликається при створенні нового запису в реєстрі.
- **OnInsertRecord** - викликається при вставці запису до реєстру.
- **_ShowRecord** - компонент "Таблиця" видає це повідомлення перед завантаженням записи у внутрішній буфер. Оброблювач цього повідомлення повинен заповнити всі обчислюваності поля, що входять до запису таблиці. Тобто сформувати запис, кіт-я потім буде відображатися.
- **_BeforeRecord** - це повідомлення надсилається при позиціонуванні на запис, після того, як поточна запис на базі встановлена. Оброблювач цієї події повинен обробляти зв'язку таблиці з іншими елементами форми: оновлювати праву частину комбореестра, оновлювати поля підвалу, оновлювати пов'язані таблиці і т.п. Тобто ця подія - сигнал навігації по таблиці.

Функції:

Функції ініціалізації і інтерфейсу:

- **function Init: integer**

Ініціалізація роботи з реєстром.

Функція ініціалізує механізм і викликає побудова / заповнення реєстру в згідно з видом, зазначеним в параметрах.

Перед викликом обов'язково повинні бути заповнені параметри: GridID, MdID, RstID, QueryID.

- **function Refresh: integer**

Оновлює дані в реєстрі. Оновлення здійснюється з повторним запитом параметрів у користувача.

- **function RefreshNoAsk: integer**

Оновлює дані в реєстрі. Оновлення здійснюється з попередніми значеннями параметрів.

- **function RefreshNoClear: integer**

Оновлює дані в реєстрі без очищення реєстру. Можна застосовувати, наприклад, для оновлення не лише реєстру, а однієї або декількох записів, попередньо їх видаливши.

- **function SelectView: integer**

Виводить вікно вибору / встановлення виду реєстру. (Відповідає натискання Alt-0)

- **function RefreshRst: integer**

Оновлює дані в реєстрі. Оновлення здійснюється з повторним запитом параметрів і очищенням маркерів (відповідає натискання **Ctrl + F2)**

- **function ShowFindWindow: integer**

Показує вікно пошуку. Відповідає натискання **F7.**

- **function ShowFilterWindow: integer**

Показує вікно фільтра. Відповідає натискання **F11.**

- **function ShowFilterWindowCurrent: integer**

Показує вікно фільтра з заповненням значення за поточним полю. Відповідає натискання **Alt + F11.**

Функції роботи з таблицею реєстру:

- **function SetBegin: integer**

Встановити курсор на перший запис

- **function GetNext: integer**

Встановити курсор на наступний запис

- **function SetEnd: integer**

Встановити курсор на останній запис

- **function GetPrev: integer**
- Встановити курсор на попередній запис
- **function GetKey: integer**

Встановити курсор на запис у відповідності зі значенням ключа **UniqueKeys.**

- **function Delete: integer**

Видаляє поточний запис в реєстрі

- **procedure BeginUpdate**

Включає режим редагування реєстру, при якому дії з функціями роботи з реєстром не відображаються на екрані.

- **procedure EndUpdate**

Відключає режим редагування реєстру

- **function ShowTable: integer**

Переотображает реєстр.

Функції роботи з полями:

- **function ParamField (FieldName: String, FieldType: String, FieldLen: Byte, FieldScale: Byte, Value: String): integer**

  Заповнює поле-параметр XML -реестра.

  - **FldName** - ім'я поля. Повинна відповідати імені в розділі <params>.
  - **FldType** - тип поля. допустимі типи: "STRING", "INT", "NUM", "DATE", "TIME".
  - **FldLen** - довжина поля для типу STRING.
  - **FldScale** - місце точки для типу NUM.
  - **Value** - значення параметра.

- **function GetFieldValue (FieldName: String): String**

Отримати значення поля по імені

- **function SetFieldValue (FieldName: String, Value: String): integer**

Встановити значення поля по імені. Використовується для заповнення полів ключа. Поля записів в таблиці не змінює.

- **function SetFieldModifiable (FieldName: String, Modifiable: boolean): Boolean**

У станавливаются можливість редагування зазначеного поля в реєстрі. Працює тільки при ReadOnly = false.

- **Set Sort (FieldNames : String) : integer Set Sort (FieldNames: String): integer**

Встановити сортування по імені полів. FieldNames заповнюється наступним шаблоном: ім'я поля, напрямок сортування, ..., ім'я поля, напрямок сортування . Напрямок сортування може бути або по зростанню: 'A', або по спадаючій: 'D'

- **AllowSort (Allow: BOOLEAN): integer**

Дозволити / заборонити зміну сортування.

Гарячі клавіші.

F7 - Пошук
F11 - Фільтр
Alt + F11 - Фільтр за поточним
F12 - Налаштування сортування

Властивості:

ReadOnly - Ознака реєстру «тільки для читання». Для редагування записів прямо в реєстрі встановити в false.

Insertable - Ознака можливості вставки. При установці в true при русі курсору нижче останнього запису створюється нова.

CaptionLines - властивість визначає кількість рівнів в шапці таблиці. Заповнюється шапка як і раніше в XML-файл у властивості name тега <field>. Приклад: name = "Загальна шапка | Поле ". Рівні шапки поділяються символом «| ». Однакову назву декількох колонок об'єднує їх в одну.

FixedCols - Це властивість вказує кількість зафіксованих колонок з лівого краю таблиці.

Приклад:

// Робимо реєстр, що редагується

```
 GridXML 1. Refresh;
  GridXML1.ReadOnly: = false;
GridXML1.Insertable: = true;
Gr idXML1.SetFieldModifiable ( 'PtnCd', true);  // Поле  код  контрагента - редагується
 procedure GridXML1OnModifyField (Sender: TISPROComponent);
var
PtnCd, PtnNm: String;
 begin
PtnCd: = GridXML1.SetFieldValue ( 'PtnCd');  // Отримали новий код контрагента
 // ..... Отримали по ньому найменування, наприклад окремим запитом
 GridXML 1. SetFieldValue (PtnNm);  // Записали в реєстр найменування
end;
```

код програми:

```
procedure Form1OnShow (Sender: TISPROC omponent);
var
ret: integer;
begin
// Запуск
GridXML1.XMLFileDescription.MdID: = 555;
GridXML1.XMLFileDescription.RstID: = 0;
GridXML1.XMLFileDescription.QueryID: = 1;
 GridXML1.XMLBaseDescription.MarkAbsField: = 'Tes tRcd';
GridXML1.XMLBaseDescription.Mrk: = 20000;
GridXML1.XMLBaseDescription.UniqueKeys: = 'TestRcd';
 ret: = GridXML1.Init;
if (ret> 0) then
begin
// Параметр  в 0
ret: = GridXML1.ParamField ( 'TestID', 'INT', 0, 0, '0');
 if (ret> 0) then
GridXML1.Refresh;
 end
end;
// Кнопка "наступний запис"
procedure But ton1OnClick (Sender: TISPROComponent);
begin
GridXML1.GetNext;
end;
// Кнопка "попередній  запис "
procedure Button2OnClick (Sender: TISPROComponent);
begin
GridXML1.GetPrev;
end;
 procedure GridXML1OnDblClick (Sender: TISPROComponent);
var
Rcd, ret: integer;
begin
// Отримуємо  Ід  поточної  записи
```

```
Rcd: = StrToInt (GridXML1.GetFieldValue ( 'TestRcd'));
// Щось робимо із записом
// .....
// ODBCQuery 1.Open;
// .....
// Оновлюємо запис з сервера
ret: = GridXML1.ParamField ( 'TestID', 'INT', 0, 0, IntToStr (Rcd));  // Із потрібного запису
if (ret> 0) then
Begin
GridXML1.BeginUpdate;
GridXML1.Delete;  // Видаляємо
Gri dXML1.RefreshNoClear;  // оновлюємо  один  запис
GridXML1.EndUpdate;
 GridXML1.ParamField ( 'TestID', 'INT', 0, 0, '0');  // Обнуляємо  параметр
GridXML1.SetFieldValue ( 'TestRcd', IntToStr (Rcd));  // позиціонуємо  на  оновленому  записи
GridXML1.GetKey;
End;
 end;
procedure GridXML1OnKeyDown (Sender: TISPROComponent; var Key: Word; Shift: Integer);
begin
 if (Key = 13) then // При натисканні Enter дані поточного запису:
 ShowMessage ( 'Номер документа:' + GridXML 1. GetFieldValue ( 'TestNmr') + 'Сума:' +
GridXML 1. GetFieldValue ( 'TestSm'));
end; begin
end.
```

Використовуваний XML- реєстр:

U555_0_001.xml

```
<? xml v ersion = "1.0" encoding = "windows-1251"?>
<Query  name = "Документи призначені для користувача">
 <Result - fields>
 <field name = "Rcd" as = "TestRcd" olap = "" progressbar = "" progressbarright = ""
progressbarcolor = "" olapicon = "" calculated = "true" scale = "0" type = "int" prgres = " "p rgtype
=" 2 ">
<program> CashDoc_Rcd </ program> </ field>
 <field name = "Номер" as = "TestNmr" olap = "" progressbar = "" progressbarright = ""
progressbarcolor = "" olapicon = "" calculated = "true" scale = "20" type = "string" prgres = "
"prgtype =" 2 ">
<program> CashDoc_DocNmr </ pr ogram> </ field>
 <field name = "Дата" as = "TestDat" olap = "" progressbar = "" progressbarright = ""
progressbarcolor = "" olapicon = "" calculated = "true" scale = "20" type = "string" prgres = "
"prgtype =" 2 ">
<program> CashDocR.CashDoc_DocDat </ program> </ field>
 <field name = "Контрагент" as = "TestPtn" olap = "" progressbar = "" progressbarright = ""
progressbarcolor = "" olapicon = "" calculated = "true" scale = "20" type = "string" prgres = "
"prgtype =" 2 ">
<program> CashDoc_Ptn </ program> </ field>
<field name = " Сума " as = "TestSm" olap = "" progressbar = "" progressbarright = ""
progressbarcolor = "" olapicon = "" calculated = "true" scale = "20" type = "string" prgres = "
"prgtype =" 2 ">
```

```
<Program> Round (CashDoc_Sm, Svl_Acc) </ program> </ field>
 </ Result-fields>
 <Tables>
FROM CashDocR
LEFT JOIN Svl on SVl_Rcd = CashDoc_Val
</ Tables>
<Where>
(CashDoc_Rcd =: TestID OR: TestID = 0)
</ Where>
 <Params>
<param name = "ID документа " id = "TestID" type = "int" />
 </ Params>
</ Query>
```

23.2.4. Компонент TISPROVirtualTree

Компонент **TISPROVirtualTree** призначений для роботи з деревом з колонками. У дизайнера вікон на вкладці **Розширені**.

Функції:

Функції створення структури дерева:

- **procedure AddN ode (ID: Int64; ParentID: Int64; Text: String)**

Додає гілку з ім'ям Text в дерево , ID якій повинно бути унікальним.
ParentID - ID гілки, в яку буде додаватися нова гілка.

- **procedure AddColumn (Name: String)**

Додає в дерево нову колонку з ім'ям Name. Текст в гілках нової колонки встановлюється з першої колонки відповідної гілки.

- **procedure AddNumericColumn (Name: String; Triads: BOOLEAN)**
  Додає колонку числового типу. Triads - встановлює відображення числа в вигляді тріад.
- **procedure AddD ateTimeColumn (Name: String; MaskType: Integer)**
  Додає колонку типу «Дата». MaskType - встановлює тип відображення дати (1 - з часом, 2 - без часу)

Функції заповнення дерева:

- **procedure SetText (NodeID: Int64; ColumnIndex: Int64; Text: String)**
  Встановлює текст Text в осередку гілки з номером NodeID і колонки з номером ColumnIndex (нумерація починається з 0).
- **procedure SetNumericValue (NodeID : Int 64; ColumnIndex : Int 64; Value : Double)**
  Аналогічно з **SetText** встановлює значення типу Double для колонки з відповідним типом.
- **procedure SetDateTimeValue (ID : Int 64; ColumnIndex : Int 64; Value : TDateTime)**
  Аналогічно з **SetText** встановлює значення типу TDateTime для колонки з відповідним типом.
- **procedure SetColumnName (ColumnIndex: Integer; Name: Strin g)**

Встановлює ім'я Name колонки з номером ColumnIndex .

- **procedure SetNodeCheck (ID : Int 64; Checked : Boolean)** Встановлює або знімає позначку з гілки з номером ID .

Функції навігації по дереву:

- **procedure SetFocusedNode (ID: Int64)**

Встановлює курсор на гілці з номером ID , при цьому фокус колонки не змінюється.

- **procedure SetFocusedColumn (ColumnIndex: Integer)**

Встановлює курсор на колонці з номером ColumnIndex , при цьому фокус гілки не змінюється.

- **procedure ExpandNode (ID: Int64)**
  Розкриває гілку з номером ID.
- **procedure CollapsNode (ID: Int64)**
  Згортає гілку з номером ID .
- **procedure ExpandAll ()**
  Розкриває всі гілки.
- **procedure CollapsAll ()**
  Згортає всі гілки.

Функції зворотного зв'язку з деревом

- **function GetText (NodeID : Int 64; ColumnIndex : Int 64): String** Повертає текст в осередку, яка розміщена в гілці з номером NodeID і колонці з номером ColumnIndex.
- **'function GetFocusedText (): String'**
  Повертає текст у виділеній комірці.
- **function GetSelectedNodeID (): Int64**
  Повертає номер гілки, на якій знаходиться курсор.
- **function GetSelectedColumnIndex (): Int64**
  Повертає номер колонки, на якій знаходиться курсор.
- **function GetNodeCheck (ID: Int64): Boolean**
  Повертає true або false , що відображають відзначена чи гілка дерева.
- **function GetNodeParentID (ID : Int 64): Int 64** Повертає ID батька гілки з номером ID .
- **function GetNodeHeight (ID : Int 64): Integer** Повертає висоту Ноди з номером ID
- **Функція function GetCurrentField: String**
  Повертає ім'я поля поточної колонки.

Приклад :

```
procedure GridXML1OnDblClick (Sender: TISPROComponent);
begin
ShowMessage (GridXML1.GetCurrentField + '' + GridXML1.GetFieldValue
(GridXML1.GetCurrentField));
end;
```

Функції:

- **function SetColumnWidth (Індекс колонки, Ширина колонки)**
  Приклад використання: VirtualTree1.SetColumnWidth (0,200);

Функції циклічного перебору структури дерева

- **function GetFirstNodeID (): Int64**
  Повертає ID першої гілки.
- **function GetNextNodeID (ID : Int 64): Int 64** Повертає ID наступної за переданої в параметрі функції гілки.
- **function GetNodeCount (): Int64**
  Повертає кількість гілок в дереві.
- **function GetColumnCount (): Int64**
  Повертає кількість колонок в дереві.

Функції Редагування дерева

- **AllowManualEditing -** дозволяє режим редагування
- **procedure EditFocusedNode**
  Включає режим редагування поточної комірки
- **procedure SetColumnWidth (ColumnIndex : Integer ; Width : Integer)** Установка ширини обраної колонки
- **procedure SetColumnAlignment (ColumnIndex: Int64; CellAlignment: string)**
  Установка вирівнювання в осередку . Варіанти: 'caRight', 'ca Center ', 'ca Left '
- **procedure SetColumnPrecision (ColumnIndex: Int64; Precision: integer)**
  Установка точності чисел (кількості знаків після коми). Зауваження : кількість знаків після коми обмежує число тільки при відображенні, введене число зберігається в пам'яті.
- **procedure SetColumnStyle (ColumnIndex : Int 64; BackgroundColor , TextColor : TColor ; TextSize : Integer ; Bold , Italic : Boolean)** Установка стилю колонки: колір фону, тексту, розмір тексту (в пікселях), стиль шрифту (Bold - жирний, Italic - похилий)
- **procedure SetCellStyle (ID: Int64; ColumnIndex: Int64; BackgroundColor, TextColor: TColor; TextSize: Integer; Bold, Italic: Boolean)**
  Аналогічно SetColumnStyle, але застосовувана до осередку . Стиль комірки має перевагу перед стилем колонки.
- **procedure SetColumnMask (ColumnIndex: Int64; MaskType: Integer)**
  Установка типу відображення дати (застосовна до колонці типу « Дата »). MaskType: 1 - дата і час, 2 - тільки дата
- **procedure DeleteNode (ID : Int 64)** Видалення гілки дерева з ідентифікатором ID Якщо видаляється батьківський вузол то всі дочірні вузли теж видаляються.
- **procedure Clear**
  Очищення дерева
- **procedure SetNodeHeight (ID: Int64; value: Integer)**
  Установка висоти Ноди .

Властивості:
SummaryHeight - **висота підсумкового поля**

Можливості:

- швидкий пошук по колонці, на якій стоїть фокус.
- форматованого виведення чисел і дат, а також налаштування форматування комірки
- форматування комірки

Приклади:

Код програми:

- ***заповнення дерева***

```
procedure Button1OnClick (Sender: TISPROComponent);
var
 i: Int64;
begin
 VirtualTree1.AddNode (1,0, '1');
 VirtualTree1.AddNode (2,1, '11 ');
 VirtualTree1.AddNode (11,0, '2');
 VirtualTree1.AddNode (12,2, '111');
 VirtualTree1.AddNode (17,2, '112');
 VirtualTree1.AddNode (18,11, '21 ');
 VirtualTree1.AddColumn ( 'col');
end;
```

- ***встановити тест в певній клітинці***

```
procedure Button2OnClick (Sender: TISPROComponent);
begin
VirtualTree1.SetText (2,1, 'dcdvdv');
end;
```

- ***встановити фокус в вибраній комірці***

```
procedure Button5OnClick (Sender: TISPROComponent);
begin
VirtualTree1.SetFocusedColumn (1);
VirtualTree1.SetFocusedNode (11);
end;
```

- ***отримати текст комірки***

```
procedure Button6OnClick (Sender: TISPROComponent);
begin
ShowMessage (VirtualTree1.GetText (2, 1));
end;
```

Для компонента **TISPROVirtualTree** реалізовані події, які реагують на прокрутку колеса миші:

- OnMouseWheel,
- OnMouseWheelUp,
- OnMouseWheelDown.

Параметри відповідають аналогічним подіям в VCL, крім MousePos: TPoint, який перетворений в MousePos.X, MousePos.Y. Спрацьовують, тільки якщо VirtualTree в фокусі, але знаходження курсору на ньому не обов'язково.

23.2.5. Компонент TISPROXML

Модуль **Конструктор** - вкладка **Ресурси**. Подвійне натискання на компоненті відкриває дизайнер. Створений в дизайнері XML-документ зберігається всередині компонента.

XML--функції:

Відобразити дизайнер: procedure ShowDesigner
Завантажити з файлу: procedure LoadFromFile (const FileName: string)
Зберегти в файл: procedure SaveToFile (const FileName: string)
Завантажити з рядка: procedure ReadFromString (const Text: String)
Зберегти в рядок: function WriteToString: String
Створити новий xml: procedure New
Додати вузол до зазначеного: function NodeInsertBefore (FocusedNode: TISxXMLNode): TISxXMLNode
Додати вузол після зазначеного: function NodeInsertAfter (FocusedNode: TISxXMLNode): TISxXMLNode
Додати підлеглий: function NodeInsertChild (FocusedNode: TISxXMLNode): TISxXMLNode

Властивості:

Кореневий вузол: Root
Вузол TISxXMLNode

Властивості:

Найменування: Name: String
Значення:Value: String
Кількість підлеглих вузлів: NodeCount: Integer;
Отримати вузол за індексом: Nodes [i]: TISxXMLNode (тільки для читання)
Кількість атрибутів: AttributeCount: Integer
Отримати атрибут за індексом: Attributes [i]: TISxXMLAttribute (тільки для читання)
Значення атрибута по імені: AttributeValueByName [Name]: String

Функції:

Додати атрибут: procedure AttributeAdd (Name: String, Value: String)
Видалити атрибут за індексом: procedure AttributeDelete (Index: integer)
Додати вузол: procedure NodeAdd (Name: String, Value: String)
Видалити вузол за індексом: procedure NodeDelete (Index: integer)
Знайти вузол на ім'я: function FindNode (const NodeName: String): TISxXMLNode

Атрибут TISxXMLAttribute

Найменування: Name: String
Значення: Value: String

Приклад використання:

```
procedure Button1OnClick (Sender: TISPROComponent);
var
Node: TISxXMLNode;
i: integer;
begin
XML1.Root.AttributeAdd ( 'A0', '1');
Node: = XML1.Root.NodeAdd ( 'Node1', '2');
Node.AttributeAdd ( 'A1', '2');
XML1.ShowDesigner;
for i: = 0 to XML1.Root.NodeCount-1 do
 ShowMessage (XML1.Root.Nodes [i] .Name + '' + XML1.Root.Nodes [i] .Value);
end;
```

23.2.6. Компонент TISPROPopupMenu

Компонент створення контекстного меню. Знаходиться на **вкладці Стандартні .**

Створення структури меню. Створення пунктів і підпунктів меню проводиться натисненням правої кнопки миші по значку контекстного меню або значку елемента меню у вікні структури модуля і вибору відповідного пункту.

- Натискання на значок контекстного меню. При виборі пункту «**Підпункт**» створиться підпункт меню в кінці першого рівня контекстного меню.
- Натискання на значок пункту меню.
  1. **Пункт:** створиться пункт меню перед обраним пунктом на цьому ж рівні;
  2. **Підпункт** : створиться пункт меню, як підпункт обраного в кінці списку.

  Пункти **Вгору** і **Вниз** дозволяють переміщати пункти меню відповідно вгору або вниз.

Властивості контекстного меню:

- ImageList: підключення ImageList, створеного в **Ресурсах** , до меню;
- Для підключення контекстного меню до компонентів, в їх властивості PopupMenu потрібно вибрати необхідне меню.

Властивості пунктів меню:

- AutoCheck: якщо властивість встановлено в true, то при натисканні на пункті меню в запустив додаток, властивість змінюється на протилежне;
- Bitmap: біля пункту меню малюється вибране зображення;
- Checked: якщо властивість встановлено в true, то біля пункту меню малюється значок галочки;
- ImageIndex: якщо до контекстного меню підключений ImageList, то біля пункту меню малюється зображення під номером ImageIndex в підключеному ImageList-ті.

23.2.7. Компонент TISPROMainMenu

Компонент створення **Головне меню** . Знаходиться на **вкладці Стандартні**

Створення структури меню. Створення пунктів і підпунктів меню проводиться натисненням правої кнопки миші по значку головного меню або значку елемента меню у вікні структури модуля і вибору відповідного пункту.

- Натискання на значок головного меню
  При виборі єдино можливого пункту **Підпункт** створиться підпункт меню в кінці першого рівня головного меню.
- Натискання на значок пункту меню
  1. **Пункт:** створиться пункт меню перед обраним пунктом на цьому ж рівні;
  2. **Підпункт:** створиться пункт меню, як підпункт обраного в кінці списку.

Властивості головного меню:

- ImageList : підключення ImageList , створеного в **Ресурсах** , до головного меню.

Властивості пунктів меню:

- AutoCheck: якщо властивість встановлено в true, то при натисканні на пункті меню в запустив додаток, властивість змінюється на протилежне;
- Bitmap: біля пункту меню малюється вибране зображення;
- Checked: якщо властивість встановлено в true, то біля пункту меню малюється значок галочки;
- ImageIndex: якщо до головного меню підключений ImageList, то біля пункту меню малюється зображення під номером ImageIndex в підключеному ImageList-ті.

23.2.8. Компонент TISPROBARCODE (штрих-код)

Знаходиться на **вкладці Розширені**. Компонент TISPROBarcode.

Властивості:

- **BarType (TISxBarcodeType)** - тип кодування штрих-коду:
  1. bcCode _2_5_ interleaved ;
  2. bcCode _2_5_ industrial ;
  3. bcCode _2_5_ matrix ;
  4. bcCode 39;
  5. bcCode 39 Extended ;
  6. bcCode 128;
  7. bcCode 128 A ;
  8. bcCode 128 B ;
  9. bcCode 128 C ;
  10. bcCode 93;
  11. bcCode 93 Extended ;
  12. bcCodeMSI ;
  13. bcCodePostNet ;
  14. bcCodeCodabar ;
  15. bcCodeEAN 8;
  16. bcCodeEAN 13;
  17. bcCodeUPC _ A ;
  18. bcCodeUPC _ E 0;
  19. bcCodeUPC _ E 1;
  20. bcCodeUPC _ Supp 2;

21. bcCodeUPC _ Supp 5;
22. bcCodeEAN 128;
23. bcCodeEAN 128 A ;
24. bcCodeEAN 128 B ;
25. bcCodeEAN 128 C .

- **CalcCheckSum (Boolean) - обчислювати / НЕ обчислювати контрольну суму;**
- **Rotation (Integer)** - поворот штрих-коду на вказаний кут, кратний 90 ^{0} ;
- **ShowText (Boolean)** - показувати / НЕ показувати кодований текст під штрих-кодом;
- **Text (String)** - кодований текст;
- **WideBarRatio (Extended)** - співвідношення ширини товстого і тонкого штриха;
- **Zoom (Extended)** - масштаб.

23.2.9. Компонент TISPRO Q RCODE (штрих-код)

Знаходиться на вкладці **Розширені**. Компонент TISPROQRCode.

Властивості:

- **Encoding(TQRCodeEncoding)** - тип кодування QR -коду;
  1. qrAuto;
  2. qrNumeric;
  3. qrAlphanumeric;
  4. qrISO88591;
  5. qrUTF8NoBOM;
  6. qrUTF8BOM;
  7. qrShift_JIS.
- **FontScaled (Boolean)** - масштабувати / НЕ масштабувати підпис QR - коду;
- **Rotation (Integer)** - поворот штрих-коду на вказаний кут, кратний 90 ^{0} ;
- **ShowText (Boolean)** - показувати / НЕ показувати кодований текст під штрих-кодом;
- **Text (String)** - кодований текст;
- **PixelSize (Integer)** - розмір пікселю;
- **Zoom (Extended)** - масштаб.

23.2.10. Компонент TISPROMAIL

Знаходиться на **вкладці Розширені**. Компонент TISPROMail.

Функції:

- **function Connect (TargetHost: AnsiString; TargetPort: AnsiString; UserName: AnsiString; Password: AnsiString): boolean -** підключення до хосту поштової служби (повертає true при вдалому підключенні);
- **procedure Disconnect -** відключення від хоста ;
- **function GetEMailsCount : Integer -** повертає кількість листів в поштовій скриньці;

- **function GetEMail (EMailIndex : Integer ; Delete : boolean): Boolean -** повертає true у разі успішного завантаження листи. У функцію передається:
  1. номер листа по списком;
  2. Чи видаляти лист з сервера після зачитуванні.
- **procedure SaveAttachByPath (i : Integer ; path : String) -** отримати додаток до листа (в разі наявності в тексті листа html розмітки у вкладення додається html -файл з цим вмістом). Параметр **path** визначає шлях, по якому зберігати файл (якщо шлях не зазначений або такого шляху не існує, то з'явиться вікно вибору шляху);
- **function GetAttachesCount : Integer -** повертає кількість вкладень в листі.

Властивості:

- **EMailTo -** ім'я одержувача листа ;
- **EMailFrom -** ім'я відправника листа;
- **EMailSubject -** тема листа;
- **EmailText -** текст листа.

Код програми отримання імені одержувача, імені відправника, теми і тексту листа:

```
procedure Button1OnClick (Sender: TISPROComponent);
var
i, j: integer;
begin
if (Mail1.Connect ( 'mail.intelserv.kiev.ua', '110', ' ******* ', ' ******** ')) then
begin
for i: = 0 to Mail1.GetEMailsCount - 1 do
begin
if (Mail1.GetEmail (i, false)) then
begin
ShowMessage ( 'To:' + Mail1.EMailTo);
ShowMessage ( 'From:' + Mail1.EMailFrom);
ShowMessage ( 'Subject:' + Mail1.EMailSubject);
ShowMessage ( 'Text:' + Mail1.EMailText);
 for j: = 0 to Mail1.GetAttachesCount - 1 do
Mail1.SaveAttachByPath (j, 'D: \\ tmpfiles ');
end;
end;
end;
end ;
```

23.3. Користувальницькі реєстри

23.3.1. Опис структури XML-файлу

Для створення призначеного для користувача XML - реєстру в першу чергу необхідно створити XML - файл з описом полів, таблиць та умов відображення даних. Приклад призначеного для користувача реєстру:

```
<? Xml version = "1.0" encoding = "windows-1251"?>
<Query name = "Документи призначені для користувача">
```

```
<Result-fields>

<Field name = "Rcd" as = "CashDoc_Rcd" olap = "" progressbar = "" progressbarright = "" progressbarcolor =
"" olapicon = "" calculated = "true" scale = "0" type = "int" prgres = " "prgtype =" 2 "visible =" false ">
<Program> CashDoc_Rcd </ program> </ field>

<Field name = "Номер" as = "CashTestNmr" olap = "" progressbar = "" progressbarright = "" progressbarcolor
= "" olapicon = "" calculated = "true" scale = "20" type = "string" prgres = " "prgtype =" 2 ">
<Program> CashDoc_DocNmr </ program> </ field>

<Field name = "Дата" as = "CashTestDat" olap = "" progressbar = "" progressbarright = "" progressbarcolor =
"" olapicon = "" calculated = "true" scale = "20" type = "string" prgres = " "prgtype =" 2 ">
<Program> CashDocR.CashDoc_DocDat </ program> </ field>

<Field name = "Контрагент" as = "CashTestPtn" olap = "" progressbar = "" progressbarright = ""
progressbarcolor = "" olapicon = "" calculated = "true" scale = "20" type = "string" prgres = " "prgtype =" 2 ">
<Program> CashDoc_Ptn </ program> </ field>

<Field name = "Сума" as = "CashTestSm" olap = "" progressbar = "" progressbarright = "" progressbarcolor = ""
olapicon = "" calculated = "true" scale = "20" type = "string" prgres = " "prgtype =" 2 ">
<Program> Round (CashDoc_Sm, Svl_Acc) </ program> </ field>

</ Result-fields>

<Tables>
FROM CashDocR
LEFT JOIN Svl on SVl_Rcd = CashDoc_Val
LEFT JOIN SprPls ON SprPls_Rcd = CashDoc_Sch
LEFT JOIN Jr ON Jr_Rcd = CashDoc_JrnRcd
LEFT JOIN DogSprOtv ON DogSprOtv_Rcd = CashDoc_Otv
LEFT JOIN SOu ON Ou_Rcd = CashDoc_CdOu
</ Tables>
<Where>
(CashDoc_Rcd =: prmValidRec OR: prmValidRec = 0)
</ Where>
 <Params>
<Param name = "ID документа" id = "prmValidRec" type = "int" />
 </ Params>
</ Query>
```

Опис параметрів файлу:

- *Query name* - назва реєстру. Відображатиметься в списку вікна Види реєстру (Alt +0) ;
- Блок < *result - fields* > </ *result - fields* > - між цими тегами йде опис всіх полів, які є в реєстрі;
- Блок *<field name = " Номер " as = "CashTestNmr" olap = "" progressbar = "" progressbarright = "" progressbarcolor = "" olapicon = "" calculated = "true" scale = "20" type = "string" prgres = "" prgtype = "2"> <program> CashDoc_DocNmr </ program> </ field>* - опис поля в реєстрі :
  - *name = "Номер"* - назва поля, буде відображатися в шапці таблиці;

- *as = " CashTestNmr "* - назва поля, з яким надалі можна буде працювати з калькуляцій;
- *scale = "20"* - довжина поля, для поля типу int довжина завжди дорівнює нулю;
- *type = " string "* - тип поля . Може бути string, int;
- *prgtype = "2"* - тип обробки програми . 1 - BasicScript, 2 - SQL запит ;
- *visible = " false "* - вказує чи буде поле видимим в реєстрі для користувача;
- *<program> CashDoc_DocNmr </ program>* - програма отримання поля ;

- Блок < *tables* > *</ tables* > - описуються таблиці, з яких потрібно отримувати поля, а так же ключі зв'язування таблиць. Синтаксис аналогічний SQL запит ам ;
- Блок < *where* > *</ where* > - в цьому блоці описуються умови вибірки даних із зазначених в тезі < tables > таблиць. У цьому блоці повинна бути обов'язково вказана рядок *(CashDoc_Rcd =: prmValIdRec OR: prmValIdRec = 0)*, де *CashDoc_Rcd* це унікальний запис (Rcd) в таблиці ;
- Блок < *params* > *</ params* > містить список переданих параметрів з **Комплексу** в запит. У всіх призначених для користувача реєстрах блок повинен містити як мінімум один параметр, як в прикладі;

Імена для користувача XML файлів завжди повинні починатися з букви U (наприклад, U047_0_001.xml). Створений файл потрібно помістити в папку XML серверної частини **Комплексу** .

23.3.2. Додавання користувальницького реєстру

Для створення нового користувальницького реєстру необхідно в модулі **Конструктор** на вкладці **Реєстри** натиснути пункт меню **Створити** (клавіша **Ins**). У діалоговому вікні в полі **Номер** вказати унікальний код реєстру, в полі **Найменування** - назва призначеного для користувача реєстру. В поле **Файл** вибрати файл XML - реєстру, назва якого починається на символ U і знаходиться в папці XML серверної частини **Комплексу.** В поле **Таблиця** потрібно вказати головну таблицю, з якої буде проводитися побудова реєстру, в полі **Ід таблиці** вказати поле, яке містить унікальний запис головної таблиці (Rcd таблиці). В поле **Маркер** ввести код для маркування списку в реєстрі. Вводяться числа від 10000, значення повинно бути унікальним для всіх призначених для користувача реєстрів. В поле **Форма** вводиться код екранної форми , значення поля може бути від 5000 до 10000. Це поле служить для зазначення груп и звітів, доступних користувачеві для друку. При натисканні на кнопки в розділі **Програми** можна створювати для користувача логіку обробки таких дій як створення, редагування, копіювання, видалення записів з реєстру. Програма групової операції буде відпрацьована для кожного зазначеного запису в реєстрі. Обробка призначеної для користувача події запускає написану програму для обраного запису реєстру після натискання комбінації клавіш **Alt+U** .

23.3.3. Редактор XML-файлу

Для створення XML-реєстрів можна використовувати редактор XML-файлів. Редактор відкривається з вікна створення / редагування користувальницького реєстру (вкладка **Реєстри** при виборі пунктів **Створити / Відкрити**) по кнопці в полі **Файл** . Редактор відображає ієрархічну структуру XML-файлу з візуальним виділенням різних елементів. Може створювати,

змінювати, видаляти елементи, атрибути, коментарі. Відображати текст XML-файлу з підсвічуванням синтаксису. Має можливість редагувати текст XML-файлу. Відображає допоміжну (відладочну) інформацію: підказки, помилки і попередження. Редактор може створювати новий XML файл зі структурою реєстрів **Комплексу**. Є вікна редагування елементів реєстрів **Комплексу**.

23.4. Користувальницькі таблиці

Для створення нової користувальницької таблиці необхідно в меню модуля **Конструктор** на вкладці **Таблиці** вибрати пункт меню **Створити** (клавіша **Ins**). У вікні потрібно вказати назву таблиці, яка буде створена в базі даних (всі призначені для користувача таблиці мають префікс U _), найменування таблиці, яке буде відображатися в конструкторі. Так само потрібно вказати тип таблиці: монопольна або спільна. Якщо обраний тип **монопольна**, то таблиця створюється в базі підприємства і доступ до неї є тільки з цього підприємства. Якщо обраний тип **спільна**, то таблиця створюється в системній базі і може бути доступною з усіх підприємств

У вікні створення таблиці є дві вкладки. На першій вкладці можна створювати поля, в другій - індекси. При створенні нового поля в таблиці потрібно вказати ідентифікатор (з цим ім'ям буде створено поле в таблиці) . Найменування - опис поля, default - значення поля за замовчуванням (якщо при вставці нового запису в таблицю поле буде порожнім , то воно автоматично буде заповнено значенням за замовчуванням), клас - тип поля, аналогічні з типами в полях, довідниках користувача. Довжина поля - кількість символів в строковому поле. Обмеження - можливі обмеження, які накладаються на це поле. Параметр Identity вказує на те, що сервер баз даних буде виділяти для поля унікальний номер. Параметр Nullable вказує, чи може бути це поле порожнім (null).

Після створення полів таблиці можна перейти до створення індексів . На вкладці **Індекси** використовувати кнопку **Створити** (клавіша **Ins**). У вікні в лівому списку буде вибір полів таблиці. Необхідно встановити курсор на поле, по якому повинен будуватися індекс і натиснути кнопку **F5** . Поле з'явиться в правому списку , по цьому полю буде будуватися індекс . Індекс можна побудувати як по одному, так і по декількох полях таблиці. В правому списку можна вибрати за яким принципом буде будуватися індекс (наростання або спадання) встановленням параметра **По убуванню** біля імені індексу. Після вибору поля потрібно вказати тип індексу: Primary , Dupl , NoDupl (основний, дубльований, що не дубльований).

Перед виходом з вікна редагування таблиця автоматично буде створена і конвертована.

Що б додати новий користувальницький модуль в головне меню **Комплексу** необхідно зайти у вікно редагування меню (модуль **Користувачі і ролі**). У лівому списку встановити курсор на пункті **Зовнішні завдання** і натиснути кнопку **Створити** (клавіша **Ins**). У діалоговому вікні вказати найменування призначеного для користувача модуля, після чого натиснути кнопку **Текст** . У вікні написати одну з описаних нижче функцій:

RUNUSERRST (uint2) - запустити реєстр користувача, вхідний параметр це код реєстру;

RU NUSERMODULE (uint4) - запустити модуль користувача, вхідний параметр це код модуля.

Використання цих функцій не обмежується модулем **Користувачі і ролі** , вони можуть бути використані в будь-який калькуляції або програмі користувача в **Комплексі.**

23.5. Експорт та імпорт призначених для користувача вікон, таблиць та реєстрів

При необхідності на кожній з вкладок можна зробити експорт відповідних вікон, реєстрів і таблиць і імпорт.

Щоб зробити експорт вікон, таблиць або реєстрів необхідно вибрати потрібні записи в списку і в меню **Реєстр** вибрати пункт **Експорт.** У діалоговому вікні потрібно вибрати папку, в яку будуть вивантажені записи. Під час вивантаження таблиць крім опису так само вивантажуються і вміст таблиці. Дані знаходяться в окремому файлі, який називається, так само як і таблиця.

Щоб імпортувати вікна, таблиці або реєстри потрібно вибрати потрібні записи в списку і в меню **Реєстр** вибрати пункт **Імпорт.** Ці записи імпортуються в одному з режимів **Додати** або **Замінити**. У режимі **Додати** імпортовані записи додаються до існуючого списку і присвоюються нові номери, а в режимі **Замінити** проводиться заміна записів з однаковими номерами. У діалоговому вікні потрібно вибрати папку, в якій знаходяться файли для імпорту. Якщо в папці XML серверної частини **Комплексу** вже знаходиться XML - файл з найменуванням, як у файлі для імпорту то імпорт проведений не буде. Це актуально і для назв таблиць.

23.6. Резервне копіювання

В резервну копію **Комплексу** потрапляють призначені для користувача вікна, реєстри і таблиці. Для того, щоб XML-файли користувальницьких реєстрів потрапили в копію, вони повинні знаходитися в папці XML серверної частини **Комплексу.** В параметрах створення резервної копії потрібно включити параметр **Зберігати звіти і види реєстри користувача**. В копію додаються описи користувацьких таблиць з папки PIC \ USR серверної частини і дані цих таблиць.

23.7. Функції для роботи з користувальницькими вікнами і реєстрами

Список функцій **Комплексу** для роботи з користувальницькими вікнами і реєстрами:

RUNUSERRST (uint2) - запустити реєстр користувача, вхідний параметр це код реєстру;

RUNUSERMODULE (uint4) - запустити модуль користувача, вхідний параметр це код модуля;

REFRESHUSERRST () - оновити всі записи в призначеному для користувача реєстрі;

REFRESH1REC ( ) - оновити поточний запис в призначеному для користувача реєстрі;

int GETFIRSTMARKID () - повертає ID першого зазначеного запису в реєстрі. Якщо жодний запис не вибрано, то функція поверне 0;

int GETNEXTMARKID () - повертає ID наступного зазначеного запису в реєстрі. Якщо проведений повний прохід по зазначеним записам, то функція поверне 0.

Додано створення і використання користувальницьких глобальних змінних:

SETVALUEFORNAME (const pchar, pStayFD fd) - створити змінну з ім'ям *pchar* , в якій буде зберігатися значення *fd* , може бути переданий будь-який тип даних *;*

GETVALUEFORNAME (const pchar) - отримати значення з змінної з ім'ям *pchar* , яка була оголошена раніше *;*

DELETEVALUEFORNAME (const pchar) - видалити змінну з ім'ям *pchar* з пам'яті

Функції зняття позначок з реєстру:

void CLAERALLMARK () - зняти всі позначки, які встановлені в реєстрі;

uint4 MARKUNMARKID (uint4 Id) - відзначити / зняти позначку з записи реєстру.

Функція виклику калькулятора Комплексу:

function ShowCalculator (const AValue : Double)

Double - п про C TRL + Enter повертає результат обчислень, в іншому випадку - вхідний параметр.

Константи атрибутів файлів. Реалізовані функції:

- **FileSearch** - Здійснює пошук файлу в одній або більше папках;
- **FindInDir** - Повертає список файлів в каталозі по шляху, включаючи маску файлів, із зазначенням атрибутів файлів.
- **FileGetAttr** - Повертає атрибути файлу;
- **FileSetAttr** - Встановлює атрибути файлу.

Приклад запуску з передачею і отримання м параметрів:

```
CALL SETVALUEFORNAME ( " Name " , " 123 " )
CALL SETVALUEFORNAME ( " Name1 " , " OK " )
RUNUSERMODULE (3)
MESSAGE ( GETVALUEFORNAME ( " Name " ))
```

Результатом виконання цього прикладу буде оповіщення 123 після виходу з користувальницького модуля з кодом 3.