Затверджено

482.362. 6050102-02 35 09-1 ЛЗ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЧЕРНІВЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМ. Ю.ФЕДЬКОВИЧА

Інститут фізико-технічних та комп'ютерних наук
Кафедра комп'ютерних систем та мереж

ВАРІАНТ 9

482.362. 6050102-02 35 09-1

(Опис мови)

Сторінок 22

2015

АНОТАЦІЯ

Даний документ містить основні відомості про інструментальні засоби розробки програмного забезпечення, порівняння різних сучасних інтегрованих середовищ та компіляторів, показані критерії вибору засобів програмування. Увага, проведеного огляду, звернута на середовище програмування Embarcadero Delphi XE.

АННОТАЦИЯ

Данный документ содержит основные сведения об инструментальных средствах разработки программного обеспечения, сравнение различных современных интегрированных сред и компиляторов, показаны критерии выбора средств программирования. Внимание проведенного осмотра, обращено на среду программирования Embarcadero Delphi XE.

482.362. 6050102-02 09-1

ABSTRACT

This document contains basic information about the development tools software comparison of various modern integrated environments and compilers, browse the selection criteria of programming. Attention reviewed, paid to the programming environment Embarcadero Delphi XE.

ЗМІСТ

ВСТУП

Стрімкий темп розвитку інформаційних технологій на даний час дійсно вражає. Але в комп'ютерному світі залишається одна найважливіша область, зміни в якій протікають досить повільно. Програмування, кодування, складання вихідних текстів – ключовий елемент в створенні будь-якого додатку сьогодні здійснюється так само, як десятки років тому. На теперішній час найважливішим аспектом є швидкість і якість створення програм, а ці характеристики може забезпечити тільки середовище візуального проектування, здатне взяти на себе значні об'єми рутинної роботи по підготовці додатків.

Можливості Delphi повністю відповідають подібним вимогам і підходять для створення системи будь-якої складності. Система Delphi дозволяє писати як маленькі програми і утиліти для персонального використання, так і корпоративні системи, що працюють з базами даних на різних платформах.

1. ЗАГАЛЬНІ ВІДОМОСТІ

Embarcadero Delphi – це об'єктно-орієнтоване середовище візуального програмування (RAD – Rapid Application Development). Вона призначена для прискореної розробки високопродуктивних 32-бітних додатків, які можуть працювати в середовищі Windows, Linux або OS X. При цьому Delphi дозволяє звести до мінімуму об'єм програмного коду який вводиться вручну. В склад Delphi входять засоби, необхідні для розробки, тестування та установки додатків, включаючи велику за обсягом бібліотеку компонентів (VCL – Visual Components Library), засоби візуального проектування, шаблони додатків і форм.

В системі Delphi використовується спеціалізована версія мови програмування Паскаль, що постійно вдосконалюється; вона називається Delphi (в шостій і більш ранішніх варіантах системи Delphi вона називалась

Object Pascal - «Об’єктний Паскаль»). Ця версія включає набір розширень, орієнтованих тільки на застосування в рамках середовища Delphi і призначених для прискореного створювання додатків.

Перед створенням програмного продукту найважливішим питанням є вибір інструментальних засобів, за допомогою яких буде реалізована програма. Даний продукт створений за допомогою мови Object Pascal в середовищі програмування Embarcadero Delphi XE**.**

Середовище Embarcadero Delphi XE являє собою інтегровану оболонку розробнику, в яку входить набір спеціалізованих програм, які відповідають за різні етапи створення готового додатку. Основні вікна системи Embarcadero Delphi XE наступні: інспектор об’єктів, провідник, проектувальник форм, вікно редактора. Вихідний текст програми готується в середовищі Embarcadero Delphi XE за допомогою вбудованого редактора вихідних текстів. Цей редактор спеціалізований. Він відрізняється гнучкими можливостями кольорового виділення різних елементів тексту програми (ключових слів, назв, операцій, чисел і рядків) і надає можливість швидкого вводу конструкцій, які часто зустрічаються. Найважливішою характеристикою програми, що розробляється є зручність її користувацького інтерфейсу, наявність і доступність необхідних елементів управління. В системі присутній проектувальник форм, за допомогою якого вікна майбутньої програми підготовляється у вигляді форм. Проектувальник дозволяє підібрати оптимальні розміри вікон, розмістити і настроїти загально можливі елементи управління і меню, добавити готові зображення, вказати заголовки, підказки, підписи та багато іншого.

2. ЕЛЕМЕНТИМОВИ ТА СПОСОБИ СТРУКТУРИЗАЦІЇ ПРОГРАМИ

Елементами мови є набори компонентів, які дозволяють створювати додатки за найрізноманітнішими тематиками. Компоненти володіють наборами властивостей, що характеризують їх особливості. Крім

властивостей, компоненти містять методи – програмний код, який обробляє значення властивостей та події – повідомлення, які компонент приймає від додатку.

Всі додатки в Embarcadero Delphi XE будуються по наступному принципу: в їхній головній частині з розширенням. DPR зберігається тільки виклик декількох команд, які відкривають головне вікно, а також виконують завершальні дії. Решта всього програмного коду міститься в файлах, що зберігають опис додаткових модулів, які підключаються. Кожен модуль має строго задану структуру, яка зазвичай автоматично генерується системою Embarcadero Delphi XE при його створенні. Модуль складається з чотирьох частин: інтерфейсної частини, частини реалізації (обов'язкова), частини ініціалізації і частини завершення (необов'язкова) [1].

Спочатку вказують заголовок модуля – ключове слово Unit, за ним довільну назву модуля (вона повинна співпадати з іменем файлу, в якому модуль зберігається) і кладуть крапку з комою: **Unit** Testunit; Інтерфейсна частина описує інформацію, яка доступна з інших частин програми, з інших модулів і головної частини. Частина реалізації описує інформацію, яка недоступна з інших модулів. Подібне розділення модуля на частини дозволяє створювати і розповсюджувати модулі у відкомпільованому вигляді (розширення .DCU), додаючи до них тільки опис інтерфейсної частини. При цьому внести зміни в такий модуль неможливо, вихідний код, який реалізує описані в інтерфейсній частині можливості, недоступний. Такий підхід дозволяє повторно використовувати раніше написані для інших програм і вже відкоректовані модулі та розмежовує доступ до модуля декількох програмістів, а також дозволяє розбивати програму на набір логічно незалежних модулів. Інтерфейсна частина завжди йде першою і починається з ключового слова interface, а частина реалізації з – implementation.

Частини ініціалізації і завершення необов'язкові. Вказані в них дії виконуються, відповідно, на самому початку та в самому кінці роботи

програми і тільки один раз. Частина ініціалізації починається з ключового слова **initialization**, частина завершення – з ключового слова **finalization**. В кінці модуля завжди ставиться слово **end** і крапка.

Базовими елементами мови являються: коментарі, змінні, константи, оператори, типи даних тощо.

2.1 Змінні

Змінна – це область пам’яті, в якій знаходяться дані, якими оперує програма. Коли програма маніпулює з даними вона, фактично, оперує з вмістом комірок пам’яті.

В якості імені змінної можна використати послідовність з букв латинського алфавіту, цифр і деяких спеціальних символів. Першим символом змінної повинна бути буква. Пробіл у назві змінної використовувати не можна.

В загальному вигляді інструкція опису змінної має наступний вигляд

Ім’я: тип;

де:

ім’я – ім’я змінної;

тип – тип даних, для збереження яких призначена змінна.

Наприклад:

a: real;

b: boolean;

i: integer;

2.2 Коментарі

Коментарі являють собою пояснювальний текст, який можна записувати у будь – якому місці програми, де дозволений пробіл. Текст коментарів обмежується символами (* і *) або аналогічними { і } і може містити в собі будь – які символи мови, в тому числі і кирилиця. Коментарій,

обмежений вказаними символами, може займати декілька рядків. Однорядковий коментар на початку рядка містить подвійний слеш //.

Приклад коментарів:

```
(*однорядковий коментар*)
// другий однорядковий коментар
(*початок багаторядкового коментарю
  кінець багаторядкового  коментарю*)
```

Коментар ігнорується компілятором і не впливає на виконання програми. За допомогою коментарів можна виключати будь – які оператори програми в процесі її налагодження, наприклад, наступним чином:

```
Sum:=0;
For:= 1 to 100 do begin
  Read (x);
// if x<0 then x:=0;
sum:= sum+x;
end;
```

Тут умовний оператор в тілі циклу оформлений як коментар і він не буде виконуватися.

2.3 Типи даних

Сувора типізація даних – одна з найважливіших властивостей мови Object Pascal. Це означає, що всі реальні змінні, які передаються в якості параметрів в функцію чи процедуру, повинні абсолютно точно відповідати типу формальних параметрів в оголошенні цієї функції чи процедури.

Дійсні типи даних: single, real, double та exendet.

Вибір одного з даних типів для представлення змінних програми визначається необхідною точністю (кількістю розрядів мантиси) їх представлення і діапазоном представлення значень їх порядку.

Цілочисельні типи: shortint, byte, word, integer, longint.

Вибір одного із типів даних визначається діапазоном використовуваних значень змінних. Цілі зберігаються в двійковій системі числення у вигляді послідовності 1 і 0.

В Delphi існує три символьних типи:

– AnsiChar – стандартний однобайтовий символ.

– WideChar – двобайтовий символ Unicode.

– Char – однобайтовий символ.

Рядкові типи:

– AnsiString – рядковий тип який складається з символів AnsiChar і теоретично не має обмежень по довжині. Цей тип сумісний з рядками, що закінчуються нульовим символом.

– ShortString – максимальна довжина рядка 255 символів.

– WideString – даний рядок складається з символів WideChar.

– PChar – являє собою вказівник на рядок з завершуючим нульовим символом, що складається з символів типу Char.

– PAnsiChar – вказівник на рядок AnsiChar з завершальним нульовим символом.

– PWideChar – вказівник на рядок WideChar з завершальним нульовим символом.

Логічний тип даних Boolean визначає одне з двох значень: true (істинно) або false (хибно). Вони впорядковані: у false порядковий номер 0, у true порядковий номер 1.

2.4 Оператори

Оператори – це ті символи в тексті програми, за допомогою яких виконуються певні дії з даними різних типів. Найпростішим прикладом можуть бути оператори додавання, віднімання, множення і ділення арифметичних типів даних; іншим прикладом може бути оператор для доступу до певного елементу масиву.

Оператор присвоєння:

Number := 5;

Оператор порівняння:

if x = y

Оператор “не дорівнює” виглядає так:

if x <> y then DoSomething

В якості логічних операторів “і” та “або” в мові Object Pascal використовуються ключові слова and і or. В основному ці оператори використовуються як елементи оператора if або циклу. Наприклад:

if (Condition1) and (Condition2) then DoSomething;

while (Condition1) or (Condition2) do DoSomething;

Побітові оператори – це оператори які дозволяють працювати з окремими бітами заданої змінної. Найчастіше побітові оператори використовуються для зсуву бітів вправо чи вліво, їх інверсії, а також побітових операцій “і”, “або” та “виключаюче або” між двома числами. Оператори зсуву вліво і вправо в Object Pascal мають вигляд shl та shr відповідно. Решта - not and or і xor.

3. ВБУДОВАНІ ЕЛЕМЕНТИ

Основними вбудованими елементами, що були використані при створенні програмного продукту є:

1. Компонент TSpeedButton.

TObject–>TPersistent–>TComponent–>TControl–>TGraphicControl–> TSpeedButton

Модуль BUTTONS. Сторінка Палітри компонентів Additional.

Ця кнопка із зображенням може мати як залежну, так і незалежну фіксацію. Вона зручна для застосування у складі панелей інструментів. Поведінка цих кнопок багато в чому визначається властивістю: property Grouplndex: Integer.

Якщо Grouplndex рівний нулю, у кнопки взагалі немає фіксації в натиснутому стані і вона не залежить від решти кнопок. Кнопки в групі (тобто з однаковим ненульовим значенням Grouplndex) мають залежну фіксацію. Вона також залежить від властивості property AllowAllUp: Boolean, яке описує поведінку кнопок в групі, а саме: чи можуть всі кнопки одночасно бути віджаті. Якщо AllowAllUp рівне False (за умовчанням), натиснуту кнопку в групі можна відпустити, лише натиснувши іншу. Якщо AllowAllUp рівне True, кнопку можна відпустити повторним натисненням.

Якщо ви хочете фіксувати одну кнопку TSpeedButton, їй потрібно привласнити унікальний груповий індекс, а AllowAllUp встановити в True.

Оскільки в групі не можуть одночасно знаходитися кнопки з різним значенням цієї властивості, при натисненні кнопки і зміні Grouplndex властивість AllowAllUp "розсилається" (привласнюється) решті кнопок з тим же значенням Grouplndex. У групі не може бути натиснуто більш за одну кнопку. Визначає, чи натиснута кнопка, властивість: property Down: Boolean.

Ця властивість може змінюватися як системою, так і програмістом. Наприклад, якщо при запуску програми необхідно, щоб одна з кнопок вже була натиснутою, її властивість Down встановлюють в True.

Текст кнопки визначає властивість Caption. Компонент має ті ж правила і властивості малювання картинки, що і TBitBtn. Вони описуються властивостями Glyph, NumGlyphs, Layout, Margin і Spacing.

Для імітації клацання передбачений метод Click. Подвійне клацання для TSpeedButton можливе тільки на натиснутій кнопці – інакше він інтерпретується як звичайний. Описується властивістю: property OnDblClick.

Компонент TMemo. TObject->TPersistent->TComponent->TControl->TWinControl-> TCustomEdit->TCustomMemo->TMemo.

Модуль STDCTRLS. Сторінка Палітри компонентів Standard.

Компонент є багаторядковий редактор тексту. Вміст редактора представлений як об'єкт, що містить текст у вигляді набору рядків: property Lines: TStrings.

Текст в редакторові може вирівнюватися по лівому, правому краям і по центру: property Alignment: TAlignment; TAlignment = (taLeftJustify, taRightJustify, taCenter).

При наборі тексту користувач може ввести різні символи, що управляють, зокрема, клавішами <Enter> і <TAB>. Ці символи можуть бути оброблені редактором, а можуть бути відразу передані формі. У випадку, якщо властивості:

– property WantReturns: Boolean;

– property WantTabs: Boolean;

обернені в True, символи передаються редакторові. Звернемо увагу на те, що якщо встановлене WantTabs, то за допомогою клавіатури передати фокус такому редакторові можна, а після цього віддати іншому компоненту – не можна. Якщо властивості рівні False, символи передаються формі. В цьому випадку для введення цих символів в редактора можна скористатися комбінаціями <Ctrl>+<Enter> і <Ctrl>+<Tab> відповідно.

Дві властивості відповідають за організацію прокрутки тексту у вікні редактора: property Wordwrap: Boolean – відповідає за поведінку редактора досягши правої межі під час набору тексту. Якщо властивість рівна True, то при цьому відбувається перехід на новий рядок. У разі False досягши правої межі відбувається горизонтальна прокрутка тексту і користувач може продовжувати набір; на новий рядок можна перейти, натиснувши <Enter>.

property ScrollBars: TScrollStyle;

TScrollStyle = (ssNone, ssHorizontal, ssVertical, ssBoth) ;

– встановлює наявність смуг прокрутки у вертикальному і горизонтальному напрямах. Якщо є горизонтальна смуга, то властивість Wordwrap втрачає сенс: замість перенесення відбувається прокрутка.

Для отримання повноцінного додатку – текстового редактора, в нього потрібно включити компонент TMemo і забезпечити засобами читання, запису і друку файлів, пошуку і заміни тексту і т.п.

Компонент TPaintBox. TObject->TPersistent–>TCornponent–>TControl–>TGraphicControl–> PaintBox.

Модуль EXTCTRLS. Сторінка Палітри компонентів System.

Найпростішою надбудовою над канвою служить компонент TPaintBox, призначений для малювання. Можна навіть розглядати його як канву, забезпечену атрибутами компоненту. Вона представлена властивістю: property Canvas: TCanvas.

Крім канви, компонент має свій колір (фону)

property Color: TColor;

і шрифт:

property Font: TFont;

Зобразити на канві компоненту можна, передбачивши обробник події:

property OnPaint: TNotifyEvent;

Компонент TStringGrid. TObject–>TPersistent–>TComponent–>TControl–>TWinControl-> TCustomControl–>TCustomGrid–>TDrawGrid->TStringGrid

Модуль GRIDS. Сторінка Палітри компонентів Additional.

Цей компонент реалізує можливості свого предка TDrawGrid стосовно таблиці рядків.

До складу компоненту доданий об'єкт класу TStrings, в якому зберігається вміст осередків. Він доступний як векторна властивість – двовимірний масив текстових рядків (розмірністю ColCount x RowCount), відповідних елементам таблиці: property Cells[ACol, ARow: Integer]: string.

Доступний і двовимірний масив об'єктів, відповідних осередкам: property Objects[ACol, ARow: Integer]: TObject.

Необхідно пам'ятати, що самих об'єктів в таблиці немає, і програміст

повинен створювати, а після закінчення використання таблиці видаляти об'єкти самостійно.

Можна працювати окремо як з рядками, так і із стовпцями таблиці. Властивості:

– property Cols[Index: Integer]: TStrings;

– property Rows[Index: Integer]: TStrings;

описують набори рядків (також у вигляді TStrings), що містять текст і об'єкти стовпців і рядків таблиці.

При значенні True властивості DefaultDrawing для цього компоненту відбувається виведення рядка у відповідному осередку. Отже якщо окрім тексту нічого відображати не потрібно, то міняти значення DefaultDrawing і визначати обробник події OnDrawCell не потрібно.

Відзначимо, що перенесення рядків і стовпців таблиці (при встановлених опціях goColMoving або goRowMoving) здійснюється разом з їх текстовими рядками.

Компоненти TOpenDialog і TSaveDialog. TObject–>TPersistent–>TCoinponent–>TConimonDialog–> TOpenDialog–>TSaveDialog.

Модуль DIALOGS. Сторінка Палітри компонентів Dialogs.

Ці компоненти-діалоги призначені для вибору імені файлу, який буде надалі використаний для читання або запису. Розглянемо їх спільно, оскільки вони відрізняються тільки інтерпретацією деяких опцій.

Діалог може бути налаштований на представлення імен файлів якого-небудь одного типу або декількох типів. При цьому тільки ці типи відображаються в списку, і лише з них може зробити вибір користувач, за допомогою властивості: property Filter: string.

Формат рядка фільтру складається з двох частин. У першій задається короткий опис типу. У другій частині, яка відділяється символом '|', – маска пошуку потрібних файлів по розширенню. У список файлів потраплять тільки ті, які мають вказані розширення. Приклади завдання фільтру: 'Delphi

projectsl*.dpr' або 'All graphics files I *-bmp,*.ico,*.wmf '.

Таких пар рядків для різних типів у фільтрі може бути декілька, при цьому формати представлення фільтру в самій системі Windows і в Delphi мають одну відмінність. При виклику стандартних діалогів Windows ці пари рядків повинні розділятися нульовим байтом, після останнього рядка також повинен стояти нульовий байт. У даних компонентах всі рядки розділяються символом '|'.

Створювати рядки уручну потрібно лише в окремих випадках – для введення значення властивості на етапі проектування призначений спеціальний редактор. Є також простій спосіб скласти фільтр для графічних файлів – для цього призначена процедура GraphicFilter.

Як початковий фільтр при виклику діалогу буде вибрана та пара рядків, номер (індекс) якої співпадає із значенням властивості: property Pilterlndex: Integer.

Якщо властивості не привласнювалося значення, то за умовчанням вибирається перша пара.

Шлях до файлів, які спочатку будуть відображені в діалозі, задається властивістю: property InitialDir: string.

У випадку, якщо в полі введення імені файлу користувач при ручному наборі не визначив його розширення, це за нього намагається зробити діалог. Для цього властивості: property DefaultExt: TFileExt; TFileExt = string[3], привласнюється рядок (до трьох символів, без крапки), який і використовуватиметься як розширення за умовчанням. Наприклад, якщо користувач ввів в поле імені файлу 'mybitmap', а DefaultExt рівне 'bmp', то компонент поверне повне ім'я 'mybitmap. bmp'.

Поле введення FileName, де відображаються і редагуються імена файлів, може бути простим редагуючим елементом, а може бути і комбінованим списком. У цей список повинні потрапити імена файлів, які були раніше прочитані (записані). Стиль редагуючого елементу заданий

властивістю:

property FileEditStyle: TFileEditStyle;

TFileEditStyle = (fsEdit, fsComboBox).

а вміст списку імен, що раніше поверталися, – властивістю:

property HistoryList: TStrings.

У разі дії стилю fsEdit ця властивість не грає ролі. Якщо ж встановлене fsComboBox, у випадному списку з'являється вміст HistoryList.

Можна створити список для цієї властивості під час розробки програми. Але для того, щоб воно дійсно грало роль "передісторії", програміст повинен поповнювати список після успішного закінчення діалогу, що ілюструється наступним фрагментом коду:

```
procedure TFormI-FileOpenClick(Sender: TObject);
begin
with OpenDialogI do if Execute then begin
Memol.Lines.LoadFromFile(FileName) ;
HistoryList.Add(FileName);
end;
end;
```

У двох діалогів є великий набір опцій. Частина з них є загальною, частина – грає роль тільки для одного з діалогів:

property Options: TOpenOptions;

TOpenOption = (ofReadOnly, ofOverwritePrompt, ofHideReadOnly, ofNoChangeDir, ofShowHelp, ofNoValidate, ofAllowMuItiSelect, ofExtensionDifferent, ofPathMustExist, ofFileMustExist, ofCreatePrompt, ofShareAware, ofNoReadOnlyReturn, ofNoTestFileCreate) ;

TOpenOptions = set of TOpenOption;

Три опції відповідають за роботу з файлами із статусом "тільки для читання":

– ofReadOnly – робить прапорець "Read only" поміченим при появі;

– ofHideReadOnly – ховає цей прапорець в діалозі, що з'являється;

– ofNoReadOnlyRetum – забороняє вибір файлів "тільки для читання", сповіщаючи про необхідність вибрати інший файл при натисненні ОК.

Також три опції обмежують введення імен для нових (неіснуючих) файлів:

– ofPathMustExist – указує на те, що файл може знаходитися тільки в одному з існуючих каталогів. У разі введення неіснуючого шляху до файлу користувач сповіщається про помилку;

– ofFileMustExist – аналогічним чином указує на те, що може бути вибраний тільки один з існуючих файлів;

– ofCreatePrompt – опція встановлює реакцію на попередню ситуацію. Якщо вона встановлена, то замість повідомлення про помилку виводиться запит на створення нового файлу.

Інші опції:

– ofOverwritePrompt – запрошує підтвердження, якщо користувач вибрав для запису вже існуючий файл;

– ofNoChangeDir – забороняє зміну початкового каталога, з яким діалог буде проініціалізований. Якщо вона встановлена, діалог кожного разу з'являється з тим каталогом, який був встановлений при першому запуску;

– ofShowHelp – включає до складу діалогу кнопку Help;

– ofNo Validate – вимикає перевірку введеного імені файлу на наявність в нім неприпустимих символів;

– ofAllowMultiSelect – дозволяє вибирати декілька файлів одночасно;

– ofShareAware – відключає перевірку на можливість сумісного доступу до вибраного файлу. У разі відсутності цієї опції файл не можна вибрати, якщо він відкритий іншим додатком;

– ofNoTestFileCreate – ця опція застосовується тільки для файлів на тих вузлах локальної мережі, яким дозволено створення, але не модифікацію файлів. Якщо вона встановлена, діалог не перевіряє можливість запису на

вибраному пристрої.

Нарешті, одна опція – ofExtensionDifTereiit – є вихідний. Вона встановлюється після завершення діалогу в тому випадку, якщо розширення у введеного імені файлу відрізняється від того, яке визначене за умовчанням (у властивості DefaultExt).

Текст, що з'являється в заголовку діалогу, визначається властивістю: property Title: string.

Якщо воно не було задане, то в заголовках діалогів з'явиться "Open" для TOpenDialog і "SaveAs" для TSaveDialog.

У разі успішного завершення діалогу у властивості property Files: TStrings; містяться імена вибраних користувачем файлів. Якщо вибраний один файл, його ім'я можна отримати як Files.Strings[0], але є для цього і окрема властивість: property FileName: TFileName;

4. ЗАСОБИ НАЛАГОДЖЕННЯ ПРОГРАМ

Покрокове налагодження.

Одна з найпоширеніших задач налагодження – виконання програми крок за кроком, по одному рядку за раз для перевірки правильності виконання. При покроковому проходженні коду програма налагодження виводить вікно редагування з виконуваною програмою. Точка виконання, що показує наступний виконуваний рядок програми, представляється у вигляді зеленої стрілки, розташованої зліва від області початкового тексту у вікні редагування. Після успішної компіляції модуля на смузі налагоджувальної інформації кожний рядок коду модуля буде позначений маленьким, синім кружком. Якщо ж рядок не помічений, значить, оптимізатор компілятора виключив дану команду з списку виконуваних. Оскільки для таких рядків виконуваний код не згенеровано, ці рядки не будуть позначені точкою виконання.

Інтегроване середовище Delphi надає користувачу декілька команд покрокового налагодження доступних в меню Run:

- Step Over (F8) – покрокове виконання рядків програми, зчитуючи виклик функції чи процедури за один рядок, тобто вхід в функції та процедури не виконується.
- Trace Info (F7) – покрокове виконання програми з входом у викликані процедури та функції.
- Trace to Next Source Line (Shift+F7) – перехід до наступного виконуваного рядку.
- Run to Cursor (F4) – команда виконує програму до того виконуваного оператора, на якому розташований курсор у вікні редактору коду.
- Show Execution Point – команда поміщає курсор на оператор, який буде виконуватись наступним.

Вікно спостереження Watches.

В складних додатках виникає необхідність перегляду значень декількох змінних під час виконання програми. Таку можливість надає вікно спостереження Watches. Зробити його видимим можна командою View | Debug Windows | Watches. Для добавлення змінної в дане вікно необхідно підвести курсор до потрібної змінної та натиснути комбінацію клавіш Ctrl+F5. При покроковому виконанні програми у вікні спостереження будуть відображатися значення змінних, які було додані до нього.

ДЖЕРЕЛА ВИКОРИСТАНІ ПРИ РОЗРОБЦІ

1) Бобровский С. Developer Sturdio 2006. Учебный курс. – СПб.: Питер, 2004. – 735с.
2) Баас Р., Фервай М., Гюнтер Х.. Delphi 5: для пользователя. Пер. с нем. – К.: Издательская группа BHVб, 2000 – 496 с.
3) Керман, Митчелл, К. Программирование и отладка в Delphi. Учебный курс.: Пер. с англ. – М.: Вильямс, 2002. – 672 с.: ил.
4) Архангельский А.Я. Программирование в Delphi 5 – 2-е изд., переработ. и дополн. – М.: ЗАО «Издательство БИНОМ», 2000. – 1072 с.
5) Гофман В.Э., Хомоненко А.Д. Delphi 6. – СПб.: БХВ – Санкт-Петербург, 2000. –800с.