**Практична робота №10.**

**Тема: Створення, компіляція й виконання найпростішого програмного проекту.**

**Мета роботи:** Практично закріпити поняття програми як автоматизованої системи; поняття даних, способи зберігання даних та їх роль у програмах; поняття об’єкта, властивостей і методів об’єкта; поняття події та обробника події, вдосконалити та перевірити знання та вміння використання основних складових середовища Delphi, засвоїти прийоми відкриття, редагування, компіляції та збереження найпростіших проектів.

**Обладнання:** ПК, інструкція практичної роботи, програма Delphi.

**Література:** Кащеєв Л.Б. Основи візуального програмування: Навч. посібник – Х.:Веста, 2011. – 192 с.

**Теоретичні відомості.**

**Мова програмування Delphi. Інтерфейс мови.**

**Delphi – сучасна технологія візуального програмування**

Існують мови програмування високого рівня, низького. Низький рівень програмування – програмування в кодах процесора. Нагадаємо, команда процесора містить 32 ( або 64) біти, тобто 32-хрозрядна послідовність нулів і одиниць. Частину коду команди займає код операції а інша частина – фізичні адреси комірок ОП та пристроїв введення-виведення, задіяних в виконанні команди. Написання програми в такій формі – дуже кропіткий процес, він існував тільки для ЕОМ І покоління. Далі вже виникли алгоритмічні мови, в яких кожна команда – це текст, що містить певні слова (найчастіше англ. мови), що визначають команду та формули, близькі по запису до математичних.

Програма в машинних кодах при завантаженні в ОП зразу виконується процесором. Як правило, програма в машинних кодах зберігається в файлах \*.ехе або \*.com, тому такі файли і називаються виконавчими програмами. Програма на алгоритмічній мові зберігається в текстових файлах, але їх розширення співзвучне мові програмування: \*.bas, \*.pas, \*.si+, тощо.

Перетворення програми з алгоритмічної мови на машинну мову виконують певні програми, що називаються трансляторами. Транслятори діляться на інтерпретатори і компілятори. Розглянемо принцип роботи кожного з них.

Інтерпретатори (Basic) перекладають кожну команду з алгоритмічної мови на машинну і тут же їх виконують. Тобто процеси трансляції і виконання не розділяються. Такі програми можна завантажувати і виконувати тільки в середовищі інтерпретатора. Тобто, файл \*.ехе або \*.com отримати не можливо.

Компілятори спочатку перекладають всю програму з алгоритмічної мови на машинну, утворюючи так званий об’єктний програмний модуль. Об’єктний модуль може зберігатися в ОП або на дисках як файли \*.ехе або \*.com. Тобто, процес трансляції і виконання програми розділені.

**Складові програми: дані, логіка, інтерфейс**.

Виконання будь-якої програми відбувається, як правило, за таким планом: відбувається звертання до користувача для введення початкових даних, далі за певним алгоритмів відбувається обробка отриманих даних, і завершує цей процес виконання програми друкування результатів.

Починаючи створення програми, треба чітко визначити, які дані будуть визначатися користувачем в ході виконання програми, які параметри будуть визначатися як константи в самій програмі, які вводитись з баз даних, з файлів на дисках.

Якщо певні дані будуть визначатись користувачем в ході виконання програми, то треба продумати систему вікон, через який буде здійснюватись зв’язок (інтерфейс) програми і користувача. Може введення даних бажано здійснювати в окремому вікні, а виведення результатів в другому вікні, може введення даних і виведення результатів здійснювати в одному вікні.

Процес обробки отриманих даних в програмі може здійснюватись по різному. Як і людина може розв’язати одну задачу різними методами, так і в програмі може бути використаний той чи інший алгоритм (логіка програми).

**Об’єкти і їх властивості**

Що таке об’єкт? Постановка серйозної задачі практично ніколи не містить опису змінних і масивів, які треба використовувати. Навпаки, більшість реальних задач формулюються як опис поведінки об’єктів, наприклад при комп’ютерному моделюванні складних систем.

В кінці 80-х років XX століття програмування опинилося в достатньо серйозній кризі. Головна проблема полягала у тому, що складність програм стала перевищувати можливості людського інтелекту. Одній людині стало не під силу охопити розумом всі аспекти складної програми.

Виходом з кризової ситуації став підхід, заснований на понятті об’єкту. Нова ідея полягала в тому, щоб розбити складну систему на частини – об’єкти і спробувати зрозуміти і удосконалювати кожну частину окремо, незалежно від інших. При цьому розробку кожного об’єкту можна доручити окремій людині. Такий підхід називають об’єктно-орієнтованим програмуванням.

**Об’єкт** – це щось, що має властивості і методи (функції).

**Властивості** – це якісь характеристики об’єкту, які ми можемо спостерігати і, можливо, змінювати. Васю Пуркіна можна розглядати як об’єкт. Його властивості – це зріст, вага, колір очей, характер, зачіска і т.п. Властивості об’єкту-автомобіля – це його марка, колір, об’єм двигуна, рік випуску.

**Методи** – це команди, які об’єкт розуміє і може виконувати, причому в документації на об’єкт сказано, як саме він реагує на ці команди. Наприклад, автомобіль реагує на натиснення педалей, поворот руля,  перемикання швидкостей.

Ми нічого не знаємо про те, що містить об’єкт всередині. Для нас це, як говорять кібернетики, «чорний ящик». Але найголовніше – для того, щоб працювати з об’єктом, нам і не потрібно знати його внутрішній устрій. Достатньо, що ми можемо визначити і змінити його властивості, а також застосовувати доступні методи управління.

Властивості і методи є інтерфейсом об’єкту, тобто спосіб його спілкування із зовнішнім світом.

Об’єднання властивостей і методів в єдиному понятті об’єкту називають **інкапсуляцією.**

Програми в середовищі Delphi найчастіше будуються на принципах об’єктно-орієнтованого програмування. Наприклад, більшість програм, що працюють в системі Windows, мають вікно. У Delphi вікно (воно називається формою) – це об’єкт, у якого є властивості (заголовок, колір, розміри і т.п.) і методи (зокрема обробники повідомлень).

На формі розташовані елементи управління – кнопки, перемикачі, поля введення і ін. Вони також є об’єктами з своїми властивостями і методами.

Ми поступово знайомитимемося з об’єктами Delphi на прикладах, заглиблюючись тільки в ті деталі, які потрібні в даній задачі.

**Процес створення програми**

**Delphi** – це грецьке місто, де жив дельфійській оракул. І цим ім’ям був названий новий програмний продукт з феноменальними характеристиками. Середовище Delphi фірми Borland  – це одне з найпопулярніших середовищ швидкої розробки додатків (RAD = *RapidApplicationsDevelopment*). Мова, що реалізована в Delphi – це сучасний варіант мови Паскаль (спочатку його називали ObjectPascal – об’єктний Паскаль, а зараз – просто мова Delphi).

Розробка програми в середовищі RAD складається з наступних етапів:

1. **Створення форми** (вікна майбутньої програми); при цьому мінімальний код будується автоматично і відразу виходить працездатна програма;

2. **Розстановка елементів інтерфейсу** на формі (поля введення, кнопки, текстові повідомлення, списки) за допомогою миші;

3. **Створення обробників подій** подвійним клацанням миші, мінімальний код також будується автоматично;

4.  **Написання коду обробників**, який реалізує потрібний алгоритми обробки даних.

Середовища RAD дозволили істотно скоротити час розробки програм. Проте потрібно пам’ятати, що будь-який інструмент – це тільки інструмент, який можна використовувати грамотно і безграмотно. Використовування середовища RAD не гарантує, що у вас автоматично вийде хороша програма з хорошим призначеним для користувача інтерфейсом.

Інтерфейс програми складають компоненти, які розробник обирає з Палітри компонентів і розміщує на формі, тобто Компоненти являють свого роду будівельними блоками. При конструюванні інтерфейса додатка діє принцип WYSIWYG (What You See Is What You Get - що бачите, те й отримаєте), і розробник при створенні додатка бачить форму майже таку ж, як і при його виконанні.

Компоненти є структурними одиницями і діляться на візуальні (видимі) і невізуальні (системні). При цьому поняття "видимий" і "невидимий" відносяться тільки до етапу виконання, на етапі проектування видно всі компоненти програми.

До візуальних компонентів належать, наприклад, кнопки, списки або перемикачі, а також форма. Так як візуальні компоненти використовуються користувачем для управління додатком, то їх також називають керуючими компонентами або елементами управління. Саме візуальні компоненти утворюють інтерфейс програми.

До невізуальні компонентів належать компоненти, що виконують додаткові, але не менш важливі дії, наприклад, таймер Timer або набір даних Table (компонент Timer дозволяє відраховувати інтервали часу, а компонент Table являє собою записи таблиці бази даних).

При створенні інтерфейсу програми для кожного компонента виконуються наступні операції:

1. вибір компонента в палітрі компонентів і розміщення його на формі;

2. зміна властивостей компонента.

Розробник виконує ці операції у вікні Конструктора форми, використовуючи Палітру компонентів і Інспектор об'єктів. При цьому дії розробника схожі на роботу в середовищі графічного редактора, а сам процес створення інтерфейсу додатку більше нагадує конструювання або малювання, ніж традиційне програмування. У зв'язку з цим часто ра-боту зі створення інтерфейсу називають не програмуванням, а конструюванням.

Вибір компонента в палітрі виконується клацанням миші на потрібному компоненти, наприклад кнопці Button, в результаті чого його піктограма приймає втоплений вид. Якщо після цього натиснути на вільному місці форми, то на ній з'являється обраний компонент, а його піктограма в палітрі приймає звичайний вид. Піктограми компонентів відображають призначення компонентів, і при наявності невеликих практичних навичок вибір потрібного компонента відбувається досить швидко. Крім того, при наведенні на кожен компонент покажчика миші відображається текст з інформацією про його призначення.

Позначення типів об'єктів в Delphi, в тому числі і компонентів, починаються з букви т. Іноді типи використовуються замість назв для означення компонентів. Будемо використовувати для компонентів саме назви, а не типи, тобто Button, а не TButton, Label, а не TLabel. Після розміщення компонента на формі система Delphi автоматично вносить зміни в файли модуля і опису. В опис класу форми (файл модуля) для кожного нового компонента додається рядок формату

<Назва компонентів <Тип компонентів>

Назва нового компонента є значенням його властивості Name, а тип збігається з типом обраного з Палітри компонента. Наприклад, для кнопки Button ця строчка спочатку буде мати вигляд:

Buttonl: TButton;

У файлі опису для кнопки Button може бути записаний наступний код:

object Buttonl: TButton

Left = 88

Top = 120

Width =75

Height = 25

Caption = 'Buttonl'

TabOrder = 0 end

Для розміщення на формі кількох однакових компонентів зручно перед вибором компонента в палітрі компонентів натиснути і утримувати клавішу <Shift>. У цьому випадку після клацання миші в області форми і розміщення там обраного компонента його піктограма в палітрі залишається втопленою, і кожний наступний клацання на формі призводить до появи на ній ще одного такого ж компонента. Для скасування вибору даного компонента досить вибрати інший компонент або клацнути мишею на зображенні стрілки в лівому кутку Палітри компонентів.  
Після розміщення компонента на формі можна за допомогою миші змінювати його положення і розміри. У разі декількох компонентів можна виконувати вирівнювання або переклад того чи іншого компонента на передній або задній план. При цьому дії розробника не відрізняються від дій в середовищі звичайного графічного редактора. Одночасне виділення на формі декількох компонентів можна виконати клацанням миші, утримуючи клавішу <Shift>.  
За замовчуванням компоненти вирівнюються на формі по лініях сітки, що визначає прапорець Snap to grid (Вирівнювати по сітці), що входить в набір параметрів інтегрованого середовища розробки. У ряді випадків цей прапорець доводиться відключати, наприклад, при щільному розміщенні компонентів на формі. За замовчуванням крок сітки дорівнює восьми пікселам, а сітка при проектуванні відображається на поверхні форми. Необхідність вирівнювання по сітці, видимість сітки (перемикач Display grid (Відображати сітку)) і розмір кроку сітки по горизонталі і вертикалі установлюються на вкладці Preferences (Параметри) діалогового вікна Environment Options (Параметри середовища), що викликається однойменною командой меню Tools (Засоби).  
Зовнішній вигляд компонента визначається його властивостями, які стають доступними у вікні Інспектора об'єктів, коли компонент виділено на формі і навколо нього з'явилися маркери виділення (рис.). Доступ до властивостей самої форми здійснюється аналогічно, однак в обраному стані форма не виділяється маркерами. Для виділення (вибору) форми досить клацнути в будь-якому її місці, вільному від інших компонентів.

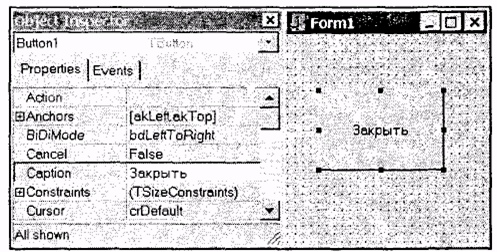


Рис.Доступ до властивостей компонента

У випадаючому списку, розташованому у верхній частині вікна Інспектора об'єктів, відображається назва компонента і його тип. Вибрати той чи інший компонент і, відповідно, отримати доступ до його властивостями можна також через список Інспектора об'єктів. Такий спосіб вибору зручний у випадках, коли компонент повністю закритий іншими об'єктами. У лівій частині вікна Інспектора об'єктів наводяться назви всіх властивостей компонента, які доступні на етапі розробки програми. Справа для кожного властивості варто його значення. Відзначимо, що крім цих властивостей компонент може мати і властивості, які доступні тільки під час виконання додатки. Властивості є атрибути, що визначають спосіб відображення і функціонування компонентів при виконанні програми. Кожен компонент має значення властивостей, що задаються за умовчанням. Після додавання компонента на форму його властивості можна змінювати в процесі проектування або в ході виконання програми. Управління властивостями в процесі проектування полягає у змінах значень властивостей компонентів безпосередньо у вікні Конструктора форми ( "малювання") або за допомогою Інспектора об'єктів. Розробник може змінити значення властивості компонента, ввівши або обравши потрібне значення. При цьому одночасно змінюється відповідний компонент, т. Е. Вже при проектуванні видно результати зроблених змін. Наприклад, при зміні назви кнопки (властивість caption) воно відразу ж відображається на її поверхні. Для підтвердження нового значення властивості досить натиснути клавішу <Enter> або перейти до іншого властивості або компоненту. Скасування змін проводиться клавішею <Esc>. Якщо властивості введено неприпустиме значення, то видається попередження, а зміна значення відкидається. Зміни властивостей автоматично враховуються в файлі опіса¬нія форми, що використовується компілятором при створенні форми, а при зміні властивості Name зміни вносяться і в опис класу форми. Кожен компонент для більшості своїх властивостей, наприклад, color (колір), Caption (назва) і visible (видимість), має значення за замовчуванням. Для звернення до компоненту в додатку призначена властивість Name, яке утворюється автоматично наступним чином: до назви компонента додається його номер в порядку приміщення на форму. Наприклад, перша кнопка Button отримує ім'я Buttonl, друга - Button2 і т. Д. першочергово від властивості Name отримує своє значення і властивість Caption. Зазвичай розробник дає компонентам більш інформативні імена, ніж імена за замовчуванням. При цьому доцільно включати в ім'я дані про тип компонента і його призначення в додатку. Так, кнопці типу TButton, призначеної для закриття вікна, може бути присвоєно ім'я btnciose або ButtonCiose. Кожен розробник самостійно встановлює зручні правила іменування компонентів. Для простоти будемо часто використовувати імена за замовчуванням, наприклад, Formi, Buttonl АБО Editl. Властивості, пов'язані з розмірами і положенням компонента (наприклад, Left і Тор), автоматично змінюють свої значення при переміщенні компонента мишею і зміні його розмірів.

Якщо на формі виділено кілька компонентів, то в інспектор об'єктів доступні властивості, загальні для всіх цих компонентів. При цьому зроблені в інспектор об'єктів зміни діють для всіх виділених компонентів. Для установки значень властивостей в Інспектора об'єктів використовуються підключені автоматично редактори властивостей:

Простий (текстовий) - значення властивості вводиться або редагується як звичайна рядок символів, яка інтерпретується як числовий або строковий тип Delphi; використовується для таких властивостей, як Caption, Left, Height І Hint. Перераховуються - значення властивості вибирається із списку. Список розкривається клацанням на стрілці, яка з'являється при установці курсора в області значення властивості. Можна не вибирати, а ввести за допомогою клавіатури потрібне значення, однак на практиці це зазвичай не робиться, т. К. Допускаються тільки пропоновані значення. Крім того, зростає трудомісткість і збільшується ймовірність помилки. Використовується для таких властивостей, як Formstyle, visible і ModalResult. Множинне значення властивості є комбінацією значень з пропонованого безлічі. У Інспектора об'єктів зліва від назви властивості множинного типу варто знак "+". Формування значення властивості виконується за допомогою додаткового списку, що розкривається подвійним клацанням на назві властивості. Цей список містить перелік усіх допустимих значень властивості, праворуч від кожного значення можна вказати True або False. Вибір True означає, що це значення включається в комбінацію значень, a False - ні. Іс-користується ДЛЯ таких ВЛАСТИВОСТЕЙ, як Borderlcons І Anchors. Об'єкту - властивість є об'єктом і, в свою чергу, містить інші властивості (підвластивості), кожне з яких можна редагувати окремо. Використовується для таких властивостей, як Font, items і Lines. В області значення властивості-об'єкта в дужках вказується тип об'єкту, наприклад, (т-Font) або (TSrings). Для властивості-об'єкта зліва від назви може стояти знак "+", в цьому випадку управління його властивостями виконується так само, як і для властивості множинного типу, т. Е. Через відкриваючій список. Цей список в лівій частині містить назви підвластивості, а в правій - значення, редаговані звичайним способом. В області значення може відображатися кнопка з трьома крапками. Це означає, що для даної властивості є спеціальний редактор, який викликається натисканням на цю кнопку. Так, для властивості Font відкривається стандартне вікно Windows для установки параметрів шрифту.

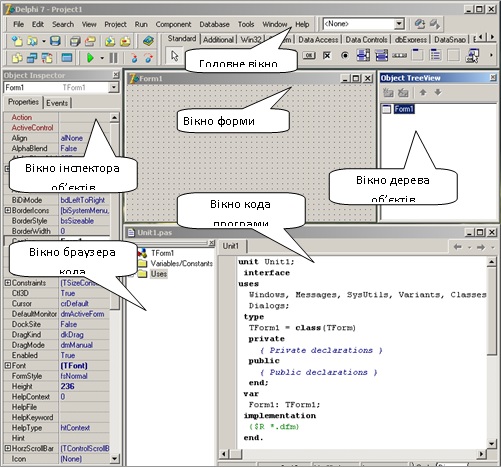
При виконанні програми значення властивостей компонентів (доступних у вікні Інспектора об'єктів) можна змінювати за допомогою операторів присвоювання, наприклад, в обробнику події створення форми. Так, зміна заголовка кнопки Buttoni можна виконати наступним чином:

Buttonl.Caption: = 'Закрити';

Такий спосіб вимагає, однак, більшого обсягу робіт, ніж в разі використання Інспектора об'єктів, крім того, подібні установки вступають в силу тільки під час виконання програми та на етапі розробки не видно, що в ряді випадків ускладнює управління візуальними компонентами. Проте, для наочності в багатьох прикладах значення окремих властивостей ми будемо встановлювати за допомогою операторів присвоювання, а не через Інспектор об'єктів. Відзначимо, що існують властивості часу виконання, недоступні через Інспектор об'єктів і з якими можна працювати тільки під час виконання додатка. До таких властивостей відносяться, наприклад, число записів RecordCount набору даних або поверхню малювання canvas візуального компонента.

**Середовище Delphi**

Інтегроване середовище розробника Delphi – це складний механізм, що забезпечує високоефективну роботу програміста. Візуально вона реалізується кількома одночасно розкритими вікнами. Завантажте Delphi, і ви отримаєте дещо схоже на малюнку, після нескладного переміщення вікон та зміни їх розмірів.

[](http://informatics.dp.ua/wp-content/uploads/2014/05/140.jpg)

**Головне меню** здійснює основні функції керування проектом створюваної програми. Не намагайтесь його розгорнути на весь екран, його розмір і вигляд при цьому майже не зміняться. Це пов’язано з його функціональним призначенням – воно містить всі інструменти для програміста і не займає місце інших вікон. Згортання головного вікна призводить до згортання всіх інших вікон середовища.

[](http://informatics.dp.ua/wp-content/uploads/2014/05/221.jpg)

**Інструментальні кнопки** відкривають швидкий доступ до найбільш важливих команд головного меню (менюView – Speedbar). По фунцціональним ознакам вони розділені на кілька груп, що займають окрему панель. Розглянемо їх вміст:

[3](http://informatics.dp.ua/wp-content/uploads/2014/05/317.jpg) – відкрити сховище об’єктів (newitems);

[4](http://informatics.dp.ua/wp-content/uploads/2014/05/417.jpg) – відкрити файл (openfile);

[5](http://informatics.dp.ua/wp-content/uploads/2014/05/516.jpg) – зберегти файл (savefile);

[6](http://informatics.dp.ua/wp-content/uploads/2014/05/611.jpg) – зберегти всі файли проекту;

[7](http://informatics.dp.ua/wp-content/uploads/2014/05/712.jpg) – відкрити проект;

[8](http://informatics.dp.ua/wp-content/uploads/2014/05/810.jpg) – додати файл до проекту (addfiletoproject);

[9](http://informatics.dp.ua/wp-content/uploads/2014/05/98.jpg) – видалити файл з проекту (removefilefromproject);

[10](http://informatics.dp.ua/wp-content/uploads/2014/05/106.jpg) – обрати модуль зі списку модулів, пов’язаних з поточним проектом (selectunitfromlist);

[11](http://informatics.dp.ua/wp-content/uploads/2014/05/1113.jpg) – обрати форму зі списку форм, пов’язаних з поточним проектом (selectformfromlist);

[12](http://informatics.dp.ua/wp-content/uploads/2014/05/1210.jpg) – перемкнути активність між вікном форми і вікном кода програми (toggleform/unit);

[13](http://informatics.dp.ua/wp-content/uploads/2014/05/1310.jpg)– створити нову форму і додати її до проекту (newform);

[14](http://informatics.dp.ua/wp-content/uploads/2014/05/143.jpg) – запустить компіляцію та виконання програми (run);

[15](http://informatics.dp.ua/wp-content/uploads/2014/05/153.jpg)– призупинити виконання програми (pause);

[16](http://informatics.dp.ua/wp-content/uploads/2014/05/163.jpg) – здійснити покрокове виконання програми з викликом підпрограм (stepinto);

[17](http://informatics.dp.ua/wp-content/uploads/2014/05/172.jpg) – здійснити покрокове виконання програми без виклику підпрограм (stepover).

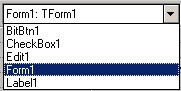
**Палітра компонентів** – це головне багатство Delphi (менюView – ComponentPalette). Вона розташована в правій частині Головного вікна і містить кілька вкладок, що забезпечують швидкий пошук потрібного компонента. Під компонентом розуміють певний функціональний вузол, який програміст може розмістити в вікні форми, який має певні властивості. Як правило за допомогою цих компонентів користувач програми вводе дані в програму або отримує результати її роботи. Це може бути вікно введення, перемикачі, прапорці, списки і т.п. Палітру можна редагувати за допомогою ДВ EnvironmentsOptions (КМ палітри – Configure).

[18](http://informatics.dp.ua/wp-content/uploads/2014/05/183.jpg)

**Вікно форми** – в майбутньому перетвориться в вікно програми. На початку це вікно пусте і містить тільки стандарті для Windows інтерфейсні елементи в рядку заголовка. Вся область вікна заповнена координатною сіткою. Значну частину часу програміст ніби з конструктором Logo бере з палітри певний компонент і викладає його на форму. В наповненні форми і полягає основна родзинка візуального програмування. Програміст в будь-який час контролює вид вікна створюваної програми і може вносити в нього певні зміни.

**Вікно інспектора об’єктів**. Будь-який розміщений на формі об’єкт має певні характеристики: розташування, розмір, колір, тощо. Частину цих параметрів програміст може визначити маніпулюючи мишкою: перетягти, змінити розмір. Але всі характеристики об’єктів визначаються в вікні інспектора об’єктів. Це вікно має 2 вкладки: Properties (Властивості) і Events (Події). Вкладка Properties використовується для визначення певних властивостей компонента, а вкладка Events дозволяє визначити реакцію компонента на ту чи іншу подію. Сукупність властивостей відображає видиму сторону компонента: відстань відносно лівого верхнього кута вікна, розмір,  колір, тощо, сукупність подій – його поведінку: чи буде реагувати на клацання мишею, як буде з’являтись на екрані, чи буде зникати після використання і т.п.

Кожна вкладка вікна постає перед програмістом в вигляді таблиці з 2-х стовпчиків: лівий – назви властивостей або подій, правий – значення властивості або ім’я підпрограми, що її обробляє. Рядки таблиці обираються кліком миші і відображають прості (одне значення) або складні властивості (сукупність значень). Виділивши складну властивість, маємо біля неї позначку-кнопку –[19](http://informatics.dp.ua/wp-content/uploads/2014/05/193.jpg) , після натискання на неї з’явиться діалогове вікно, в якому визначаємо сукупність параметрів властивості. В верхній частині вікна маємо відкриваючий список всіх компонентів,

[](http://informatics.dp.ua/wp-content/uploads/2014/05/203.jpg)розміщених в формі. Оскільки сама форма також є компонентом, тої ім’я також є в цьому списку.

**Вікно кода програми** призначене для створення і редагування тексту програми. Мова програмування Delphi є надбудовою над мовою Pascal. Зразу після відкриття нового проекту маємо рядки, що описують присутністю в програмі форму. Як вже зазначено, вікно кода програми описує поведінку програми, а вікно форми – її зовнішнє відображення. Обидва вікна тісно пов’язані між собою, і будь-які зміни в формі вносять зміни в коді програми між рядками **unit** і **implementation**.

**Вікно дерева об’єктів** – призначене для візуального відображення зв’язків між окремими компонентами, розміщеними на активній формі або в активному модулі. Клік на будь-який компонент в цьому вікна активізує відповідний компонент в формі і відображає властивості цього компонента в вікні інспектора об’єктів.

**Хід роботи.**

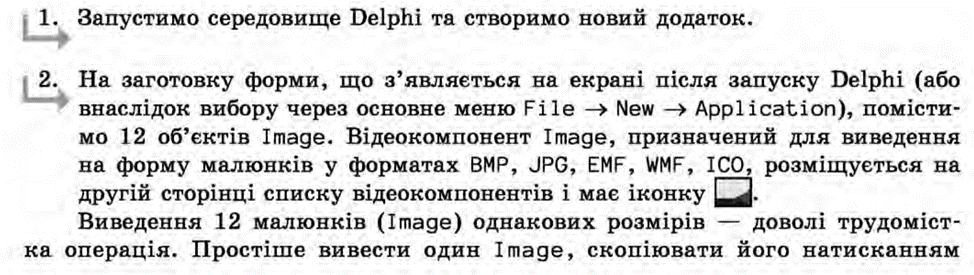
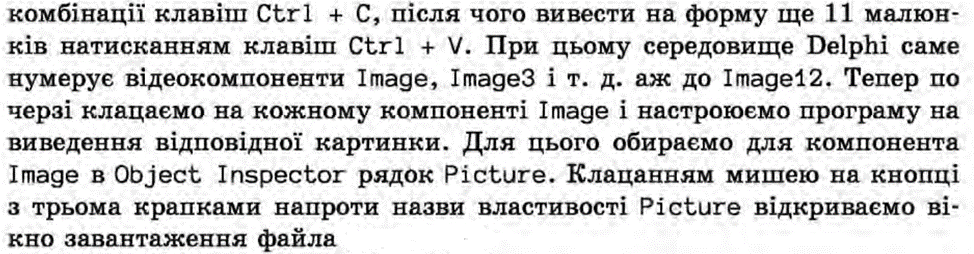




Рис. Програма, що виводить англійські назви предметів на картинках



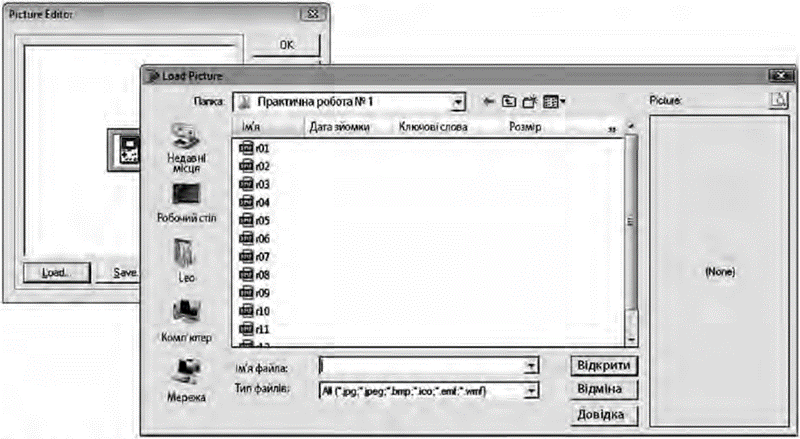
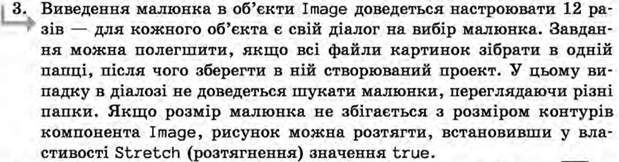
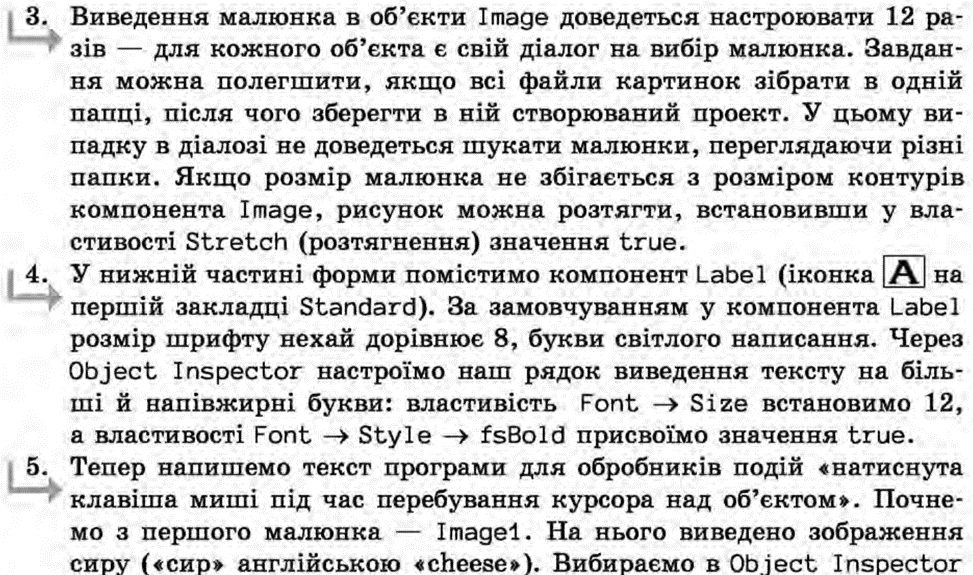
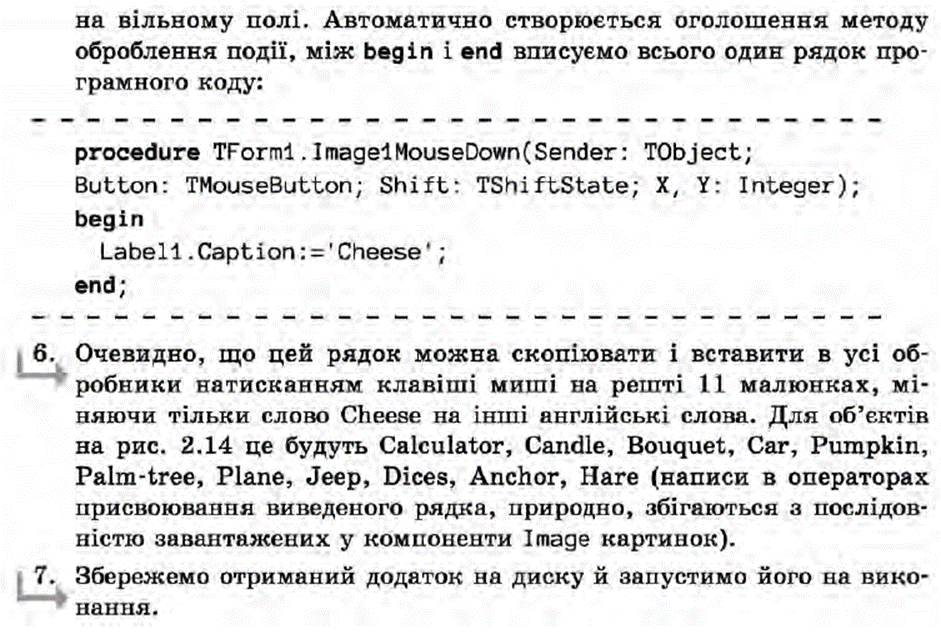


Рис. Завантаження малюнка з файла в компонент Image клацанням на кнопці Load







8. Оформіть звіт та зробіть висновки.

9. Контрольні питання.

* Низький рівень програмування – це …
* Транслятори діляться на …
* Властивості – це ….
* Інтерпретатори перекладають кожну команду з ….
* Методи – це ….
* Об'єкт – це …
* Програма в машинних кодах зберігається в файлах…
* Що таке компілятори.
* Що таке транслятори.
* Назвіть елементи інтерфейсу Delphi.