Міністерство освіти і науки України

Рівненський професійний ліцей

ДИПЛОМНА РОБОТА

Основні концепції візуального програмування в середовищі Delphi. Об'єднання елементів управління.

Виконала:
учениця групи № 9
За фахом: оператор ЕОМ та ОМ
Колодій Н.О.

Керівник:
Раковець М.Г.

2013 рік

ЗМІСТ

ВСТУП

На сучасному етапі розвитку комп'ютерних систем в світі розвинулася і поширилася велика кількість мов програмування, які призначені для різних областей людської діяльності. Визначальним моментом при виборі мови програмування додатків в більшості випадків являється досвід у використанні тієї чи іншої мови.

Delphi - є середовищем розробки програм, яке використовує багато передових ідей і концепцій, закладених в графічному інтерфейсі Windows. Як відомо, середовище розробки великою мірою визначає ефективність роботи програміста. У середовищі програмування є всі необхідні інструменти для того, щоб створювати повноцінні програми. Писати, компілювати і тестувати програму - все це можна робити, не виходячи з Delphi.

Простота, швидкість і ефективність Delphi пояснюють її популярність. Delphi має одне із найбільш швидких компіляторів, який породжує, тим щонайменше, дуже й дуже непоганий об'єктний код. Є й інші гідності: простота вивченняObjectPascal; які полегшують життя нововведення - на кшталт

вастивостей (>properties); програми, написані Delphi, непотрібен постачати додатковими бібліотеками (на відміну зв'язкиC++/MFC). У насправді,VCL надає зручний, легко розширюваний обєктно-орієнтованний інтерфейс до Windows, що ні жодному разі корисно програмісту опускатися в глибини Windows АПІ. Творцям оригінальних компонентів це робити частенько, на відміну "просто програмістів". Як було вказано вище, модель програмування в Delphi -компонентна, що дозволяє користуватися компонентами, написаними іншими розробниками, не маючи їх вихідного коду і вже поготів не вивчаючи його. У Інтернеті величезна кількість компонентів, значна частина поширюється безплатно. Застосування компонентної моделі призводить до того, що досить багато речей поведінці об'єктів програмувати непотрібно взагалі, і що, потім за іншими середовищах пішли б тижня, можна за годинник і

навіть хвилини. Звісно,ObjectPascal накладає певні обмеження, але тим речей, котрим її писали, Delphi підходить практично оптимально. З запропонованих (але, зрозуміло, нестандартних) "поліпшень", внесенихBorland вObjectPascal, хотілося б виділити властивості (>properties) і перезагрузка процедур і державних функцій (>overloading). Певні незручності під час роботи знизкоурівневими функціями АПІ може викликати те, що стандартним мовою для АПІ усе є З, і саме ньому пишуться дедалі нові SoftwareDevelopmentKit (SDK) і заголовкові файли до них.

Згідно практичного завдання мені необхідно використати середовище Delphi для створення програми «Кулінарні рецепти».

РОЗДІЛ І.ЗНАЙОМСТВО З СЕРЕДОВИЩЕМ DELPHI .

1.1. Історія розвитку середовища Delphi.

Середовище програмування Delphi було розроблено компанією Borland у 1993 році. Назва програмного продукту походить від назви давньогрецького міста Дельфі, що у Фокіді.

Delphi - це нащадок Турбо Паскаля, який був випущений для операційної системи Cp/m в 1983 році. Улютому1994 року Турбо Паскаль був перенесений на операційну систему MS-DOS. На ранньому етапі розвитку комп'ютерів IBM PC, Турбо Паскаль був однією з найбільш популярних мов розробки програмного забезпечення - головним чином тому, що це був цілком серйозний компілятор, який, включаючи компілятор, редактор і відгадчик. Середовище мало змогу працювати на машині з 64 Kb оперативної пам'яті.

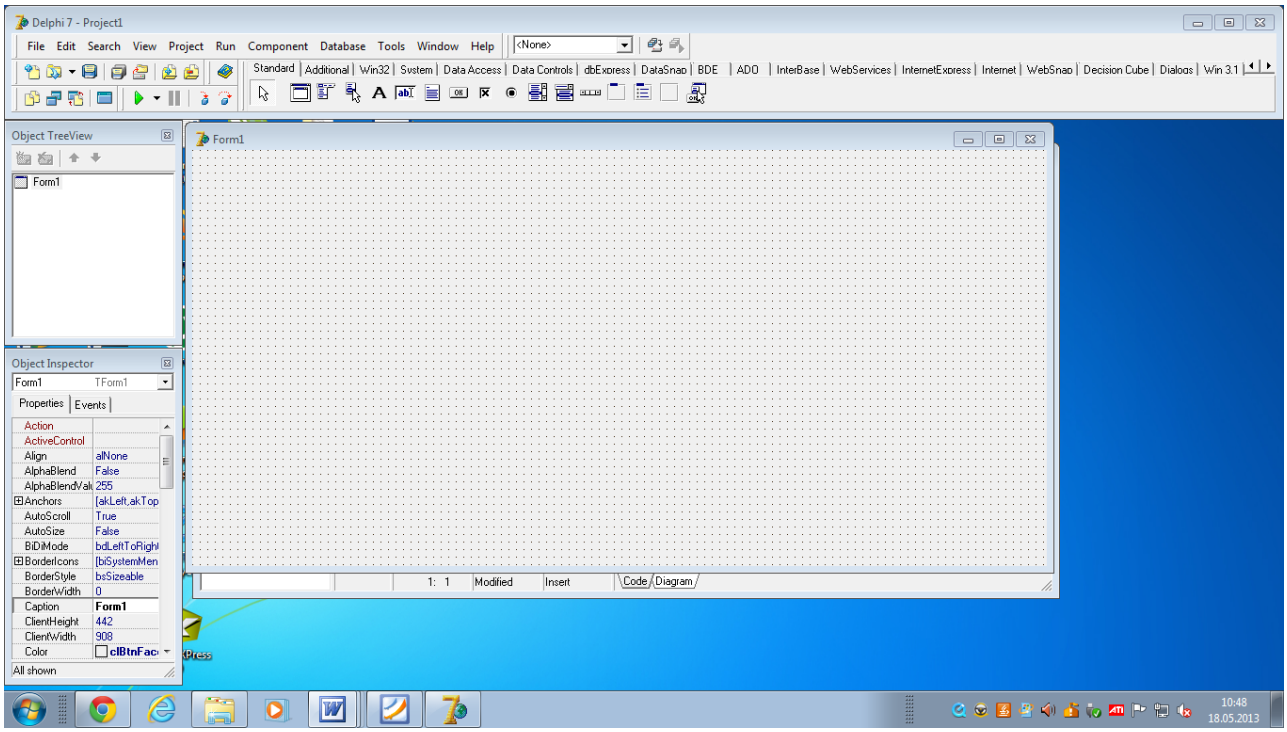
Під Windows - Турбо Паскаль був перенесений фірмою Borland в 1990 році. А найостанніша версія Borland Pascal 7.0 (що має тепер таку назву), не рахуючи Delphi, вийшла в світ в 1992 році. Розробка Delphi почалася в 1993 році. Після проведення beta-тестування Delphi показали на "Software Development '95".Спочатку на Delphi можна було програмувати під MS Windows 3.1. Починаючи з версії 2.0 на Delphi можна створювати програми під будь-яку з 32-бітних версій MS Windows.

В 2000 році була спроба створити варіант Delphi під операційну систему на базі ядра Linux, така модифікація Delphi мала назву Kylix. Було випущено 3 версії Kylix, проте експеримент виявився невдалим і 2003 року проект був заморожений.

2003 року була створена модифікація мови під платформу Microsoft.NET, що отримала назву Delphi.NET. Цей варіант мови послідовно розвивається в версіях Delphi 8, 2005, 2006, 2007.

Частково Delphi підтримується також у відкритому проекті FreePascal, що потенційно дозволяє створювати програми під велику кількість платформ.

Розглянемо тепер середовище програмування Delphi.



1.2. Головне меню середовища Delphi.

Головне меню Delphi розташоване у верхній частині робочого вікна середовища і має вигляд.

У ньому розташовані головні настройки та дії, що пов'язані як з проектом, так і з самим середовищем.

Розглянемо деякі компоненти меню:

File.

Зупинимось спочатку на пункті File.(див.Рис. 1).

Рис. 1

New… – Створити нові елементи: на екрані виводиться перелік елементів, які можна створити. Це нова програма, нова форма, новий модуль тощо.

New Application – Створити нову програму (цей елемент також є і у переліку New…)

New Form – Створити нову форму (цей елемент також є і у переліку New…)

Open… - Відкрити існуючий проект, модуль та ін.

Save – Зберегти зміни у поточному проекті чи модулі. Якщо до цього часу проект чи модуль не зберігався на диск, то автоматично Delphi присвоїть йому ім’я Project1 або Unit1 відповідно.

Save As… - Зберегти проект ти модуль під іншим ім’ям на диск.

Save Project As… - Зберегти проект під іншим ім’ям

Save All - Зберегти всі зміни у модулях та проектах на диск

Close – Закрити поточне вікно проекту чи модуля

Close All – Закрити всі вікна з елементами проекту

Exit – Вихід з програми

1.2.3. Головне вікно.

Головне вікно здійснює основні функції керування проектом створюваної програми. Воно завжди присутнє на екрані і займає його верхню частину. Не намагайтеся його розгорнути на весь екран: навіть у максимізованому стані його розміри і положення майже не відрізняються від звичайних. Пов'язано це з функціональністю головного вікна: з одного боку, воно несе в собі елементи, що завжди повинні бути під рукою у програміста, з другого - вікно не повинне віднімати в інших вікон Delphi значного простору екрана. Мінімізація головного вікна призводить до зникнення з екрана інших вікон Delphi (ці вікна з'являться, як тільки будуть відновлені розміри головного вікна), а його закриття означає закінчення роботи програміста із системою програмування закриття означає закінчення роботи програміста із системою програмування.

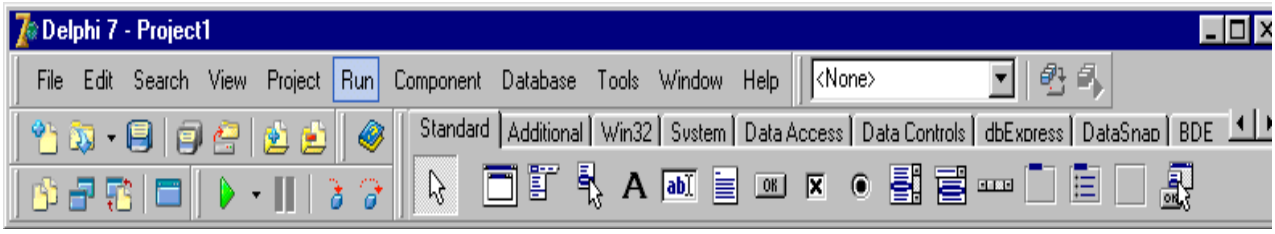


Рис. 2

Всі елементи головного вікна розташовуються на спеціальних панельках, у лівій частині яких знаходяться кнопки керування, що дозволяють за дрпомогою миші перетягувати панельки з розміщеними на них елементами. Будь-яку панельку (крім головного меню) можна забрати з вікна (зробити її невидимою) або “пустити плавати” по екрану в окремому вікні. Для цього потрібно лише “стягнути” панельку за допомогою миші за межі головного вікна.

1.3.Піктографічні кнопки.

Піктографічні кнопки відкривають швидкий доступ до найбільш важливих опцій головного меню. За функціональною ознакою вони розділені на сім груп:

1. **Група Standard**
2. **Група View**
3. **Група Debug**
4. **Група Custome**
5. **Група Desktops**
6. **Група Internet.**

1.4. Палітра компонентів.

Палітра компонентів займає праву частину головного вікна і має закладки, що забезпечують швидкий пошук потрібного компонента. Під компонентом розуміється якийсь функціональний елемент, що містить визначені властивості і розміщується програмістом у вікні форми. Компоненти являють собою елементи, з яких конструюється видиме зображення, що створюється працюючою.

Рис. 3

Як і панель кнопок, палітра компонентів може налагоджуватись. Для цього використовується спеціальний редактор, вікно якого з'являється на екрані після клацання правою кнопкою миші на будь-якій піктограмі в палітрі компонентів і вибору опції properties (Властивості)Object Pascal.

Стандартні компоненти вкладки Standart

Познайомимося із компонентами вкладки **Standart** Палітри компонентів. На першій сторінці Палітри компонентів розміщені 14 об'єктів. Це стандартні для Windows елементи інтерфейсу: списки, кнопки, текстові поля і т.п. Набір і порядок компонентів на кожній сторінці не є постійним. Так, можна додати до наявних компонентів кнопки, текстові поля і т.п. Набір і порядок компонентів на кожній сторінці не є постійним. Так, можна додати до наявних компонентів нові, змінити їхню кількість і порядок. На **рис. 3** зображено зовнішній вид Палітри компонентів.

Далі наведемо перелік основних компонентів вкладки **Standart:**

• **TMainMenu** - дозволяє помістити командне меню в програму. При розміщенні **TMainMenu** на формі це виглядає, якпросто іконка. Компоненти даного типу називають "невізуальними компонентами", оскільки вони невидимі під час виконання програми.

• **TPopupMenu** - дозволяє створювати випливаючі меню. Цей тип меню з'являється після натискання правої кнопки миші.

• **TLabel -** служить для відображення тексту на екрані.

• **TEdit** - стандартний керуючий елемент Windows для введення інформації. Він може бути використаний для відображення короткого фрагменту тексту дозволяє вводити текст під час виконання програми.

• **TMemo -** стандартний керуючий елемент Windows для введення великих фрагментів тексту. **TMemo** може виконувати основні функції редактора.

• **TButton** - дозволяє виконати будь-які дії при натисканні кнопки під час виконання програми.

• **TCheckBox -** відображає рядок тексту з маленьким віконцем поруч. У віконці можна поставити прапорець, що означає, що щось обрано.

• **TRadioButton -** дозволяє вибрати тільки одну опцію з декількох.

• **TListBox** - потрібний для показу списку, що випадає.

• **TComboBox -** багато в чому нагадує **ListBox**, за винятком того, що дозволяє вводити інформацію в поле введення. Є кілька типів **ComboBox**, але найбільш популярний, що випадає униз (dropdown combo box).

• **TScrollbar -** смуга прокручування**,** з'являється автоматично в об'єктах редагування.

• **TGroupBox -** використовується для візуальних цілей і для вказівки Windows, який порядок переміщення за компонентами на формі (при натисканні клавіші TAB).

• **TPanel** - керуючий елемент, схожий на **TGroupBox**, застосовується з декоративною метою. Компонент **TPanel**, розміщується на форму, а потім на

нього розташовуються інші компоненти. Тепер при переміщенні **TPanel** ці компоненти пересуваються разом з ним. **TPanel** можна також використовувати для створення лінійки інструментів і вікна статусу.

• **TScrollBox –** являє собою місце на формі, яке можна “прокрутити” у вертикальному чи у горизонтальному напрямах. Якщо в явному вигляді не вимкнути цю можливість, форма сама по собі так і діє. Однак, може знадобиться “прокрутити” тільки частину форми, у таких випадках і використовується TScrollBox.

.

1.5. Вікно форми.

Вікно форми в Delphi можна трактувати як своєрідний будівельний майданчик, на якому відповідно до плану розташовують компоненти майбутньої програми.

Форма - це вікно Windows, побудоване одним із допустимих стилів. Зверху над вікном є смуга з заголовком і стандартними кнопками керування вікном. Решта простору вікна форми є робочою ділянкою, що ознак середовища. Сітка призначена для вирівнювання компонент під час їх розміщування на площині форми. Водночас зі створенням нового проекту автоматично створюється нова форма, яка з'являтиметься під час виконання готової програми. Таку форму називають головною формою проекту. Якщо ж проект потребує декількох форм, то нову стандартну форму можна додати команду File ® New Form, а шаблон форми вибрати в діалоговому вікні командою File ® New ® Forms. Шаблон містить готовий набір елементів керування для типових випадків розробки проектів. Delphi дає змогу створювати власні шаблони та зберігати їх разом зі стандартними. Розглянемо головні етапи роботи з компонентами в робочій ділянці вікна форми. Активна в конкретний момент компоненту (та, що має фокус) виділена контурною рамкою з маленьких чорних квадратиків по периметру. Фокус між компонентами переміщують мишкою або клавішею Tab.

Декілька компонентів, розташованих на формі, можна об'єднати в групу. Для цього треба натиснути на ЛКМ і пересунути її так, щоб ділянка, обмежена штриховою лінією, що тягнеться за мишкою, захопила потрібні компоненти. Після відпускання клавіші мишки всі об'єднані в групу компоненти будуть виділені контурною рамкою кожен, однак сірого кольору, сама рамка зафіксована лише по кутках кожного компоненту. Щоб відмінити об'єднання компоненту, досить клацнути мишкою в будь-якому вільному від

компоненту місці робочої ділянки вікна форм. Для вилучення компоненти з форми потрібно перевести на неї фокус і натиснути на клавішу Del або вибрати з головного меню команду Edit ® Delete. Потрібно вилучити всі відразу компоненти, об’єднані попередньо в групі. Вилучену компонентну або групу відновлюють командою Edit ® Undelete.

У межах робочої ділянки вікна пересування компоненту найзручніше мишкою на ліву клавішу. Для точного розташування компоненту у вікні використовують комбінацію клавіш керування курсором та клавіші Ctrl. Змінити розташування компоненти можна також шляхом задання відповідних координат її лівого верхнього кута в інспекторі об’єктів (властивості Left, Top).

Розмір компоненту найзручніше змінювати також мишкою. Для цього спершу компоненту роблять активнішою. Далі пересувають мишкою на один з маркерних чорних квадратів, розташованих по периметру. Курсор набуває вигляду двонапрямленої стрілки. Зафіксувавши ЛКМ, пересувають мишкою в потрібному напрямі і відпускають клавішу. Точну зміну розмірів виконують комбінацією клавіш керування курсором та клавіші Shift. Крім того, розміри можна визначити відповідними властивостями (Height, Width) в інспекторі об’єктів. Розміри та розташування всіх компонентів у вікні можна одночасно і пропорційно змінити. Для цього використовують команду головного меню Edit ® Scale, зазначивши зміну масштабу у відсотках.

Якщо у вікні форми вже розташовані декілька компонентів, то їх можна вирівнювати як щодо одної. Для цього застосовують команду головного меню Edit® Align або палітру вирівнювання, яку викликають командою View ® Alignment Palette. Ця палітра є окремим вікном, яке можна розташувати на екрані в зручному місці на тривалий час. Вирівнювання виконують для окремих компонентів, а також для груп. Компоненти в групі вирівнюються стосовно того компоненту, яка потрапила в групу першою. Над компонентами та їхніми групами можна виконувати операції вирізання, копіювання в буфет обміну,

вставлення з буфету обміну. Однак копіювання та вирізання можливе лише для таких груп компоненту, які мають спільного “родича” (форму або компоненту-контейнер).

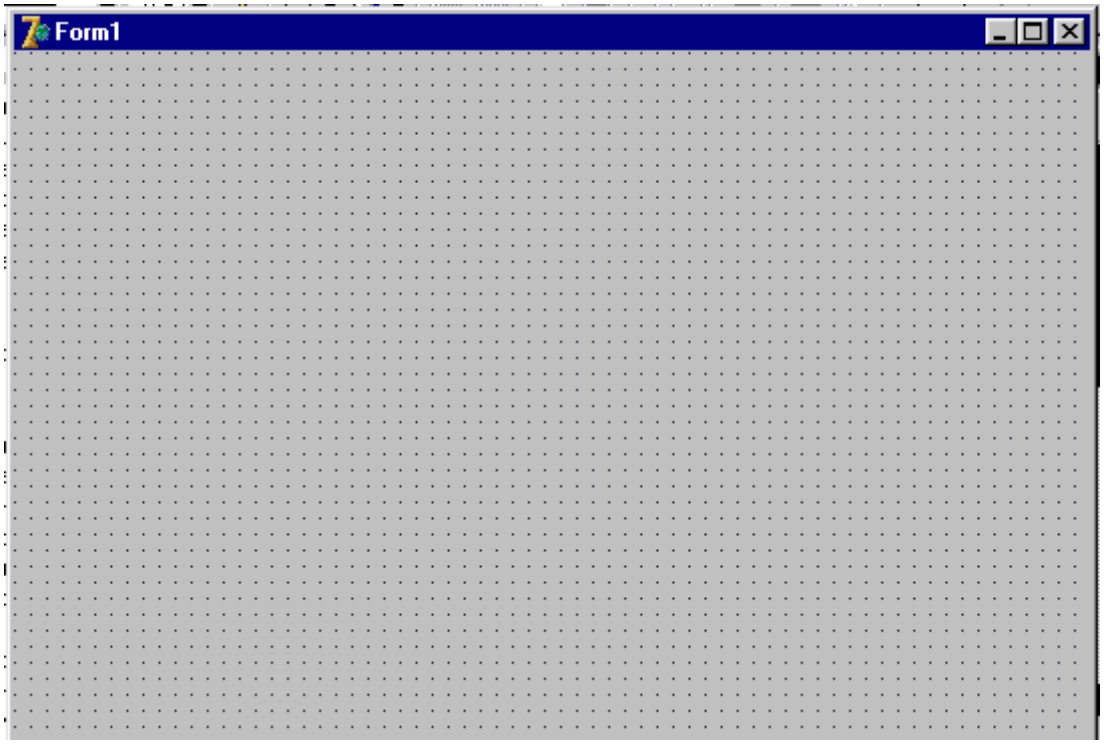


Рис. 4

1.6. Вікно дерева обє'ктів.

Це вікно призначене для візуального відображення зв'язків між окремими компонентами, розташованими на активній формі або в активному модулі даних. Клацання по будь-якому компонету в цьому вікні активізує компонент у вікні форми і відображає властивості цього компонента у вікні Інспектора об'єктів. Подвійне клацання приводить до спрацьовування механізму Code Insight, що вставляє у вікно коду заготовку для обробника події OnClick. Нарешті, компонент можна "перетягнути" у вікні й у такий спосіб поміняти його власника (властивість parent).

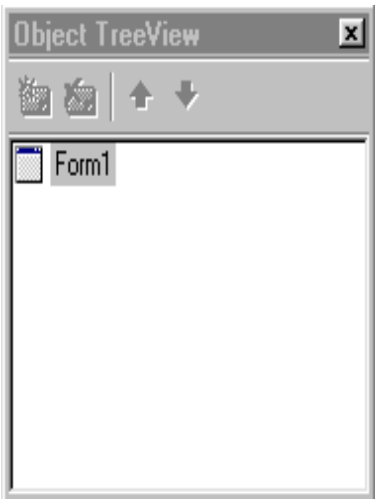


Рис.5

1.7. Вікно інспектора обє'ктів.

Інспектор об'єктів є інструментом середовища розробки Delphi, який дає змогу повністю визначити вигляд та поведінку компоненту, що розташовані на формах проекту, а також самої форми. За його допомогою можна задавати потрібні значення властивостей компоненту (об'єктів), а також реакцією на стандартні події. Інспектор об'єктів розташований в окремому вікні, яке створюється автоматично під час завантаження Delphi.

Вікно інспектора об'єктів має список компонентів поточної форми, а також дві сторінки: властивостей **Properties** та подій **Events**.

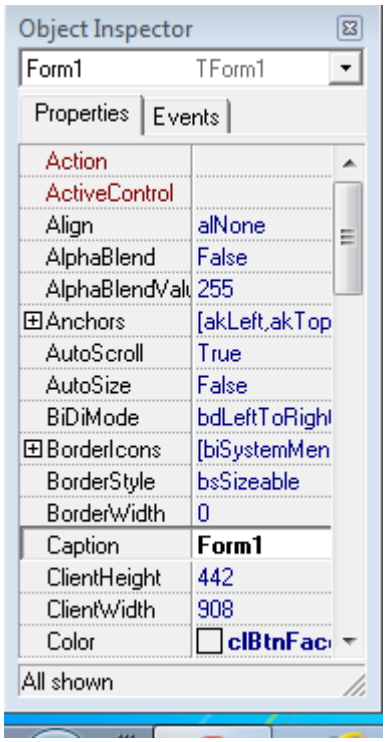


Рис.6

Дані з'являються на сторінках інспектора об'єктів лише за наявності відкритого файлу з текстом хоча б одного модуля в редакторі коду.

У верхній частині інспектора розміщений список компонентів в алфавітному порядку, які розташовані на активній формі, включаючи саму форму. Побачити список можна, натиснувши на кнопку списку компонентів. Якщо вибрати який-небудь компонент із списку мишкою, то компонент стає активним, а обидві сторінки інспектора об’єктів заповняться значеннями її опублікованих властивостей та іменами процедур опрацювання подій, у вікні форми відповідно компонент буде виокремлена контурною рамкою. Якщо в списку нема імені, то це означає, що сторінки інспектора об’єктів містять властивості та методи опрацювання подій групи об’єднаних компонентів. Видиме ім’я активного компоненту побудоване з двох частин: власне імені компоненту та імені типу (класу), до якого вона належить.

Сторінка властивостей складається з двох стовпців: лівий з назвами імен властивостей компоненту, а правий з їхніми значеннями. Для поточної властивості в правому стовпці стає активним поле редактора, вигляд якого залежить від конкретної властивості. Якщо поле редактора не містить ніяких позначок, то це властивість, значення якої треба ввести з клавіатури. Якщо ж поле має кнопку значень переліченого типу (зі стрілкою вниз), то значення можна вибрати зі списку допустимих наперед визначених значень. Побачити список можна, якщо натиснути на цю кнопку. Деякі властивості (наприклад, Font або Icon) для вибору значень мають діалогове вікно. В полі редактора властивостей у цьому випадку видно кнопку з трьома крапками. Після натискання на неї з’явиться діалогове вікно, в якому треба задати по черзі декілька різних значень, що стосується тієї властивості, а потім його закрити.

Сторінка подій також має два стовпці. Лівий – з іменами стандартних подій, наякі може реагувати компоненту, а правий – з іменами процедур (методів) опрацювання подій, які реалізують реакцію компоненту. Стандартні події виникають під час створення та знищення компоненту, зміни її видимості, натискання на клавіші клавіатури, клацання мишкою тощо. Багато компонентів можуть реагувати на специфічні, характерні лише для цих подій. Кожній

стандартній події відповідає фіксоване ім’я методу (процедури) опрацювання. Початково правий стовбець є порожній, тобто компоненту не реагує на жодні події. Для визначення реакції компоненту на подію потрібно перевисти курсор на поле редактора події в правому стовпці та двічі клацнути ЛКМ. Після цього в стовпці імен з’явиться ім’я методу опрацювання, а до вікна відповідного модуля в редакторі коду автоматично додасться базовий (початковий) текст цієї програми, і курсор розташується на місці майбутнього першого оператора. Відповідні заголовки методів так само додають до опису класу форми, на якій розташована компоненти. Початковий текст програми не дає жодного оператора, отже, фактичної реакції компоненту на дію не буде доти, доки не будуть записані в тексті потрібні оператори. Допустимих імен методів опрацювання тієї ж подій може бути і декілька, тоді потрібне ім’я можна вибрати зі списку, який відкривають за допомогою кнопки. Методам опрацювання подій можна давати і власні імена, однак без належного досвіду цього робити не варто. Якщо декілька різних компонентів можуть реагувати на одноімунну подію, то можна задати для всіх них подібну реакцію створенням лише одного методу опрацювання. Це особливо актуально, якщо компоненти однотипні, наприклад, група кнопок або група позначок. Для цього такі компоненти треба спочатку об’єднати в групу (див.вище). У списку компонентів у верхній частині вікна після цього не буде жодного імені. Тепер визначення реакції подію буде стосуватися відразу всіх компонентів групи.

1.8. Вікно коду.

Редактор коду розташований в окремому вікні, яке можна закрити та відкривати незалежно від головного та інших вікон. У цьому вікні відображають та редагують тексти програм. Праворуч та знизу від тесту є стандартні смуги його перегляду. Зверху над текстом зображені закладки для позначення файлів. Кожна закладка містить ім'я відповідного модуля. Для перемикання вікна на потрібний модуль достатньо клацнути мишкою на його закладці.

У нижній частині вікна є рядок стану. Він містить інформацію про місце курсора на активній сторінці (перше поле ліворуч), друге поле відображає, чи був зміщений текст у цьому вікні з моменту останнього запам'ятовування, третє поле повідомляє про режим набору символів на клавіатурі: Insert – уставлення, Overwrite – зміна, Real Only – текст не можна змінювати. Перемикання між режимами Insert та Overwrite виконують клавишею **Ins**. Крім того, рядок стану призначений для виведення повідомлень компілятора про помилки в програмі, а також для введення тексту під час виконання команди Search® Incremental Search. Смугу ліворуч від тексту використовують для відокремлення рядків, на яких будуть розташовані місця зупинки програми. Для цього достатньо клацнути мишкою на смузі навпроти потрібного рядка. Повторне клацання мишкою змінює місце зупинки.

Редактор коду може працювати спільно з налагоджувачем Delphi. Налагоджування дає змогу трасувати програму безпосередньо у вікні редактора і відображати в ньому всю поточну інформацію. Він виконує значну частину роботи програміста. Зокрема, під час перенесення у вікно форми нових компонентів у клас форми автоматично додає відповідні поля, а в проекті – відповідні модулі.

Зауважимо, що редактор коду можна використовувати для редагування будь-яких текстових файлів, наприклад, файлів вхідних даних програми, і навіть файлів, не пов’язаних з цією програмою.

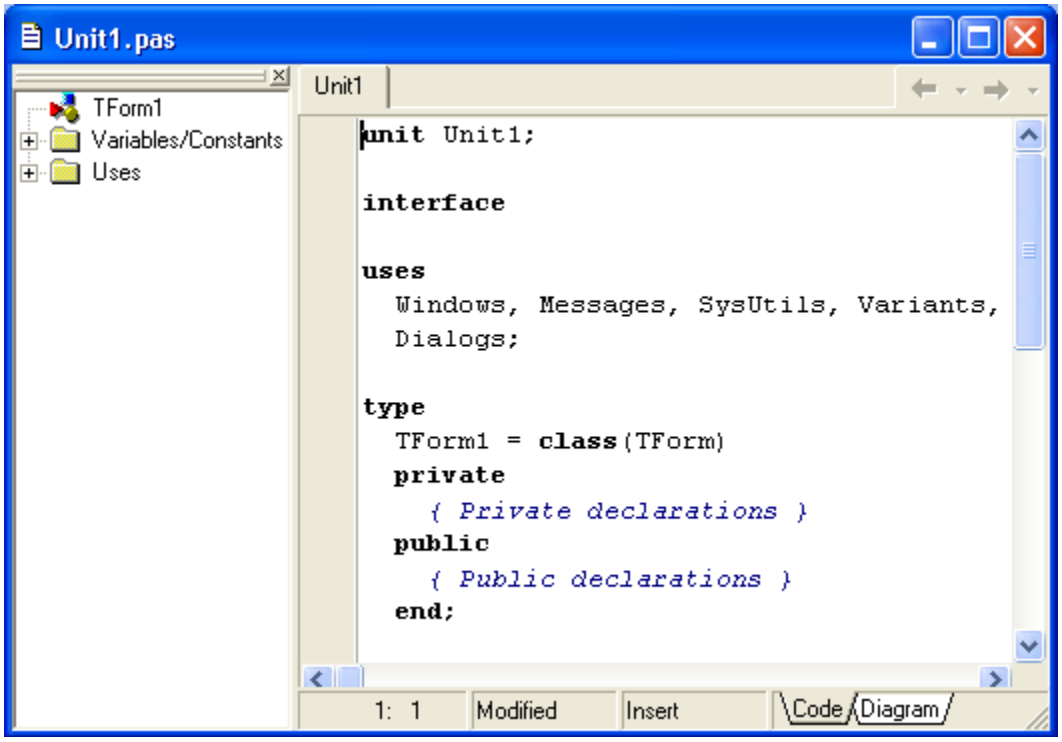


Рис. 7

1.9. Структура проекту.

Середовище Delphi містить у собі повний набір візуальних інструментів для швидкої розробки програм (RAD - rapid application development), що підтримує розробку інтерфейсу користувача та підключення до корпоративних баз даних. VCL - бібліотека візуальних компонентів, що містить стандартні об’єкти побудови інтерфейсу користувача, графічні об’єкти, діалоги, об’єкти мультимедіа, об’єкти керування базами даних, об’єкти керування файлами тощо.

Саму розробку програми можна поділити на дві частини: візуальну та алгоритмічну. У Delphi візуальна частина просто “ліпиться”, як скульптор з кусків глини ліпить фігуру. Центром “ліплення”, полотном, де будуть розгортатись всі події програми є форма (Form), а вже на формі “ліпимо” компоненти, які є доступними у Delphi. Найпростіша програма складається з однієї форми, складні – з 2 і більше.

У Delphi розробляється проект програми. При цьому автоматично створюється кілька файлів:

project1.dpr – delphi project – саме ядро програми – текстовий файл де описана головна програма, що викликає всі компоненти.

unit1.pas – текстовий файл, де описаний алгоритм всіх подій, що відбуватимуться з елементами (компонентами) форми. Ця частина у Delphi носить назву модуль (unit).

unit1.dfm – файл опису розташування компонентів на полотні формі.

Таким чином, при написанні програми, алгоритмічна частина робота ведеться з pas–файлом. dpr–файл керується автоматично самим проектом при роботі з компонентами інтерфейсу. Спочатку вважатимемо, що кожній формі відповідає один модуль. Ще раз зауважимо, що у формі створюється інтерфейс, тобто “ліпляться” компоненти, а у модулі описуються події, що можуть відбуватись з цими компонентами.

При запуску або створенні нового проекту Delphi автоматично створює pas – файл форми з базовими елементами, необхідними для роботи проекту.

```
unit Unit1;

interface

uses

Windows, Messages, SysUtils, Classes, Graphics, Controls, Forms, Dialogs;

type

TForm1 = class(TForm)

private

{ Private declarations }

public

{ Public declarations }

end;

var
```

Form1: TForm1;

Implementation

*{$R *.DFM}*

end.

Слово *Unit* говорить про те, що це модуль.

Interface – блок програми, в якому визначаються типи, змінні, константи, процедури та функції, що доступні іншим модулям.

Uses – перераховані всі зовнішні (у даному випадку стандартні) модулі, які необхідні даному модулю в процесі роботи.

Type TForm1 = class(TForm) – створена форма належить до класу Tform. Tform – абстрактний об’єкт, у якому описані всі події та можливості керування всіма елементами форми. Він є спадкоємцем класу Tform, тобто, успадковує його властивості та методи, додаючи до них власні.

РОЗДІЛ ІІ. ОСНОВИ ВІЗУАЛЬНОГО ПРГРАМУВАННЯ.ОБЄДНАННЯ ЕЛЕМЕНТІВ УПРАВЛІННЯ.

2.1. Основні концепції візуального програмування в Delphi.

Програмування в Delphi будується на взаємодії двох процесів: процессу конструювання візуального прояву програми (тобто її Windows-вікна) і процессу написання коду, що додаєелементам цього вікна і програмі в цілому необхідну функціональність. Для написання коду використовується вікно коду, для конструювання програми - інші вікна Delphi, насамперед - вікно форми. Між змістом вікон форми і коду існує нерозривний зв'язок, що строго відслідковується Delphi. Це означає, що розміщення на формі компонента приводить до автоматичної зміни коду програми і навпаки - видалення тих або інших автоматично вставлених фрагментів коду може привести до видалення відповідних компонентів. Пам'ятаючи про це, програмісти спочатку конструюють форму, розміщуючина ній черговий компонент, а вже тільки після цього переходять, якщо це необхідно, до написання фрагмента коду, що забезпечує необхідну поведінку компонента в працюючій програмі. Щоразу, коли Ви створюєте нову форму, створюєтьсямодуль. У програмі може бути і, найчастіше буває, не одна, а декілька - іноді кілька десятків форм і зв'язаних з ними модулів. У першому наближенні модулем можна вважати самостійний розділ програми, в чомусь подібний до глави в книзі. При компіляції програми Delphi створює файли з розширеннями pas, dfm і dcu для кожного модуля. Файл із розширенням pas містить копію тексту з вікна коду програми, у файлі з розширенням dfm зберігається опис змісту вікна форми, а в dcu-файлі - результат перетворення в машинні інструкції тексту з обох файлів. Файли dcu створюються компілятором і дають необхідну базу для роботи компонувальника, що перетворить їх у єдиний файл, що завантажується, з розширенням exe.

2.2. Властивості компонентів.

Кожний компонент, який ви розміщуєте на формі, має своє відображення у вікні Інспектора об’єктів (Object Inspector). Інспектор об’єктів має дві “сторінки” - “Properties” (Властивості) та “Events” (Події). Створення програми в Delphi зводиться до “розміщення” компонент на форму (яка також є компонентом) та настройки взаємодії між ними шляхом:

• заміни значення ***властивостей*** цих компонент,

• написання адекватних реакцій на ***події***.

Властивість є важливим атрибутом компонента. Для користувача (програміста) властивість виглядає як просте поле будь- якої структури, що містить деяке значення. Однак, на відміну від “просто” поля, будь-яка заміна значення деякої властивості любого компонента відразу призводить до зміни візуального. Представлення цього компонента, оскільки властивість інкапсулює у собі методи (дії, процедури, функції), пов’язані з читанням та записом цього поля. Властивості служать двома головним цілям. По-перше, вони визначають зовнішній вид форми чи компонента. А по-друге, властивості визначають поведінку форми чи компонента.Існує декілька типів властивостей, в залежності від їх “природи”, тобто внутрішньої конструкції.

• **Прості** властивості – це ті, значення яких є числами або рядками. Наприклад, властивості **Left** та **Top** приймають цілі значення, що визначають положення лівого верхнього кута компонента чи форми. Властивості **Caption** та **Name** представляють собою рядки та визначають заголовок та ім’я компонента чи форми.

• **Перераховні (** *рос.* Перечислимые **)** властивості – це ті, які можуть приймати значення з поперед визначеного списку. Найпростіший приклад – це властивість типу **Boolean**, яка може приймати значення *True* чи *False*.

• **Вкладені** властивості – це ті, які підтримують вкладені значення (чи об'єкти). Інспектор об'єктів зображує знак "+" зліва назви таких властивостей. Є два види таких властивостей: *множини значень* та *комбіновані значення*. Інспектор об'єктів зображує множини у квадратних дужках. Якщо множина пуста, то це відображається як []. Установки для вкладених властивостей виду "множина" звичайно мають значення типу. Найбільш розповсюдженим прикладом такої властивості є властивість **Style** з вкладеною множиною значень. Комбіновані значення відображаються в Інспекторі об'єктів як колекція деяких величин, кожний зі своїм типом даних. Деякі властивості, наприклад, **Font**, для зміни своїх значень мають можливість визвати діалогове вікно. Для цього достатньо натиснути маленьку кнопку з трьома точками у правій частині рядка Інспектора об'єктів, що показує дану властивість.

Середовище Delphi дозволяє легко маніпулювати властивостями компонент як у режимі проектування (design time), так і в режимі виконання програми (run time).

2.3.Керування властивостями візуальних компонентів в режимі проектування.

В режимі проектування маніпулювання властивостями виконується за допомогою Дизайнера форм (Forms Designer) чи, на сторінці “Properties” Інспектора об’єктів. Наприклад, для того щоб змінити властивість **Height** (Висота) та **Width** (Ширина) кнопки, достатньо “заціпити” мишкою за будь-який її кут та розсунути до потрібного представлення. Цього ж результату можна досягти, якщо присвоїти нові значення властивостям **Height** та **Width** у вікні Інспектора об’єктів. З іншого боку, в **режимі виконання** користувач (програміст) має можливість не тільки маніпулювати всіма властивостями, що відображаються в Інспекторі об’єктів, а й керувати більш широким їх списком.

2.4. Керування властивостями візуальних компонентів в режимі виконання програми.

Всі заміни значень властивостей компонентів в режимі виконання повинні виконуватись шляхом прямого запису коду на мові Pascal. В режимі виконання неможливо використовувати Інспектор об’єктів. Проте, доступ до властивостей компонентів можливо отримати програмним шляхом. Для зміни якоїсь властивості необхідно написати програмний код, наприклад: NameComponent.Width := 35; Вищепоказаний рядок встановлює ширину (**Width**) компонента **NameComponent** у значення 35. У момент виконання даного рядка програми, компонент візуально змінить свою ширину. Таким чином, Інспектор об’єктів є зручним способом виконання в режимі

проектування того, що може бути здійснено програмним шляхом в режимі виконання. Більш того, як уже було сказано вище, у компонента можуть бути властивості, що не відображуються у вікні Інспектора об’єктів.

Об’єктно-орієнтована мова Object Pascal, що лежить в основі Delphi, як базовий має принцип відповідності візуальних компонент тим речам, які вони представляють. Розроблювачі Delphi поставили перед собою мету, щоб, наприклад, подання компонента **Button** (Кнопка), що інкапсулює деякий код, відповідало візуальному зображенню кнопки на екрані й являлося близьким еквівалентом реальної кнопки, яку можна знайти на клавіатурі. А саме з цього принципу народилось поняття “властивість”. Якщо змінити властивість **Width** та **Height** компонента **Button**, кнопка відповідним чином змінить свої ширину та висоту. Немає необхідності після зміни властивості **Width** вказувати об’єкту, щоб він перемальовував себе, хоч при звичайному програмуванні сааме так и треба робити. Властивості - це більш ніж просто дані. Властивості роблять ілюзію, начебто розроблювач має справу з реальними об’єктами, а не з їх програмним зображенням.

2.5. Моделювання форми.

1. Зміна заголовку вікна форми. За замовчуванням заголовок форми (властивість **Caption**) збігається з ім’ям форми (властивість **Name**), наприклад **Form1**. Щоб змінити заголовок, потрібно звернутися до вікна Інспектора Об'єктів, вкладка **Properties**, властивість **Caption**, де змінити заголовок Form1 на «Кулінарні рецепти». Після чого запустити на виконання програму клавішею **<F9>**. Отже, Інспектор об'єктів дозволяє дуже просто змінювати властивості форми, а так само інших об'єктів, розміщених на формі.

2. Зміна розмірів і положення компонентів на формі. Виділіть кліком миші текстову мітку **Label**. Компонент **Label**, як і усі компоненти мають такі властивості: **Left** (положення лівого верхнього кута компонента - ліворуч); **Top** (положення лівого верхнього кута компонента - зверху); **Width** (розмір компонента - ширина); **Height** (розмір компонента - висота). Змініти розміри і положення компонентів розміщених на Формі можна за допомогою миші та у вікні Інспектора об'єктів, задавши значення для відповідних властивостей.

3. Зміна властивостей компонента Label. Змініть текст мітки за допомогою властивості **Caption**. Змініть колір і накреслення тексту мітки за допомогою властивості **Font** Інспектора об'єктів, для цього натиснувши на кнопку біля властивості **Font,** відкрийте вікно діалогу настроювання шрифту. Виберіть атрибути тексту за бажанням.

4. Виділення об’єктів на формі. Для виділення декількох об'єктів одночасно натисніть клавішу <**Shift**> і клацніть на першому і потім на другому об’єкті. Тепер обидва об'єкти мають по краях маленькі квадратики, що показують, що об'єкти обрані. Вибравши два або більш об'єкти одночасно, Ви можете виконати операції над ними. Наприклад, пересувати на формі.

5. Зміна кольору форми. Для цього використовується властивість **Color** об'єкту **TForm**. Є три способи змінити його значення в Інспекторі об'єктів. Перший - просто надрукувати ім'я кольору (**clRed**) або номер кольору. Другий шлях - натиснути на маленьку стрілку праворуч і вибрати колір зі списку. Третій шлях - двічі клацнути на поле уведення властивості **Color**. При цьому з'явиться діалог вибору кольору.

6. Наповнення списку ListBox. Для цього двічі клацніть на властивість **Items** об'єкта **ListBox**. З'явиться вікно діалогу, у якому вводять рядки, які будуть відображені в списку **ListBox**. Надрукуйте кілька слів, по одному у кожному рядку, і натисніть кнопку OK. Текст відобразиться в списку **ListBox.**

7. Зміна тексту на кнопці. Виділіть кнопку **TButton**. Текст, що Ви бачите на поверхні кнопки - це вміст властивості Caption. Змініть значення властивості **Caption** на «кнопка» - змінився текст на кнопці.

8. Зміна імені компонентів. Властивість **Name** служить для внутрішніх посилань і використовується при написанні коду програми. Змініть значення властивості **Name** кнопки на „Terminate” - це ім'я екземпляра класу **TButton.** Відкрийте вікно Редактора коду і досліджуйте розділ опису класу форми. У цьому фрагменті кнопка TButton називається Terminate через те, що таке значення має властивість Name. Об’єкти **TMemo**, **TListBox**, **TLabel** мають імена, що привласнюються за замовчуванням.

РОЗДІЛ ІІІ. ПРАКТИЧНА ЧАСТИНА

1.1. Опис практичної частини.

Практична частина моєї дипломної роботи полягала у створенні програми «Кулінарні рецепти». Попередньо мною був вивчений та підготовлений з даної теми матеріал. Мною були підібрані картинки страв, переведені через конвертер у картинки з розширенням .bmp. Отже, хід виконання практичного завдання наступний:

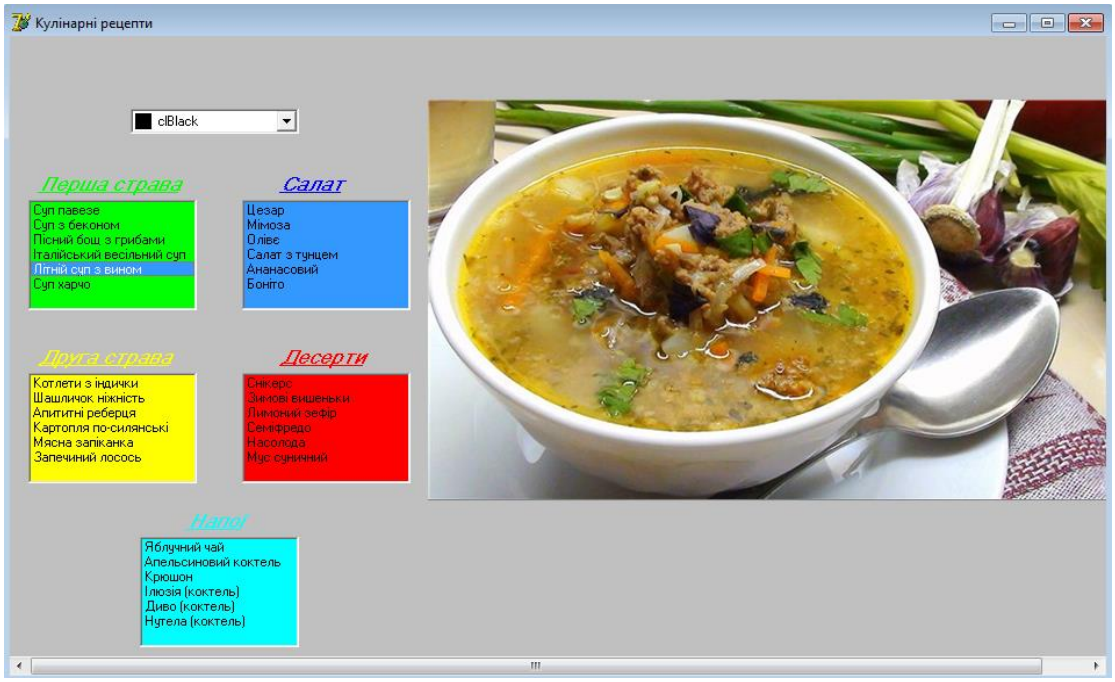
1. Запустила програму Delphi.
2. Реалізувала неведену задачу в середовищі програми Delphi.
3. На форму винисимо ListBox1, ComboBox1, Image1, Label1, Label2, Label3, ColorBox1,Image2.Програма повинна надати відвідувачі можливість вибору страв із меню: салату, першої страви, другої страви, десерту та напоїв. При виборі страви має зявитися її зображення(пам’ятаємо, шо всі зображення повинні бути у форматі bmp).
4. Для реалізіції створила наступні процедури:

```
procedure TForm1.ColorBox1Change(Sender: TObject);
begin
  Form1.Color:=ColorBox1.Selected;
end;
procedure TForm1.ListBox1Click(Sender: TObject);
begin
case ListBox1.ItemIndex of
0:Image1.Picture. LoadFromFile('f:\1\PS\1.bmp');
1:Image1.Picture. LoadFromFile('f:\1\PS\2.bmp');
2:Image1.Picture. LoadFromFile('f:\1\PS\3.bmp');
3:Image1.Picture. LoadFromFile('f:\1\PS\4.bmp');
4:Image1.Picture. LoadFromFile('f:\1\PS\5.bmp');
5:Image1.Picture. LoadFromFile('f:\1\PS\6.bmp');
end;end;
procedure TForm1.ListBox2Click(Sender: TObject);
begin
   case ListBox2.ItemIndex of
```

```
0:Image1.Picture. LoadFromFile('f:\1\DS\11.bmp');
1:Image1.Picture. LoadFromFile('f:\1\DS\22.bmp');
2:Image1.Picture. LoadFromFile('f:\1\DS\33.bmp');
3:Image1.Picture. LoadFromFile('f:\1\DS\44.bmp');
4:Image1.Picture. LoadFromFile('f:\1\DS\55.bmp');
5:Image1.Picture. LoadFromFile('f:\1\DS\66.bmp');
end;end;
procedure TForm1.ListBox3Click(Sender: TObject);
begin
case ListBox1.ItemIndex of
0:Image1.Picture. LoadFromFile('f:\1\SL\1.bmp');
1:Image1.Picture. LoadFromFile('f:\1\SL\2.bmp');
2:Image1.Picture. LoadFromFile('f:\1\SL\3.bmp');
3:Image1.Picture. LoadFromFile('f:\1\SL\4.bmp');
4:Image1.Picture. LoadFromFile('f:\1\SL\5.bmp');
5:Image1.Picture. LoadFromFile('f:\1\SL\6.bmp');
end;end;
procedure TForm1.ListBox4Click(Sender: TObject);
begin
case ListBox1.ItemIndex of
0:Image1.Picture. LoadFromFile('f:\1\DSR\1.bmp');
1:Image1.Picture. LoadFromFile('f:\1\DSR\2.bmp');
2:Image1.Picture. LoadFromFile('f:\1\DSR\3.bmp');
3:Image1.Picture. LoadFromFile('f:\1\DSR\4.bmp');
4:Image1.Picture. LoadFromFile('f:\1\DSR\5.bmp');
5:Image1.Picture. LoadFromFile('f:\1\DSR\6.bmp');
end;end;
procedure TForm1.ListBox5Click(Sender: TObject);
begin
case ListBox1.ItemIndex of
0:Image1.Picture. LoadFromFile('f:\1\NP\1.bmp');
1:Image1.Picture. LoadFromFile('f:\1\NP\2.bmp');
2:Image1.Picture. LoadFromFile('f:\1\NP\3.bmp');
3:Image1.Picture. LoadFromFile('f:\1\NP\4.bmp');
4:Image1.Picture. LoadFromFile('f:\1\NP\5.bmp');
5:Image1.Picture. LoadFromFile('f:\1\NP\6.bmp');
end;end;end.
```

5. Створила повне меню з салату, першої страви, другої страви, десерту та напоїв, з вибором не менше 5 страв у кожній категорії.

5. Робоча форма програми матиме вигляд:



ВИСНОВКИ

Мова програмування Delphi дуже широко використовується і має дуже велику кількість розроблених елементів та засобів, що значно полегшують розробку додатків. У Delphi, використовується багато передових ідей і концепцій, закладених в графічному інтерфейсі Windows. У середовищі програмування Delphi є всі необхідні інструменти для того, щоб створювати повноцінні програми. Писати, компілювати і тестувати програму - все це можна робити, не виходячи з Delphi. Завдяки можливості вбудови в програму ассемблерних вставок, програміст отримує змогу створювати програми з інтерфейсом Windows та дуже високою швидкодією, характерною для низькорівневих мов програмування. Також Delphi має користувальницький графічний інтерфейс, подібний VisualBasic й C++. Весь вихідний текст програми на Delphi пишеться мовою Object Pascal (об'єктний Паскаль), практично нічим не відрізняється від принципів, закладених в Turbo Pascal. Синтаксис, принцип модульності, процедури, функції все взято за основу.

Отже, суть візуального програмування полягає в конструюванні розв'язку поставленої задачі методом вставляння компонентів (візуальних заготовок) у форму, наданні значень їхнім властивостям і в застосуванні чи створенні методів, потрібних для розв'язування задачі.

Отже в даній дипломній роботі я розглянула основні концепції візуального програмування мови Delphi, і як здійснюється об'єднання елементів управління.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ТАЛІТЕРАТУРИ

1. Морзе Н.В. Інформатика: підруч. Для 10 кл. загальноосвіт. навч. закл.: рівень стандарту/ Н.В.Морзе, В.П. Вембер, О.Г.Кузьмінська – К.: Школяр, 2010 – 304с.
2. http://office.microsoft.com/uk-ua/
3. http://www.liveinternet.ru/users/hecate_in_ua/post191865103/
4. http://kinotorba.narod.ru/progvideoredaktory1.html
5. http://www.vuho.com.ua/content/newsitem/id/93/page/1