ФАКУЛЬТЕТ АВТОМАТИКИ, ТЕЛЕМЕХАНІКИ ТА ЗВ'ЯЗКУ

Кафедра «Обчислювальна техніка та системи управління»

В.С. Меркулов, І.Г. Бізюк, О.В. Чаленко

ПРОГРАМУВАННЯ В СЕРЕДОВИЩІ Visual Basic 6.0

Конспект лекцій
з дисциплін
«ОБЧИСЛЮВАЛЬНА ТЕХНІКА, ПРОГРАМУВАННЯ, МОДЕЛЮВАННЯ СИСТЕМ», «ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ»

Частина І

Харків 2012

Меркулов В.С., Бізюк І.Г., Чаленко О.В. Програмування в середовищі Visual Basic 6.0: Конспект лекцій. – Харків: УкрДАЗТ, 2012. – Ч.1. – 78 с.

Даний конспект лекцій розроблено відповідно до програм з дисциплін «Обчислювальна техніка, програмування, моделювання систем» та «Обчислювальна техніка та програмування».

Метою конспекту лекцій є опанування студентами алгоритмічної мови високого рівня, здобуття необхідних знань для організації обчислень у середовищі Visual Basic 6.0 використання теоретичних засад об'єктно-орієнтованого підходу до розроблення програмних проектів.

Рекомендується для студентів спеціальностей ЕТ, ЕСК, В, Л усіх форм навчання.

Іл. 34, табл. 18, бібліогр.: 26 назв.

Конспект лекцій розглянуто і рекомендовано до друку на засіданні кафедри «Обчислювальна техніка та системи управління» 30 листопада 2010 р., протокол № 4.

Рецензент

проф. Г.І. Загарій

В.С. Меркулов, І.Г. Бізюк, О.В. Чаленко

ПРОГРАМУВАННЯ В СЕРЕДОВИЩІ Visual Basic 6.0

Конспект лекцій
з дисциплін
«ОБЧИСЛЮВАЛЬНА ТЕХНІКА, ПРОГРАМУВАННЯ, МОДЕЛЮВАННЯ СИСТЕМ», «ОБЧИСЛЮВАЛЬНА ТЕХНІКА ТА ПРОГРАМУВАННЯ»

Частина І

Відповідальний за випуск Бізюк І.Г.

Редактор Буранова Н.В.

Підписано до друку 25.01.11 р.

Формат паперу 60х84 1/16 . Папір писальний.

Умовн.-друк.арк. 2,75. Тираж 100. Замовлення №

Видавець та виготовлювач Українська державна академія залізничного транспорту
61050, Харків - 50, майдан Фейєрбаха, 7
Свідоцтво суб'єкта видавничої справи ДК № 2874 від 12.06.2007 р.

ЗМІСТ

ВСТУП

Середовище програмування QBasic використовується переважно для розроблення консольних додатків[1] (тобто програм, які виконуються в середовищі MS-DOS). Виникає питання: чим відрізняється Windows і як писати для нього програми?

Windows – це **графічний інтерфейс користувача** – **GUI** (Graphical User Interface).

Іноді його ще називають "візуальний інтерфейс" або "графічне віконне середовище" (поява – середина 70-х років). Зараз усі погоджуються, що графічний інтерфейс користувача є найважливішим "великим досягненням" у сфері ПК.
Усі графічні інтерфейси користувача уможливлюють використання графіки на растровому екрані дисплея.
Графіка дає краще сприйняття дійсного стану речей на екрані, візуально багате середовище для передачі інформації й можливість **WYSIWYG (What you see is what you get – що ви бачите, те й отримаєте)***, як для графіки, так і для форматованого друку документів тексту.*
У перші дні свого існування дисплеї використовувалися виняткова для відображення на екрані тексту, що користувач вводив із клавіатури. У графічному інтерфейсі користувача дисплей сам стає джерелом, звідки в машину вводиться інформація. Дисплей показує різні графічні об'єкти у вигляді картинок і конструкцій для введення інформації, таких як кнопки або смуги прокручування для занадто великого для екрана обсягу інформації. Використовуючи клавіатуру або мишу, користувач може безпосередньо маніпулювати цими об'єктами на екрані. Графічні об'єкти можна перетягувати, кнопки можна натискати.

Будь-яка програма для Windows має **вікно** – прямокутну область на екрані. Вікно ідентифікується **заголовком**. Більшість функцій програми запускається за допомогою **меню**. Деякі пункти меню викликають появу **вікон діалогу**, у які користувач вводить додаткову інформацію.

Під Windows будь-яка програма стає **резидентною[2].**

[1] **Додаток** (прикладна програма) – програма, що реалізовує обробку даних у певній сфері застосування.
[2] **Резидентна програма –** завантажується під час роботи компютера в оперативну пам'ять, не вивантажується з неї після виконання, виконується по мірі необхідності й не заважає роботі інших програм.

Одночасно кілька програм Windows можуть бути виведені на екран і виконуватися.

Користувач може переміщати по всьому екрану вікна, які займає кожна програма, змінювати їхній розмір, перемикатися між різними програмами й передавати дані від однієї програми до іншої.

У програм, написаних для Windows, немає прямого доступу до апаратної частини пристроїв відображення інформації.

Замість цього Windows містить у собі мову графічного програмування, яка називається **графічним інтерфейсом пристрою – GDI** (Graphics Device Interface).

Windows абстрагується від конкретного пристрою, і програми, написані для Windows, будуть працювати з будь-яким типом дисплея і будь-яким типом принтера, для яких є в наявності **драйвер**[3] Windows.

1. ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ

1.1. Об'єктно-орієнтовані технології

Розв'язок будь-якої задачі вимагає певного рівня абстракції. Мови програмування високого рівня дають змогу піднятись на більш високий рівень абстракції порівняно з машинними кодами, які є основою мови асемблер. Ця можливість пов'язана з тим, що всі мови програмування високого рівня дозволяють, крім використання стандартних типів даних, створювати свої власні типи даних, які б найбільш повно відбивали характер задачі. Крім того, принцип модульності дозволяє програмну реалізацію задачі формулювати в термінах незалежних модулів, які використовують певну структуру даних задачі.

На відміну від традиційних поглядів, коли програму розглядали як набір підпрограм або як перелік інструкцій комп'ютеру, ООП – програми можна вважати сукупністю об'єктів, кожний з яких здатний отримувати повідомлення, обробляти дані та надсилати повідомлення іншим об'єктам.

Принципи структурного програмування є основою для організації, з одного боку, оптимальної структури програми, а з іншого – для конструювання необхідної структури даних для збереження і обробки інформації в програмі. Проте, дані і модулі, що їх обробляють, співіснують у структурних програмах досить окремо. Точніше кажучи, дані відіграють доволі пасивну роль –

[33] **Драйвер пристрою** – програма, написана спеціально для конкретного периферійного пристрою з метою забезпечити управління цим пристроєм з боку операційної системи.

вони дозволяють модулям їх обробляти, будучи не в змозі захистити себе від некоректного використання.

Сучасна концепція **об'єктно-орієнтованого програмування (ООП** – *Object Oriented Programming***)** визначається як технологія створення складного програмного забезпечення на основі подання програми у вигляді сукупності об'єктів, кожний з яких є екземпляром певного типу (класу), а класи утворюють ієрархію зі спадкуванням властивостей.

Таким чином, кожний об'єкт може містити не лише дані, які характеризують його стан, а і методи їх обробки (функції), які демонструють поведінку об'єкта. Об'єктно-орієнтований підхід заснований на тому, що будь-який елемент оточуючого нас світу можна вважати об'єктом.

Наприклад, на рівні об'єктної абстракції можна розглядати довільні математичні поняття. Скажімо, комплексне число можна визначити як об'єкт, який містить дані – значення дійсної та уявної частин комплексного числа, і визначає методи – допустимі дії над такими об'єктами (обчислення модуля та аргументу, кореня певного степеня з комплексного числа тощо). Довільний технічний прилад також можна вважати об'єктом. Наприклад, телевізор дозволяє сприймати зображення та звук (це інформація, або дані), а також виконувати певні дії – перемикати канали, змінювати гучність, яскравість (ці дії є методами даного об'єкта).

Останній приклад демонструє також захист вмісту об'єкта, адже він використовується як "чорна скринька" – об'єкт лише реагує на зовнішні повідомлення (запити): перемкнути канал, збільшити контрастність, вимкнутись. При цьому користувач навіть не повинен знати, як саме (на технічному рівні) обробляються його запити, і позбавлений права (звісно, якщо не збирається загинути від високої напруги з викруткою в руках) втручатись у вміст об'єкта.

Предметну область будь-якої задачі програмування будемо вважати представленою у вигляді сукупності функціональних елементів (об'єктів), що обмінюються в процесі виконання програми повідомленнями. Процес такого представлення називається **об'єктною декомпозицією**.

При цьому кожний об'єкт предметної області відповідає за виконання деяких дій залежно від одержаних повідомлень та параметрів самого об'єкта.

Сукупність значень параметрів об'єкта визначає його **стан** або **властивості**, а сукупність реакцій на одержані повідомлення **–**

поведінку або **методи** (які задаються як функції об’єкта). Причому **властивості і методи не існують окремо, а об’єднані і утворюють єдиний об’єкт**. Це принципова відміна від процедурного підходу, в якому властивості (тобто дані) існують окремо від процедур, які їх обробляють.

1.2 Основні концепції ООП

Визначимо основні концепції ООП:

- система складається з об'єктів;
- об'єкти взаємодіють між собою;
- кожний об'єкт характеризується станом і поведінкою;
- стан об'єкта задається значенням полів даних;
- поведінка об'єкта задається методами.

Детальніше розглянемо ключові принципи об’єктно-орієнтованого підходу:

1 Що завгодно є об’єктом. Об’єкт можна представляти як своєрідну змінну: він містить дані, але водночас здатний виконувати певні дії над ними. Теоретично будь-який елемент предметної області задачі може бути представлений у програмі як об’єкт.

2 Програма – це комплекс об’єктів, які обмінюються повідомленнями. Щоб попросити об’єкт щось зробити, йому треба надіслати повідомлення. Більш конкретно повідомлення можна представляти як виклик функції, яка є методом деякого об’єкта.

3 Можна створювати нові типи об’єктів, використовуючи вже наявні. Ця можливість реалізується завдяки механізму спадкування.

4 Кожний об’єкт має певний тип. Тип об’єкта визначається повідомленнями, які йому можна надсилати.

5 Різні об’єкти певного типу можуть отримувати однакові повідомлення, реагуючи на них по-різному.

Наприклад, оскільки об’єкт типу “коло” є одночасно об’єктом типу “форма”, успадковуючи його методи, то об’єкт “коло” гарантовано повинен приймати всі повідомлення, призначені для об’єкта “форма”.

1.3 Фундаментальні поняття ООП

Клас

Визначає абстрактні характеристики деякої сутності, включаючи характеристики самої сутності (її атрибути або властивості) та дії, які вона здатна виконувати (її поведінку, методи або можливості).

Клас можна порівняти із кресленням, відповідно до якого створюються об'єкти.

Клас – це тип, що описує будову об'єктів – екземплярів. Класи вносять модульність та структурованість в об'єктно-орієнтовану програму. Властивості та методи класу разом називаються його **членами.**

Клас являє собою об'єктний тип даних, зовні схожий на тип даних «структура» у процедурно-орієнтованій мові QBasic. При цьому елементи такої «структури» (члени класу) можуть самі бути не тільки даними, але й «методами» (тобто процедурами або функціями).

Об'єкт

Це окремий екземпляр класу.

Загальна характеристика об'єкта: об'єкт має стан (сукупність значень атрибутів), поведінку і особистість. При цьому мається на увазі, що об'єкт має внутрішні дані (стан), методи (які визначають поведінку) і кожний об'єкт відрізняється від будь-якого іншого (принаймні тим, що має унікальну адресу в пам'яті).

Об'єкт ООП – це сукупність змінних стану і пов'язаних з ними методів. Ці методи визначають, як об'єкт взаємодіє із зовнішнім світом.

Визначення об'єкта доцільно дати, використовуючи такі підходи:

І. Реалізаційний підхід

Кожний об'єкт в ООП має свій тип (клас), що містить:

- **Поля даних** – параметри об'єкта (звичайно, не всі, а тільки необхідні в програмі), що задають його стан (властивості об'єкта предметної області).
- **Методи** – дії, які можна виконувати над об'єктом такого типу або які сам об'єкт може виконувати (тобто можливості об'єкта).
- **Властивості** – значення, доступні через методи доступу. Вони забезпечують читання й установлення значень полів даних, пов'язаних із властивостями.

Властивості можна розглядати як «розумні» поля даних, які супроводжують доступ до поля даних якими-небудь додатко-

вими діями (наприклад, коли зміна координати об'єкта супроводжується його перемальовуванням на новому місці).

Властивості і методи в ООП називаються **інтерфейсом об'єкта.**

II. Концептуальний підхід

Кожний об'єкт є екземпляром деякого класу об'єктів. Один клас відрізняється від інших ім'ям і, звичайно, набором полів даних, методів і властивостей.

Обмін повідомленнями – передача даних від одного об'єкта іншому, або надсилання *викликів-методів*.

Управління об'єктами (виклик методів) здійснюється не прямо, як це прийнято в процедурних мовах програмування, а через надсилання *об'єктів-повідомлень*.

Взаємодія об'єктів

Об'єкти в програмі не існують самі по собі. Програмування в термінах об'єктів має зміст лише тоді, коли можливо організувати їх взаємодію. Одним із базових принципів ООП є такий: об'єкти можуть взаємодіяти один з одним, надсилаючи повідомлення з проханням виконати деякий належний метод або виконуючи метод у відповідь на запит іншого об'єкта. Взаємодіючи, об'єкти утворюють програму.

Повідомлення – це практично те саме, що і виклик функції у процедурному програмуванні. Об'єкт не може прямо змінити стан іншого об'єкта, він може тільки попросити його виконати деякі дії, надсилаючи повідомлення.

Будь-яке *повідомлення* складається з трьох частин:

1) *ім'я об'єкта*, якому воно адресоване;
2) *ім'я методу*, який повинен виконати об'єкт-адресат;
3) *параметри*, необхідні для виконання методу.

Таким чином, звертання до властивості чи методу об'єкта відбувається з використанням стандартного для об'єктно-орієнтованих мов синтаксису – крапкової нотації. Різні об'єкти можуть мати властивості і методи з однаковими назвами. Саме тому, щоб вказати, до методу якого саме об'єкта відбувається звертання, перед іменем методу вказується ім'я об'єкта, відокремлене крапкою.

Наприклад, (виклик методу «рух об'єкта Тролейбус» з параметром «швидкість»): Тролейбус.рух (швидкість=30 км/год).

Кожен об'єкт – своєрідний незалежний автомат з окремим призначенням та відповідальністю.

В результаті дослідження Дебори Дж. Армстронг (англ. Deborah J. Armstrong) комп'ютерної літератури, що була видана протягом останніх 40 років, вдалось відокремити фундаментальні поняття (принципи), використані у переважній більшості визначень ООП.

1.4 Принципи ООП

Визначимо основні принципи ООП:
I. Абстракція даних.
II. Інкапсуляція.
III. Ієрархія.
IV. Поліморфізм.

I. Абстракція даних

Абстракція даних – виділення істотних характеристик деякого об'єкта, що відрізняють його від усіх інших видів об'єктів.

Об'єкти являють собою спрощений, ідеалізований опис реальних сутностей предметної області.

Вибір правильного набору абстракцій для заданої предметної області являє собою *головне завдання ООП*.

Мабуть, Аристотель був першим з тих, хто почав вивчати ідею "типу". Він говорив про "клас риб" і "клас птахів".

Ідея про об'єкти, які, хоча і є унікальними, належать до певного класу і мають спільні характеристики і поведінку, використовується в усіх об'єктно-орієнтованих мовах. Фундаментальним поняттям ООП є **абстрактні типи даних** (або класи).

За визначенням, ***абстрактний тип даних*** – це група пов'язаних між собою даних і методів (функцій), які можуть здійснювати операції над цими даними.

Абстрактні типи даних дуже схожі на вбудовані типи: можна створювати змінні відповідного типу (які в ООП називаються об'єктами, а, наприклад, в мові VB – екземплярами класу) і виконувати якісь дії з цими змінними (наприклад, надсилати повідомлення, при цьому об'єкт одержує повідомлення і сам вирішує, що з ним робити).

Відмінність ООП від традиційного підходу полягає у тому, що тут програміст може визначати клас (тип даних), який є зручним для розв'язання конкретної задачі, а не примушений використовувати для цього лише існуючі типи даних. Можна в

рамках ООП розширювати мову програмування шляхом створення нових типів даних, необхідних для розв'язання конкретної задачі. Програмна система надає цим новим типам, або класам, таку саму підтримку, яка існує для вбудованих типів.

Тип об'єкта – це сукупність методів, які підтримує об'єкт, тобто інтерфейс. Інтерфейс визначає те, що може робити даний об'єкт. Проте десь повинен існувати код, в якому запрограмовані відповідні дії. Цей код, разом з прихованими даними об'єкта, складає його реалізацію.

Це просто зрозуміти по аналогії з процедурним програмуванням. Для кожного типу визначені функції, які реалізовують усі можливі дії із змінними цього типу. В разі потреби виконати певну дію викликається відповідна функція. Іншими словами, в термінах ООП: об'єкту надсилається запит (повідомлення), а об'єкт вирішує, як його обробити (виконує код).

II. Інкапсуляція (приховування інформації):

- відділення одне від одного елементів об'єкта, що визначають його призначення і поведінку;
- приховування деталей про роботу класів від об'єктів, що їх використовують або надсилають їм повідомлення.

Інкапсуляція ізолює інтерфейс абстракції від її реалізації. Відповідно до цього принципу, будь-який клас повинен розглядатися як чорний ящик – користувач класу повинен бачити й використовувати тільки інтерфейсну частину класу (тобто список декларованих властивостей і методів класу) і не вникати в його внутрішню реалізацію. Тому дані прийнято інкапсулювати у класі таким чином, щоб доступ до них здійснювався не прямо, а за допомогою методів.

Інкапсуляція – це механізм, який об'єднує дані і методи, що маніпулюють цими даними, і захищає і те, і інше від зовнішнього втручання або неправильного використання. Коли методи і дані об'єднуються в такий спосіб, утворюється об'єкт.

Інкапсуляція досягається шляхом вказування, які класи можуть звертатися до членів об'єкта. Здійснюється об'єднання даних і процедур їхньої обробки в рамках однієї синтаксичної структури мови програмування.

Подібно до того, як у процедурному програмуванні глобальна функція не має доступу до локальних змінних іншої функції, так і дані всередині об'єкта приховані від інших частин програми. Для маніпулювання даними об'єкта інші частини програми повинні

попросити об’єкт викликати його власні інкапсульовані методи, які і повертають інформацію про стан об’єкта (або дані) модулю, що послав запит. Реалізація класу об’єкта може бути змінена непомітно для решти програми, якщо незмінним залишається інтерфейс класу.

Часто члени класу позначаються як публічні (англ. **public**), захищені (англ. **protected**) та приватні (англ. **private**), визначаючи, чи доступні вони всім класам, підкласам або лише до класу, в якому їх визначено.

ІІІ. Ієрархія

Ієрархія – впорядкування абстракцій, розташування їх по рівнях.

Види ***ієрархій*** – **агрегація, спадкування.**

- Новий клас може бути створений з довільної кількості об’єктів інших класів, у будь-якому їх поєднанні.

Після того, як клас створений і відтестований, він може бути використаний неодноразово. Найпростішим способом повторно використати клас є створення об’єкта (екземпляра) даного класу. Але і сам клас може стати цеглиною при побудові нового класу. Останній може бути створений з довільної кількості об’єктів інших класів, у будь-якому їх поєднанні. Цей процес називається **агрегацією** (або композицією).

- Інший спосіб утворення нового класу – це механізм **спадкування**.

Спадкування – це процес, шляхом якого деякий клас може успадковувати властивості та методи деякого існуючого класу, додаючи до них деякі особисті риси або здатність об'єкта зберігати атрибути класу або батьківського об’єкта.

Існує можливість породжувати один клас від іншого зі збереженням усіх властивостей і методів класу-предка і додаючи, при необхідності, нові властивості й методи.

Спадкування використовується у випадку, коли треба створити клас з функціональністю, схожою на функціональність вже існуючого класу. При успадкуванні утворюється клон існуючого (батьківського, базового) класу, який є коренем ієрархії успадкування, і цей клон (нащадок) відповідним чином модифікується. Ці два класи можуть мати спільні характеристики і поведінку, але один з них (нащадок) може мати більше властивостей і обробляти більше повідомлень (або обробляти їх інакше).

В одного батьківського класу може бути декілька нащадків, при цьому батьківський (базовий) клас містить усі властивості і методи, спільні для його нащадків.

Набір класів, зв'язаних відношенням спадкування, називають **ієрархією**.

Приклад

Форма (базовий клас)

Властивості: колір, розмір, розташування,

…

Методи: обчислити площу, намалювати, знищити, зсунути,

…

Трикутник (нащадок)

Властивості:…

Методи: віддзеркалити (додається), обчислити площу (змінюється),

…

Квадрат (нащадок)

Властивості:…

Методи: обчислити площу (змінюється),

…

Круг (нащадок)

Властивості:…

Методи: обчислити площу (змінюється),

…

Клас-нащадок успадковує інтерфейс базового класу. Це означає, що всі повідомлення, які може обробляти базовий клас, може обробляти і клас-нащадок. Оскільки клас розпізнається за повідомленнями, які він може обробляти, то клас-нащадок має той самий тип, що і базовий клас. "Круг – це форма". Це називається еквівалентністю типів через успадкування.

IV. Поліморфізм

Поліморфізм – здатність об'єкта набувати різних форм.

Можна сказати, що ***поліморфізм*** – це властивість різних об'єктів по-різному відповідати на однакові повідомлення, тобто одне й те саме ім'я може використовуватись для методів різних класів.

У попередньому прикладі метод "обчислити площу" міститься в різних нащадках: Трикутник, Квадрат, Круг і в самому базовому класі Форма, але в кожному з них діє по-

різному. Надсилаючи одне й те саме повідомлення "обчислити площу" об'єктам всіх цих різних класів, одержимо різні (правильні для відповідного класу) результати.

Похідний об'єкт успадковує методи й властивості об'єкта-батька. ***Поліморфізм*** дозволяє додавати видозмінювати і навіть видаляти деякі особливості поведінки похідного об'єкта.

Поліморфізм *забезпечується тим, що в класі-нащадку змінюють реалізацію методу класу-предка з обов'язковим збереженням* ***сигнатури***[4] *методу.*

Поліморфізм означає залежність поведінки від класу, в якому ця поведінка викликається, тобто, два або більше класів можуть реагувати по-різному на однакові повідомлення.

Основна кількість властивостей і методів об'єктів притаманні всім екземплярам класу, але деякі можуть бути змінені при необхідності, щоб характеризувати тільки даний екземпляр.

Стосовно програмування ***поліморфізм*** – це явище, при якому той самий програмний код виконується по-різному, залежно від того, об'єкт якого класу використовується при виклику даного коду.

Основною концепцією ***поліморфізму*** є ідея про те, що один інтерфейс дозволяє реалізувати багато методів.

При цьому вибір конкретного методу, залежно від ситуації, здійснює компілятор. Механізм цього вибору такий: під час виклику певного методу класу спочатку шукається такий метод у самому класі. Якщо його знайдено – він виконується, якщо ж ні – відбувається звертання до класу, який є базовим (батьківським) для заданого класу, і пошук методу відбувається в ньому. Цей процес повторюється до тих пір, поки не знайдено корінь (верхній клас) ієрархії успадкування.

Це забезпечує збереження незмінним інтерфейсу класу-предка й дозволяє здійснити зв'язування імені методу в коді з різними класами – з об'єкта якого класу здійснюється виклик, з того класу й береться метод з даним ім'ям.

Такий механізм називається ***динамічним*** (або пізнім) ***зв'язуванням*** – на відміну від ***статичного*** (раннього) ***зв'язування***, здійснюваного на етапі компіляції програми.

---

[4] **Сигнатура** методу – це скорочена форма запису параметрів методу й типів значення, що повертається. Варто підкреслити, що в сигнатуру не входять ні ім'я методу, ні імена параметрів.

При читанні даного розділу варто мати на увазі, що зараз ООП є основою всієї індустрії прикладного програмування завдяки виграшу в конкурентній боротьбі з альтернативними технологіями програмування. У промисловому програмуванні тільки в системному програмуванні позиції ООП ще не дуже сильні. Тому, з одного боку, теоретичні міркування про непридатність ООП не відповідають спостережуваній на практиці ситуації. З іншого боку,справедливості заради не можна вважати, що ООП у всіх випадках є найкращою з методик програмування.

Процедурне програмування краще підходить для випадків, коли важлива швидкодія й споживані ресурси, об'єктне – коли важлива керованість проекту і його модифікованість, а також безпека програм. Процедурне програмування звичайно краще підходить для невеликих проектів, об'єктне – для великих.

Дослідження Thomas E. Potok, Mladen Vouk і Andy Rindos показало відсутність значимої різниці в продуктивності розроблення програмного забезпечення між ООП і процедурним підходом.

Кристофер Дейт указує на неможливість порівняння ООП і інших технологій багато в чому через відсутність строгого й загальновизнаного визначення ООП .

Олександр Степанов в одному зі своїх інтерв'ю указував на те, що ООП «методологічно неправильно» і що «... ООП практично така сама містифікація, як і штучний інтелект...».

Фредерик Брукс указав на те, що найбільш складною частиною створення програмного забезпечення є: «… специфікація, дизайн і тестування концептуальних конструкцій, і аж ніяк не робота з вираження цих концептуальних конструкцій…»... ООП (поряд з такими технологіями, як штучний інтелект, верифікація програм, автоматичне програмування, графічне програмування , експертні системи й ін), на його думку, не є «срібною кулею», що може знизити складність розроблення програмних систем. На його думку, «...ООП дозволяє скоротити тільки привнесену складність у виразу дизайну. Дизайн залишається складним по своїй природі...».

Едсгер Дейкстра указував: «... те, про що суспільство в більшості випадків просить, – це змііне масло. Природно, "змііне

масло" має дуже вражаючі імена, інакше буде дуже важко щось продати: "структурний аналіз і дизайн", "програмна інженерія", "моделі зрілості", "управляючі інформаційні системи", "інтегровані середовища підтримки проектів", "об'єктна орієнтованість", "реінжиніринг бізнес-процесів"...»

Патрік Кіллеліа у своїй книзі "Тюнінг веб-сервера" писав: «...ООП надає вам безліч способів сповільнити роботу ваших програм...».

Багато сучасних мов спеціально створені для полегшення ООП. Однак слід зазначити, що можна застосовувати техніки ООП і для необ'єктно-орієнтованої мови й, навпаки, застосування об'єктно-орієнтованої мови зовсім не означає, що код автоматично стає об'єктно-орієнтованим і т.д.

1.5 Переваги та недоліки ООП

Переваги ООП

- можливість модифікації коду окремих частин програми незалежно від інших;
- підвищення надійності програм;
- можливість повторного використання коду компонентів програми;
- більш легке розуміння розроблювачами процесів;
- більш простий супровід візуальних класів (пошук помилок і доопрацювання), в результаті чого підвищується продуктивність праці програмістів і знижуються працевитрати.

Недоліки ООП:

- вимагає додаткових витрат пам'яті;
- призводить до зниження швидкості виконання додатків.

Класична схема навчання програмуванню раніше мала такий вигляд: спочатку теоретичне вивчення синтаксису деякої мови, а вже потім виконння практичних завдань.
Для написання перших програм для Windows взагалі не потрібно знати про особливості мови – потрібно розуміти загальну логіку розроблення додатка і вміти працювати в середовищі та використовувати потрібний інструмент. І тільки після цього можна переходити до вивчення мови для "вишуканого програмування".

Одним із засобів розробки вищезгаданих Windows-програм є Visual Basic (VB), застосування якого:

- *по-перше, дозволяє у максимальному ступені абстрагуватися від мовних проблем;*
- *по-друге, знайомить із середовищем програмування, характерним для усіх засобів розроблення Microsoft.*

Освоєння ж мови реально можливе тільки в ході практичної роботи й самонавчання. В 1991 році під гаслом "тепер і програмісти-початківці можуть легко створювати додатки для Windows" з'явилася перша версія нового інструментального засобу Microsoft Visual Basic.

У той момент Microsoft досить скромно оцінювала можливості цієї системи, орієнтуючи її, насамперед, на категорію початківців і непрофесійних програмістів. Основне завдання тоді полягало в тому, щоб випустити на ринок простий і зручний інструмент розробки в новому середовищі Windows, програмування в якому було проблемним навіть для досвідчених фахівців. Тому VB версії 1.0 був схожий скоріше на діючий макет майбутнього середовища розробки, ніж на робочий інструмент.

Однак уже тоді принципове нововведення VB полягало в реалізації ідей ***подійно-керованого й візуального програмування*** *в середовищі Window, які радикально відрізнялися від класичних схем розробки програм.*

VB став родоначальником нового покоління інструментів, які сьогодні називаються **засобами швидкого розроблення програм** – **RAD** (Rapid Application Development).

Зараз ця ідеологія вже звична, але тоді вона здавалася зовсім новою, і це створювало серйозні проблеми (у тому числі чисто психологічного плану) для програмістів "старих часів".

Проте кількість VB-користувачів зростала, причому багато в чому за рахунок величезної популярності її попередника – QuickBasic. При цьому VB швидко "мужнів" як у результаті розвитку середовища програмування, так і за рахунок включення в нього професійних елементів мови і проблемно-орієнтованих засобів.

На початку 90-х років намітилася тенденція включати в додатки засоби внутрішнього програмування, які повинні

були вирішувати завдання настроювання і адаптації цих пакетів для конкретних умов їхнього застосування.
*Наприкінці 1993 року Microsoft оголосила про намір створити на основі VB нову універсальну систему програмування для прикладних програм, що одержала назву Visual Basic for Applications (VB для додатків), або **VBA**. Реалізацію цього проекту вона почала із власних офісних пакетів.*
До складу MS Office 2000 увійшла версія VBA 6.0. Тепер вона використовується вже в шести програмах – Word, Excel, PowerPoint, Access, Outlook, Frontpage. Тому в останні три роки Microsoft представляє свій пакет MS Office не просто як набір прикладних програм, а як комплексну платформу для створення бізнес-додатків, для розв'язування широкого кола спеціалізованих задач користувачів.

Основна відмінність мови програмування VB і мови Visual Basic for Application в тому, що мовою Visual Basic створюються самостійні додатки, а проекти VBA працюють у середовищі Microsoft Office.

Крім того, Microsoft оголосила про можливості ліцензування VBA для того, щоб зробити це середовище фактичним стандартом для управління додатками.
Написані на VB програми досить близькі за стилем до традиційного програмування ранньої епохи Basic (якщо не зважати на деякі дивні, але необхідні синтаксичні конструкції).
*При програмуванні для Windows ви фактично займаєтесь одним з видів **об'єктно-орієнтованого програмування.***

Об'єктно-орієнтоване програмування – одна з **парадигм**[5] програмування, яка розглядає програму як множину елементів, що взаємодіють між собою, і у якій основними концепціями є поняття «**об'єктів**» і «**класів**».

Незважаючи на те, що ця парадигма з'явилась в 1960-х роках, вона не мала широкого застосування до 1990-х.
ООП сягає своїм корінням до створення мови програмування Симула в 1960-х роках, одночасно з посиленням дискусій про кризу програмного забезпечення. Разом із тим, як ускладнювалось апаратне та програмне забезпечення, було дуже важко зберегти якість програм.

[5] **Парадигма** (від гр. Paradeigma-приклад, зразок) – вихідна концептуальна схема, модель постановки проблем і їхнього вирішення, методів дослідження, що панують протягом певного історичного періоду в науковому співтоваристві.

На сьогодні багато із мов програмування (зокрема, Java, ActionScript3, C#, C++, Python, PHP, Ruby та Objective-C) підтримують ООП.

2 СЕРЕДОВИЩЕ VB 6.0

Можливості VB нічим не обмежені. Ви можете розширювати їх за допомогою використання додаткових функцій, наприклад, ***бібліотек динамічного компонування*** *(**DLL-**бібліотек), які можуть бути написані будь-якою мовою програмування.*

На VB можна написати різноманітні програми: від обслуговуючих рутинні операції введення даних до складних інформаційних і комунікаційних систем. У США 60% програмних продуктів написані на VB. Є звичайно дуже невеликі обмеження, наприклад, в VB не можна використовувати асемблер.

Технологія роботи в середовищі VB базується на описаних вище ідеях об'єктно-орієнтованого та візуального програмування.

Ідея ООП втілена у VB в об'єднанні (інкапсуляції) даних і засобів їх опрацювання (методів) в об'єкт.

Середовище візуального програмування VB – це графічна автоматизована оболонка над об'єктно-орієнтованою версією мови **Basic**.

Об'єкт у VB – комбінація програмного коду і даних, яку можна вважати одним цілим. Об'єктом є частина додатка: як форма або елемент управління у вікні: кнопки, списки, текстові поля тощо. Цілий додаток теж є об'єктом**.**

Властивістю об'єкта у VB є якісна або кількісна його характеристика: розміри, положення на екранній формі, кольори самого об'єкта, кольори тексту, поміщеного на об'єкт, характеристики шрифту і т.д.

Значення цих властивостей, наприклад, розмір тексту – кількісна характеристика; видимість об'єкта – якісна характеристика.

Методи у VB – це програмні процедури (або фрагменти коду), які виконують деяку обробку, пов'язану з об'єктом.

Примітка – *Клацнувши по командній кнопці, виконуються певні розрахунки – для цього фрагмент програмного коду необхідно додати до тіла процедури командної кнопки*.

Інтерфейс – це сукупність засобів, що забезпечують фізичну або логічну взаємодію пристроїв і програм обчислювальної системи або користувача з комп'ютером.

Якщо у мові Basic структурними одиницями є дані та команди, то у VB це візуальний об’єкт, який називається **компонентом**.

Автоматизація програмування досягається завдяки можливості переносити компонент на форму (у програму) з панелі компонентів і змінювати його властивості, не вносячи вручну змін до програмного коду.

Проектом називають сукупність файлів, що входять у додаток і зберігають інформацію про його компоненти, й з яких VB створює готову для виконання програму.

Процес створення проекта (і при бажанні, Windows-додатка) **складається з таких етапів**:

1 Створення екранної форми (з об'єктами і їхніми властивостями).

2 Написання програми (програмного коду).

3 Відлагодження програми, тобто усунення в ній логічних помилок.

4 Перетворення проекту у Windows-додаток.

2.1 Структура VB-проекту

У VB будь-який проект складається з однієї або декількох форм, кожна з яких вирішує певну задачу й містить такі файли:

- файл **екранної форми** (розширення **frm**) – текстовий ASCII-файл, який містить графічне зображення вікна додатка разом з його змістом. Він включає:
  - перелік властивостей цього вікна з їхніми значеннями;
  - перелік об'єктів, що містяться у цьому вікні;
  - властивості цих об'єктів з їхніми значеннями;
- бінарний файл форми або **ресурсів програми** (**frx**) – по одному для кожної форми – містить рисунки, які розташовані на формі (наприклад, картинка в **PictureBox**);

- файл **проекту** (**vbp**), який містить посилання на всі файли проекту (скільки форм містить проект і в якій папці вони розміщені) та ініціалізує програму;
- файл **параметрів проекту** (**vbw**) з інформацією про робочу область проекту (workspace).

Примітка *– Це необхідний мінімум. Хоча, бувають і винятки, наприклад, коли в проекті не використовуються форми. Тоді замість* ***frm****-файлу, буде* ***bas****-файл.*

Далі перелічені **необов'язкові додаткові файли**, які можуть бути підключені до проекту:

- текстовий файл кожного **програмного модуля** (**bas**) (окремо збережених текстів підпрограм, не відповідальних за візуальну складову додатка);
- текстовий файл кожного **модуля класів** (**cls**);
- текстовий файл кожного **додаткового елемента управління** (**ctl**);
- один файл **ресурсів** (**res**);
- файли **елементів управління стандарту ActiveX (ocx)**;
- інші файли (**tlb** і т.д.).

Примітка *– Запам'ятовувати призначення всіх цих файлів не обов'язково, досить запам'ятати два файли:* **frm***-файл, у якому зберігаються код форми й властивості всіх поміщених на дану форму елементів управління,* і **bas***-файл – модуль, в якому можуть бути оголошені глобальні змінні, константи, функції й т.д. (код без елементів управління).*

Рисунок 2.1 – Вікно нового проекту

При першому запуску ***VB*** *запускається майстер* ***Project Wizard*** *і на екрані з'являється діалогове вікно* ***New Project*** *(Новий проект) (рисунок 2.1)*

У цьому вікні можна вибрати один з декількох типів шаблонів проектів, що полегшують початок роботи над додатком. Вікно складається із трьох вкладок: ***New*** *(Нові*

*проекти), **Existing** (Існуючі проекти) і **Recent** (проекти, що використалися недавно).*
*Вибираючи **New**, ви доручаєте **VB** створити основу додатка. Це заощаджує чимало часу. Згодом ви навчитеся користуватися багатьма із поданих тут шаблонів.*
*На вкладці **Existing** можна вибрати існуючий проект. Це може бути проект прикладу, що входить у комплект **VB**, або ж проект, над яким ви працювали в минулому. Чим більше ви будете працювати з **VB**, тим вам частіше доведеться звертатися до цієї вкладки.*
*Вкладка **Recent** дозволяє вибрати з проектів, що недавно використовувалися. Зовні вона схожа на **Existing**, але в ній перелічуються лише ті проекти, над якими ви працювали останнім часом , а не всі існуючі.*
*Якщо ви хочете обходитися без вибору типу проекту – встановіть внизу прапорець **Don't show this dialog in the future**, і при наступному запуску це вікно не з'явиться.*
*Якщо двічі клацнути мишею піктограму **Standart.exe** та обрати вкладку **New** – з'явиться **Головна панель (вікно) проекту** або **Інтегроване середовище розробки** – **IDE** (Integrated Development Environment) – важлива складова частина VB. Його вигляд показаний на рисунку 2.2. Саме тут ви збираєте воєдино додаток і проводите основний час у роботі над ним.*

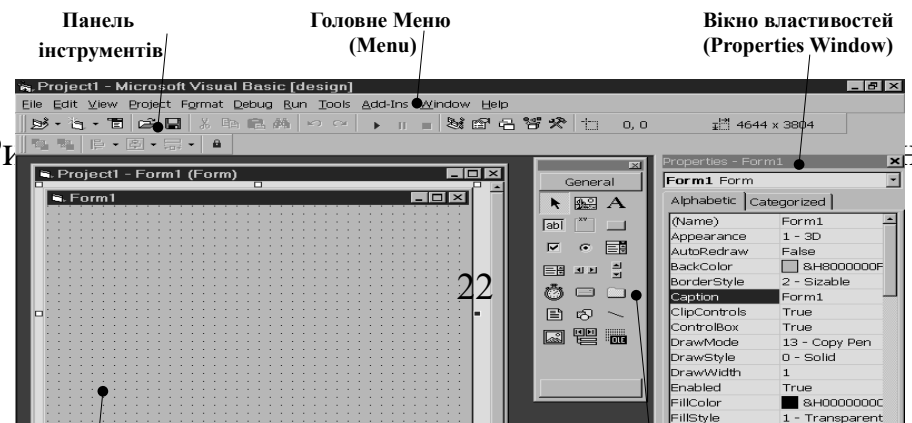


Рис. 2.2. Головна панель (вікно) проекту VB

2.2. Інструменти середовища VB

Вікно середовища містить:

- головне меню (**Menu**);
- стандартну (**Standard)** панель інструментів редактора (**ToolBar**);
- вікно елементів управління (**ToolBox**);
- вікно властивостей об'єктів (**Properties Window**);
- вікно конструктора форм (**Form Designer**), яке може містити – вікно форми (**Object**) або вікно редактора коду(**Code**);
- вікно (макет) розміщення форм (**Form Layout**);
- вікно провідника проектів (**Project Explorer**);
- вікно перегляду характеристик (класів) об'єктів (**Object Browser**);
- вікно редактора меню (**Menu Editor**).

Крім перелічених у середовищі VB, є інші вікна:

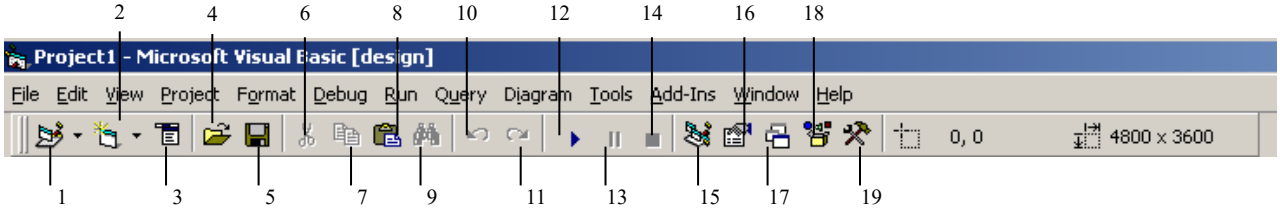
- вікно редагування форми (**Form Editor)**;
- вікно редагування коду (**Edit)**;
- вікно відладчика програм (**Debug)**

та вікна, що з'являються під час запуску та відлагодження програми:

- вікно безпосереднього виконання (**Immediate)**;
- вікно стеження (**Watch)**;
- вікно локальних змінних (**Locals).**

2.2.1 Головне меню

Розглянемо три верхні рядки (рисунок 2.3).



1 – Add Standart EXE Project; 2 – Add Form; 3 – Menu Editor; 4 – Open Project; 5 – Save Project; 6 – Cut; 7– Copy; 8 – Paste ; 9 – Find; 10– Undo; 11 – Redo; 12 – Start;13 – Break;14 – End; 15 – Project Explorer; 16 – Properties Window; 17 – Form Layout Window ; 18 – Object Browser; 19 – Toolbox

Рисунок 2.3 – Вікно головного меню з панеллю інструментів редактора VB

Перший з них – **рядок заголовка**.

У центральній частині рядка розташовується ***ім'я вікна****. Воно складається з* ***імені проекту*** *(**Project1**), після якого зазначене* ***програмне середовище*** *(**Microsoft Visual Basic**). Далі,* виразом [**design]** зазначений поточний режим додатка – **проектування**. У режимі **виконання** проекту текст у квадратних дужках заміняється на **[run**].

Під рядком заголовка розташовується ***рядок головного меню****. Головне меню, як і у всіх додатках Microsoft, являє собою лінійку.*

Багато команд виявляться знайомими, оскільки головне меню VB організоване й працює так само, як і в інших додатках Microsoft, – наприклад, у текстовому редакторі Microsoft Word або в електронній таблиці Microsoft Excel.

Головне меню основного вікна містить стандартне для Windows меню:

- **File (*Файл*)** – команди для відкриття, збереження, друку й компіляції проекту VB;
- **Edit (*Редагування***) – команди редагування;
- **View (*Вид*)** – команди перегляду компонентів VB і меню, властиве VB;
- **Project** (***Проект***) – команди додавання форм, програмних модулів й т.д;
- **Format** (***Формат***) – команди для розташування й встановлення розмірів елементів і форм;
- **Debug** (***Відлагодження***) – команди запуску й зупинки додатків, виконання інших операції, що допомагають стежити за роботою програми;
- **Run (*Виконати*)** – команди для виконання й компіляції проекту;
- **Query** ***(Запит)***– команди для створення й виконання запитів до бази даних;
- **Diagram** ***(Діаграма)*** – команди для побудови діаграми, що відображає структуру бази даних;
- **Tools (*Засоби*)** – команди для конфігурування середовища програмування VB;
- **Add-in (*Доповнення***) – додаткові засоби для розширення можливостей VB;
- **Window (*Вікно*);**
- **Help (*Допомога*)** – доступ до довідкового керівництва.

Розглянемо **основні команди головного меню**:

- *У меню **File** згруповані команди для роботи з файлами проекту: створення нового проекту, відкриття проекту для внесення змін, додавання проекту для паралельної роботи над декількома проектами або копіювання форм із проекту в проект тощо (таблиця А.1).*

- *Меню **Edit** містить команди, призначені для редагування (таблиця А.2). Частина з них відповідає командам редагування інших додатків Microsoft, наприклад, скасувати попередню команду, повторити попередню команду, вирізати фрагмент тексту, скопіювати фрагмент у буфер обміну. У складі цього меню цілий набір команд для роботи з таблицями бази даних: додавати й видаляти поля, призначати первинний ключ у таблиці бази даних.*

- *У складі меню **View** містяться команди виклику та розкриття вікон і панелей **інструментального середовища** VB (таблиця А.3).*

- *У меню **Project** згруповані команди управління проектом і його елементами (таблиця А.4). Вони дозволяють додати в проект і видалити з нього такі елементи, як форма, програмний модуль, клас і інші. За допомогою команд даного меню ви також можете відкрити вікно властивостей додатка, додати посилання на бібліотеки, що підключаються, додати додаткові компоненти на панель елементів.*

- *Меню **Format** містить команди форматування об'єктів у формі, які дозволяють змінювати їх розміри, вирівнювати розміри виділеної групи об’єктів і т.д. (таблиця А.5).*

- *У меню **Debug** згруповані команди (таблиця А.6), призначені для відлагодження додатків. Використовуючи команди цього меню, можна встановити точки останова програми й перевірити значення виразів й змінних, виконуючи програму по кроках.*

- *Команди меню **Run** використовуються для управління запуском додатка (таблиця А.7). Використовуючи команди даного меню, можна запустити додаток на виконання з компіляцією або без компіляції, призупинити або перервати виконання додатка.*

▪ *Меню **Query** містить команди для створення й виконання запитів до бази даних (таблиця А.8). За допомогою цього меню, наприклад, можна виконати запит, перетворити результати запиту в табличний вигляд.*

▪ *У меню **Diagram** згруповані команди для побудови діаграми, що відображає структуру бази даних (таблиця А.9).*

▪ *Команди меню **Tools** управляють настроюваннями (таблиця А.10) інструментарію середовища VB. Вони дозволяють додати процедуру й визначати її атрибути, викликати вікно редактора меню, змінити настроювання програми в діалоговому вікні **Options** (Параметри), а також управляти сховищем проектів **SourseSafe**.*

▪ *У меню **Add-Ins** містяться команди (таблиця А.11) виклику додаткових утиліт, що називаються надбудовами.*

▪ *Меню **Window** містить стандартні команди управління вікнами (таблиця А.12). При виклику вікон додатка меню **Window** поповнюється відповідними командами активізації вікон додатка. При цьому команди активізації вікон називаються іменами відкритих вікон.*

▪ *У меню **Help** згруповані команди виклику довідкової системи VB (таблиця А.13). За допомогою цього меню можна звернутися до довідкової системи з різними варіантами подання інформації. Відмінність між ними складається у вкладці, що відкривається за умовчанням.*

Стандартна лінійка панелі інструментів редактора розташована під головним меню.

Якщо ця панель відсутня в головному вікні програми, для її відображення в меню **View** виберіть команду **Toolbars,** а потім значення **Standard**.

*У її лівій частині є подвійний вертикальний рельєф, схопивши за який ми можемо "вирвати" цю панель із її стандартного місця розташування й розмістити в будь-якому іншому місці екрана. Ця панель після відокремлення одержує власний рядок заголовка з ім'ям **Standard** і кнопкою закриття. Подвійним клацанням по цьому рядку можна повернути панель на її звичайне місце.*

Можна перетворити лінійку панелі інструментів у вікно панелі інструментів(двічі клацнути мишею по правій частині лінійки) та навпаки (двічі клацнути мишею заголовок вікна).

Лінійка містить п'ять груп кнопок-піктограм команд, що викликаються найчастіше та дублюють найбільш часто використовувані команди, доступні через ті або інші пункти головного меню.

Вони показані на рисунку 2.3 і їхнє призначення описане в таблиці 2.1 (19 кнопок-піктограм).

Якщо ви забули назву та призначення кнопки, встановіть на ній покажчик миші – під кнопкою з'явиться маленьке віконце (віконце покажчика), в якому буде необхідна назва.

Таблиця 2.1 – Інструменти редактора

Кнопка	Назва	Призначення
1	2	3
	**Add Standard EXE Project** (Додати стандартний проект)	Додає (створює) стандартний exe-проект
	**Add Form** (Додати форму)	Додає (створює) форму в проект
	**Menu Editor** (Редактор меню)	Викликає редактор меню

Продовження таблиці 2.1

1	2	3
	**Open Project** (Відкрити проект)	Відкриває проект
	**Save Project** (Зберегти проект)	Зберігає проект
	**Cut** (Вирізати)	Вирізує об'єкт у буфер обміну
	**Copy** (Копіювати)	Копіює об'єкт у буфер обміну
	**Paste** (Вставити)	Вставляє об'єкт з буфера обміну
	**Find** (Знайти)	Здійснює пошук інформації з контексту (заданому імені й ознакам)
	**Can't Undo** (Скасування попереднього)	Скасовує попередню дію
	**Can't Redo** (Скасування повторного)	Відновлює скасовану дію
	**Start** (Запустити)	Запускає програму на виконання
	**End** (Закінчити)	Припиняє виконання програми
	**Break** (Перервати)	Перериває виконання програми

	**Project Explorer** (Провідник проекту)	Відкриває вікно провідника проектів
	**Properties Window** (Вікно властивостей)	Відкриває вікно властивостей
	**Form Layout Window** (Вікно макета форм)	Відкриває вікно макета форми
	**Object Browser** (Браузер об'єктів)	Відкриває вікно браузера об'єктів
	**Toolbox** (Панель елементів управління)	Відкриває панель елементів управління
	**Data View Window** (Вікно перегляду даних)	Відкриває вікно перегляду даних
	**Visual Component Manager** (Менеджер візуальних компонентів)	Відкриває вікно управління візуальними компонентами Visual Component Manager

*Кнопки **створення форми, створення, відкриття та збереження проекту** – для створення нових або відкриття існуючих форм або проектів, введення нових діалогових вікон (форм) і збереження проекту на диску із внесеними змінами.*
*Кнопка **редактора меню** – для створення користувальницького меню проекту й визначення його властивостей.*
*Кнопка **властивостей** – для виведення вікна зі списком властивостей і їхніх значень для форми й елементів управління.*
*Кнопка **перегляду об'єктів** – виводить вікно, у якому можна для обраного проекту одержати список його компонентів або модулів і стосовних до кожного з них список методів і властивостей. Для обраних елементів списку виводиться коротка анотація й можна одержати розгорнуту довідку.*
*Кнопки "**Старт", "Перервати виконання", "Кінець виконання"** використовуються при відлагодженні програм.*
Праворуч від першої кнопки  розташована стрілка, направлена униз; клацнувши її, можна побачити перелік видів проектів, які можна додатково створити, не закриваючи поточний проект.
Праворуч від другої кнопки  розташована стрілка, направлена униз; клацнувши її, можна побачити перелік типів файлів, які можна додавати в поточний проект (це можуть бути форми, програмні модулі і т.ін.).

У правій частині цієї панелі розташовується група із двох координатних пар: **положення** (щодо лівої та верхньої внутрішніх

границь головного вікна в режимі виконання) і **розмірів поточного (виділеного) об'єкта**, виражені у **твіпсах (twips)** – умовних одиницях, рівних **1/1440 дюйма** на принтерній роздруківці.

Всі ці параметри встановлюються при проектуванні в процесі компонування форми майбутнього додатка й можуть програмно змінюватися в процесі його роботи.

Ця група наявна тільки при максимізованому статусі головного вікна. Коли проект тільки що відкрився, таким об'єктом є розташована у вікні документа екранна форма майбутнього додатка з ім'ям ***Form1****.*

*Координатні пари (0, 0 і 4644, 3804) відображають значення таких властивостей виділеного об'єкта (****Form1****):* ***Left*** *(Лівий край),* ***Top*** *(Правий край)* ***Width*** *(Ширина),* ***Height*** *(Висота) відповідно.*

Положення на екрані вікна проектованого додатка під час його виконання встановлюється в спеціальному вікні ***Form Layout*** *(Зовнішній вигляд форми), про яке піде мова нижче.*

*До стандартної лінійки панелі інструментів, що входить до складу вихідної конфігурації головної панелі проекту, можна додати додаткові (спеціальні) набори інструментів, про які ми вже згадували, говорячи про вікно середовища:****Form Editor*** *(Редактор форми) – вирівнювання, зміна розмірів та позицій об'єктів, що розташовані на формі;****Edit*** *(Редактор коду);****Debug*** *(Відладчик).*

Примітка – *На початку роботи з* ***VB*** *не приєднуйте нічого додаткового.*

Перейдемо до розгляду інструментальних вікон, які розташовані на робочій поверхні головного вікна. Ці вікна надають нам ті або інші інструментальні засоби, використовувані при проектуванні додатків.

2.2.2 Інструментальні вікна

Позитивна якість графічного інтерфейсу користувача – наявність стандартного набору об'єктів діалогу (вікна, кнопки, лінійки прокручування й т.д.) для створення стандартного Windows-інтерфейсу проекту.

*Вікна пристиковані одне до одного (****Docking****). Щоб відстикувати вікно, виконайте подвійне клацання на рядку заголовка або просто перемістіть його. Щоб пристикувати вікно, перемістіть його на границю й*

відпустіть кнопку миші. Можна переміщати границі між пристикованими вікнами й змінювати їхні розміри. Кожне з вікон можна забрати з екрана й у потрібний момент повернути їх назад за допомогою команд меню ***View*** *або кнопок на панелі інструментів.*

2.2.2.1 Конструктор форм

В центрі екрана розташовано ***вікно конструктора форм (Form Designer).*** *Саме в цю область виводиться зображення форми або вікно (редактора) програмного коду проекту.*

Вікно конструктора форм є основним робочим вікном, у якому виконується візуальне проектування додатка (рисунок 2.4). Викликати це вікно можна з головного меню командою **Object** меню **View** або командою **View Object** контекстного меню об'єкта з групи **Forms** у **Провіднику проекту**.

Для точного позиціонування об'єктів у формі у вікні є сітка. Розмір комірок сітки можна змінювати. При необхідності сітку можна відключати, скориставшись меню **Tools**, діалоговим вікном **Options**, його вкладкою **General** та віконцем **Show Grid.**

Вікно має рядок заголовка, подібний за структурою рядку заголовка головного вікна.

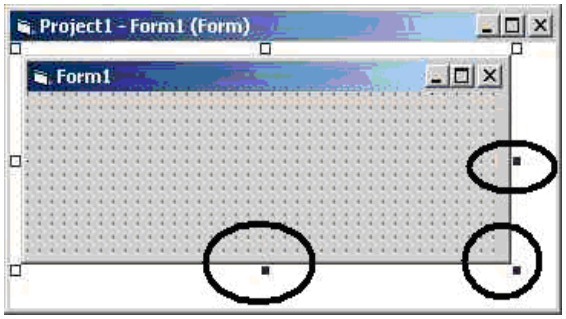


Рисунок 2.4 – Вікно конструктора форм

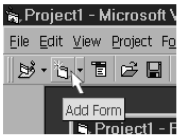
Ім'я вікна складається з імені проекту (у нас – **Project1**), до якого через дефіс приписане ім'я розміщеної в ньому форми (у нас – **Form1**); за цим ім'ям у круглих дужках зазначений клас форми (**Form**).

За замовчуванням, при відкритті нового проекту відкривається одне вікно екранної форми ***Form1.*** *При потребі можна відкривати додаткові вікна форм, імена яких, за замовчуванням,* ***Form2****,* ***Form3*** *і т.д. Всі ці вікна будуть рівноправні й незалежні одне від одного.*

Щоб зберегти форми та проект для подальшої роботи, над ними треба змінити їх назви. ***У*** *вікні провідника проекту потрібно клацнути мишею рядок з ім'ям проекту і у вікні властивостей проекту змінити значення єдиної його властивості (****Name****) – змінити слово* ***Project1*** *на обране вами*

*ім'я. Аналогічно зробіть зі значенням властивості (**Name**) екранної форми – змініть **Form1** на обране вами ім'я.*

Інтерфейс створюваного додатка з такого типу незалежними вікнами називається **SDI** (Single Document Interface) – однодокументальний інтерфейс. Бувають додатки й з іншим типом інтерфейсу **MDI (**Multiple Document Interface) – багатодокументальний інтерфейс, у яких існує єдине головне ("батьківське") вікно, що містить дочірні вікна документів, як, наприклад, у текстовому процесорі Microsoft Word або в самому інтегрованому середовищі VB (рисунок 2.5).



Діалог, що відкривається при виконанні команди Add Form

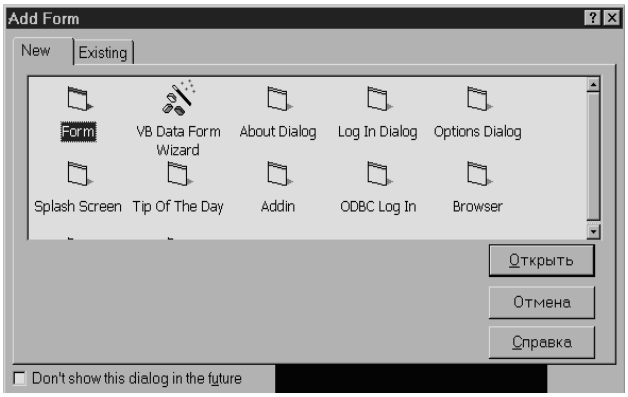


Рисунок 2.5 – Вибір піктограми **Add Form** (Додати форму)

Вікно екранної форми завжди має "зворотний бік", що являє собою вікно коду (точніше – вікно редактора коду).

У файлі форми завжди присутній й стосовний до оброблювачів пов'язаних з нею подій код.

До нього можемо перейти, викликаючи вікно коду подвійним клацанням на формі, або з головного меню командою **Object** через значок **View Code** вікна **Провідника проекту**.

Таким чином ми входимо у вікно текстового редактора коду VB, що забезпечує доступ до оброблювачів подій для активної форми й управляючих елементів, що містяться в ній.

Вікно екранної форми займає більшу частину головного вікна, воно нібито зрослося з ним. Три відомі нам кнопки управління вікном форми розташовані у правому кінці лінійки рядку заголовка. Клацнувши другу з них, можна відділити вікно екранної форми від головного вікна проекту.

При максимізації активного вікна форми область його заголовка сполучається з областю заголовка головного вікна, й ім'я вікна форми (у квадратних дужках) приписується праворуч до імені головного вікна через дефіс. Розмір вікна форми можна змінювати, використовуючи маркери виділення форми й миша. Для зміни розміру необхідно встановити покажчик миші на маркер і, коли він

набуде вигляду двонаправленої стрілки, переміщати до одержання необхідного розміру.

Крім того, в лівому кінці вікна розташовується значок системного меню ,що дозволяє змінювати геометрію вікна і його місце розташування, а також робити над ним ті ж дії, на які здатні й три відомі нам кнопки зміни статусу вікна у правому кінці цього рядка (згортання, максимізації/відновлення, закриття).

2.2.2.2 Вікно елементів управління

Вікно, на якому розташовується безліч піктограм різних елементів управління (рисунок 2.6), називається **вікном** або **панеллю елементів управління** (**Toolbox**).

Елемент управління – це візуалізований засіб для створення об'єктів управління на формі.

Це основний робочий інструмент при розробленні форм додатка. До складу панелі елементів управління входять компоненти, з яких буде складатися інтерфейс вашого додатка (крім покажчика).Елементи, які ви бачите на рисунку, стандартні – це мітки, текстові поля, кнопки, списки й інші елементи для швидкого візуального проектування макета форми. На панелі подані кнопки, призначення яких описане в таблиці 2.2.

Кнопки у вікні **Toolbox –** це не самі об'єкти управління, а тільки їх зразки, шаблони. Обравши той чи інший зразок, можна створити його копію на екранній формі, яка називається екземпляром класу об'єктів управління. Таких копій можна створити стільки, скільки зможе вмістити екранна форма.

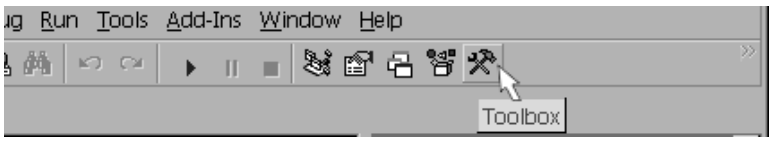


Рисунок 2.6 – Панель елементів управління **Toolbox** (ліворуч) і піктограма управління нею на стандартній панелі інструментів (праворуч)

Таблиця 2.2 – Елементи управління

Кнопка	Назва	Призначення
1	2	3
	**Pointer** (Покажчик)	Використовується для позиціювання маркера (покажчика) миші
	**PictureBox** (Графічне вікно)	Розміщує на формі графічне вікно, призначене для об'єднання елементів у групи, для виведення в нього графічних зображень, а також тексту, графічних елементів і анімації
	**Label** (Мітка)	Розміщує на формі об'єкти, призначені для створення текстової інформації, написів і приміток
	**TextBox** (Текстове поле)	Розміщує на формі текстове поле, призначене для введення текстової інформації, чисел і дат
	**Frame** (Рамка)	Створює на формі рамку із заголовком для об’єднання об'єктів у логічну групу
	**CommandButton** (Кнопка управління)	Розміщує на формі кнопки управління для ініціації дій, виконання команд, запуску програм
	**CheckBox** (Прапорець)	Розміщує на формі прапорець, призначений для формування умов виконання програм або яких-небудь настроювань, що працює за принципом “так-ні”
	**OptionButton** (Перемикач)	Створює у формі перемикачі для вибору режиму роботи або налаштування виконання програми
	**ComboBox** (Поле зі списком)	Створює на формі об'єкт, що містить одночасно поле введення й список, що розкривається
	**ListBox** (Список)	Створює на формі список для вибору одного або декількох значень із пропонованого списку значень

Продовження таблиці 2.2

1	2	3
	**HScrollBar** (Горизонтальна смуга прокручування)	Розміщує у формі горизонтальну смугу прокручування, яка використовується як покажчик для вибору значення із заданого діапазону
	**VScrollBar** (Вертикальна смуга прокручування)	Розміщує на формі вертикальну смугу прокручування, яка використовується як покажчик для вибору значення із заданого діапазону
	**Timer** (Таймер)	Розміщує на формі таймер
	**DriveListBox** (Список пристроїв)	Створює на формі список пристроїв
	**DirListBox** (Список папок)	Створює на формі деревоподібний список папок
	**FileListBox** (Список файлів)	Створює на формі список файлів

	Shape (Обрис)	Створює у формі геометричні фігури, такі як прямокутник, квадрат, коло, еліпс, прямокутник і квадрат з округленими кутами
	Line (Лінія)	Створює лінії
	Image (Зображення)	Створює у формі поля, призначені для відображення графічних зображень
	Data (Дані)	Створює елемент управління даними в базі даних для переміщення по записах і відображення результату навігації

*В інтегрованому середовищі VB активно використовуються контекстні меню, що викликаються клацанням по об'єкту правою клавішею миші (або комбінацією **Shift+F10**). Зробивши таке клацання по панелі елементів управління, ми одержуємо таке меню (рисунок 2.7).*

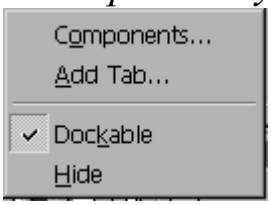


Рисунок 2.7. – Контекстне меню панелі елементів управління

Горизонтальною рискою меню розділено на дві групи. Верхня містить дві команди:

- **Components** (Компоненти) відповідає за додавання на панель нових **додаткових елементів управління**. По цій команді відкривається вікно компонентів для створення об’єктів найрізноманітнішого призначення: лінійки статусу, смуг закладок, вікон і сітки даних, а також елементів Кліп і Мультімедіа – для включення в проекти анімації та звуку.
- **Add Tab** (Додати вкладку) використовується при недостачі місця на панелі (і небажанні збільшувати її розмір) і полягає у створенні іменованих вкладок, на яких можна розміщати додаткові компоненти.

Первинна вкладка має ім'я **General** (Головна).

Нижня група надає можливість забирати панель елементів з екрана, клацаючи по стану **Hide** (Схована), причому того ж досягають, клацаючи по кнопці закриття інструментального вікна. Коли панель на екрані, галочкою відзначений її стан – **Dockable** (Виставлена).

*Пункти **Hide** і **Dockable** є в контекстному меню й для ряду інших типів вікон.*

2.2.2.3 Вікно провідника проекту

Це вікно містить **графічне уявлення вмісту проекту** – дерево або список усіх файлів, що входять у проект.

*Вікно провідника проекту **Project** (рисунок 2.8) дуже схоже на аналогічне вікно провідника системи Windows і дозволяє легко й швидко переглядати склад і властивості обраного проекту, переміщатися між проектами, якщо їх відкрито відразу декілька, копіювати необхідні об'єкти з вікна одного проекту в інший, як це здійснюється в провіднику системи Windows.*

Воно призначене для швидкого доступу до тієї або іншої екранної форми та одержання інформації про об'єкти, які становлять проект, наприклад: форми, модулі і т.д. За допомогою цього вікна можна легко переходити від редагування одного об'єкта до іншого.

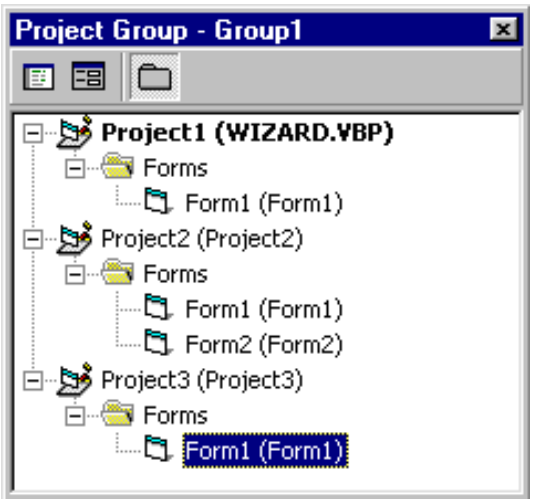


Рисунок 2.8 – Вікно провідника проекту

Воно в багато разів полегшує роботу з великими проектами.

Провідник проекту можна викликати командою **Project Explorer** меню **View** або комбінацією клавіш **<Ctrl>+<R>.**

Панель управління вікна провідника проекту містить три кнопки, які мають такі призначення (таблиця 2.3).

Таблиця 2.3 – Кнопки панелі управління провідника проекту

Кнопка	Призначення
	Відкриває вікно редактора з кодом програми, обраного в провіднику об'єкта
	Відкриває в конструкторі форм обраний об'єкт
	Включає/виключає відображення папок

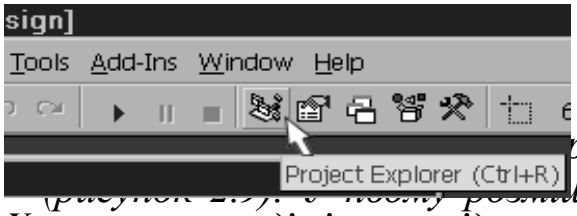
При натисканні правої кнопки миші у вікні провідника з'являється контекстне меню, що містить команди для додавання,

збереження, видалення форм, елементів управління й інших об'єктів. У контекстному меню продубльовані дії кнопок вікна провідника.

У вікні **Провідника проекту**, що має ім'я **Project**, до якого через дефіс додається ім'я проекту (рисунок 2.9), перелічуються всі файли, що формують проект.

Щоб переглянути яку-небудь форму, потрібно виділити її, клацнувши лівою кнопкою миші по її імені, потім натиснути кнопку **View Object**  . Якщо вам потрібно переглянути код програми, також виділіть ім'я об'єкта, але натисніть на кнопку **View Code**  .

Можна зробити подвійне клацання по імені файлу, що містить екранну форму, активізувати її та відобразити на екрані поверх інших.



У нашому прикладі вікно провідника проекту відображає файл проекту Project1.vbp і два файли форм – Form1.frm та Form2.frm.

Рисунок 2.9 – Вікно **Провідника проекту** (**Project Explorer**) (праворуч) і піктограма його виклику на стандартній панелі інструментів (ліворуч)

Перші дві піктограми забезпечують можливість переходу від його візуальної складової до програмної (тобто до кодів оброблювачів подій) і навпаки. Третя піктограма перемикає режим відображення структури файлів проекту у вікні: у формі звичайного списку або у вигляді дерева.

*Ми вже казали, що **Контекстне меню**, яке викликається натисканням правої клавіші усередині вікна провідника проекту, забезпечує швидкий доступ до багатьох команд, що дублюють команди пунктів **File** і **Project** головного меню. Склад цього меню залежить від того, який тип файлу*

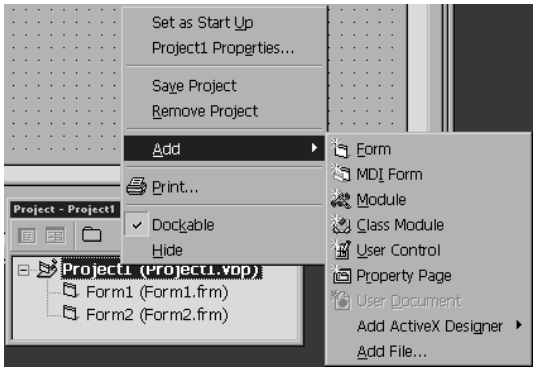


Рисунок 2.10 – Контекстне меню для швидкого доступу

*виділений у даний момент у списку вікна: файл проекту або файл форми. Більш повним є меню при виділеному файлі проекту (для повноти картини на рисунку 2.10 показано також підлегле меню команди **Add**).*

Пункт **Set as Start Up** (Установити як стартовий) має сенс у випадку одночасної роботи з декількома проектами. Другий пункт (**Project1 Properties**...) викликає діалогове вікно властивостей проекту, дублюючи відповідну команду з пункту **Project** головного меню.

Команда **Print** (Друк) дозволяє роздруковувати як код файлу екранної форми, так і саму форму. Команди **Remove Project** – Видалити проект і **Save Project** – Зберегти проект.

2.2.2.4 Вікно перегляду характеристик (класів) об'єктів

Для перегляду всіх елементів, що входять до складу проекту, VB надає дуже зручну можливість – вікно перегляду об'єктів **Object Browser** (рисунок 2.11). У цьому вікні є два списки:

- список *Класів* (***Classes***);
- список характеристик обраного *Класу* (***Members of Class***).

Виділивши певний *Клас*, ви отримуєте доступ до характеристик виділеного *Класу*: властивостей, методів, подій.

Будь-який рядок кожного списку містить не тільки назву класу та назву характеристики цього класу, але і відповідну піктограму.

В нижній частині вікна виводиться текст з довідковою інформацією про характеристику, яка в даний момент виділена в правому списку.

Вікно перегляду об'єктів звичайно не визуалізовано і його можна викликати командою **Object Browser** (Браузер об'єктів) з меню **View**.

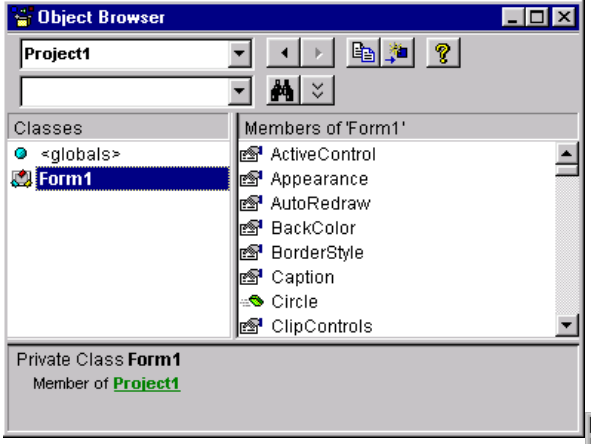
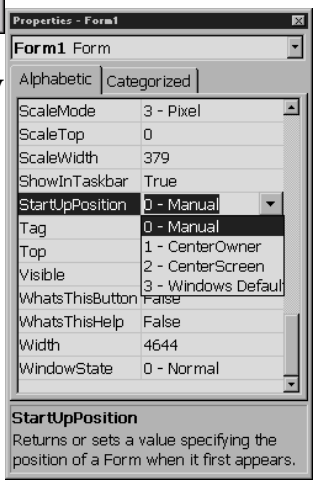


Рисунок 2.11 – Вікно перегляду об'єктів **Object Browser**

2.2.2.5 Вікно властивостей



Всі об'єкти VB, розміщені у формі (заголовок, поля, надписи, кнопки, лінії й т.д.), а також сама форма характеризуються властивостями, які ви можете настроїти відповідно до своїх вимог. **Вікно властивостей (Properties Window)** призначено для установлення властивостей об'єктів на стадії проектування (рисунок 2.12).

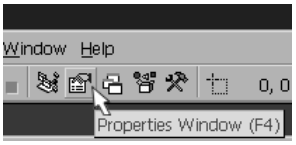


Рисунок 2.12 – Вибір піктограми **Properties Window** (ліворуч) і вікно властивостей (праворуч)

Оскільки елементи управління кожний сам по собі є об'єктом, набір властивостей у цьому вікні змінюється залежно від обраного об'єкта.

Список властивостей розділений на два стовпчики. У лівому розміщені імена властивостей зазначеного у верхнім віконці об'єкта, а в правому – значення, установлені для цих властивостей за замовчуванням або програмістом.

Можна згорнути або розгорнути список властивостей конкретної категорії, клацнувши по квадратику ліворуч від заголовка кожної з категорій.

Список властивостей міститься на двох вкладках: **Alphabetic** (Алфавітний) і **Categorized** (за категоріями), що позначають принцип упорядкування списку властивостей: за алфавітом або за категоріями.

Перемикатися на вкладку ***Categorized*** *може бути зручно тоді, коли ім'я властивості забуте, але відомі його функціональні ознаки.*

Під списком властивостей у вікні **Properties** виводиться короткий опис поточної виділеної властивості.

Властивістю об’єкта є якісна або кількісна характеристика цього об’єкта – його розміри, положення на екранній формі, колір самого об’єкта та колір розміщеного на ньому тексту, характеристики шрифту цього тексту та багато іншого.

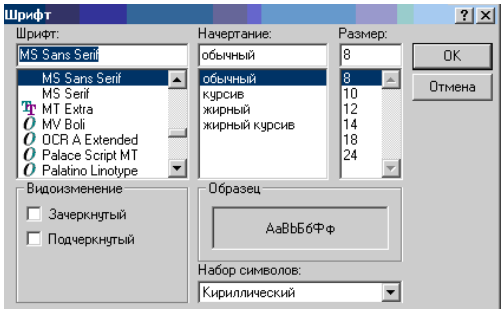


Рисунок 2.13 – Діалогова панель Windows для вибору складних властивостей

Для більш складних властивостей, наприклад ***Font,*** *існує можливість скористатися для вибору стандартними діалоговими панелями Windows (рисунок 2.13)*

Використовуючи діалогове вікно ***Properties****, можна змінити попередньо встановлені властивості об'єктів. Частина властивостей об'єкта, наприклад, розміри й розташування, можна задати переміщенням і зміною його розмірів за допомогою миші в конструкторі форм. Як правило, форма містить багато об'єктів.*

Якщо вибрати одразу декілька об'єктів, то у вікні властивостей будуть відображені загальні для цих об'єктів властивості.

2.2.2.6 Вікно макета форми

Вікно **Form Layout** (*Вікно макета або розташування форми)* служить для задання положення екранної форми на екрані при її завантаженні в процесі виконання проекту (рисунок 2.14).

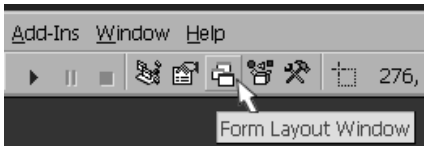


Рисунок 2.14 – Вибір піктограми **Form Layout Window** (Вікно зовнішнього вигляду). Праворуч – розміщення двох форм за допомогою цього вікна на екрані монітора

Вікно макета форми викликається командою **Form Layout Window** з меню **View**. У цьому вікні показується зменшене зображення проектованої форми в тому вигляді, який ця форма буде мати вигляд на екрані монітора при виконанні додатка.

Розміри форми у вікні макета ***Form Layout*** *пропорційні розмірам зображення монітора в цьому ж вікні, тобто всі реальні пропорції в точності дотримані.*

Для перевірки вигляду форми в працюючому додатку відкрийте вікно макета форми й запустіть форму командою ***Start*** *з меню* ***Run.***

Розміри форми встановлюються у вікні форми перетягуванням розмірних маркерів (темних квадратиків), розташованих на правій і нижній границях форми.

А от задавати положення форми із установленими розмірами на екрані під час виконання можна, викликаючи вікно **Form Layout** і перетягуючи форму мишею усередині стилізованого зображення екрана монітора. При перетягуванні форми змінюються значення її властивостей **Left** та **Top,** що можна побачити на індикаторі положення та розмірів об’єкта в правій частині лінійки інструментів *Головної панелі* проекту.

Для точного установлення форми в центрі екрана можна викликати контекстне меню вікна й у підлеглому меню пункту **Startup Position** (Стартова позиція) вибрати значення **Center Screen** (Центр екрана).

Альтернативними значеннями є: **Manual** (Задавати вручну), **Center Owner** (По центру в області додатка) і **Windows Default** (За замовчуванням – форма у верхньому лівому куті екрана).

2.2.2.7 Вікно редактора вихідного коду

Редактор коду VB – це досить потужний текстовий редактор з великою кількістю можливостей. Він є основним інструментом програміста для створення й відлагодження додатка.

Відразу під рядком заголовка у вікні редактора (рисунок 2.14) розміщені два вікна зі списками, що розкриваються. Кожне з них містить один рядок та кнопку зі стрілкою справа від вікна, щоб відкрити відповідний список.

- перший список (**Object)** – перелік процедур об'єктів проекту – забезпечує вибір об'єктів додатка. При виборі об'єкта в цьому списку синхронно змінюється вміст списку **Procedure** (*цей список розміщений у лівому верхньому куті вікна редактора)*;
- другий список (**Procedure)** – перелік процедур об'єкта, виділеного у першому списку – дає можливість обрати події, які можуть відбуватися з цим об'єктом.

Винятком є елемент (**General**) у першому списку – це не об'єкт, а загальна частина програми, яка належить до всіх об'єктів відразу, та елемент (**Declaration**) у другому списку – це не процедура, а *Розділ загальних описів*, які належать до усіх процедур відразу (наприклад, обява глобальних змінних, що використовуються в різних процедурах).

Під вікном і праворуч розташовані відповідно горизонтальна та вертикальна смуги прокручування.

Над вертикальною смугою є невеличка кнопка (**Split**), яка дозволяє розщепити вікно на дві частини, в кожній з яких розміщується увесь текст програми. Коли текст великий – цим зручно користуватися при редагуванні.

Зліва від горизонтальної смуги є дві кнопки:

- кнопка **Procedure View** (Перегляд процедур) – включає режим перегляду процедур для кожного об'єкта окремо;
- кнопка **Full Module View** (Повний перегляд модулів) – включає режим повного перегляду процедур, при якому у вікні редактора показані всі процедури, розділені горизонтальною лінією (якщо встановлено відповідний прапорець настроювання).

Як можна зрозуміти зі списку елементів управління, редактор коду працює у двох режимах: у режимі перегляду всього тексту додатка – повний перегляд процедур (рисунок 2.15) і в режимі перегляду процедур окремо – перегляд окремих процедур (рисунок 2.16).

Редактор коду викликається автоматично при подвійному клацанні миші на формі проекту або командою **Code** меню **View**.

Рисунок 2.15 – Редактор коду в режимі повного перегляду процедур

Рисунок 2.16 – Редактор коду в режимі перегляду процедур

Для кожного елемента проекту (форми або програмного модуля) відкривається окреме вікно редактора коду. Відповідно це вікно з'являється в списку вікон меню **Window**.

За допомогою цієї опції можна зробити так, щоб вікно коду та вікно відповідної форми було у вас перед очима одночасно (рисунок 2.17).

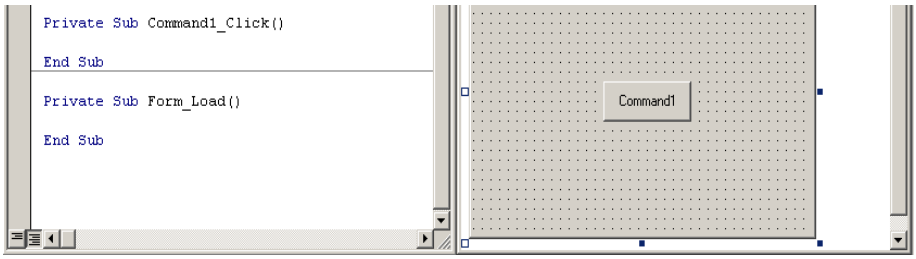


Рисунок 2.17 – Одночасне зображення вікна коду та вікна форми

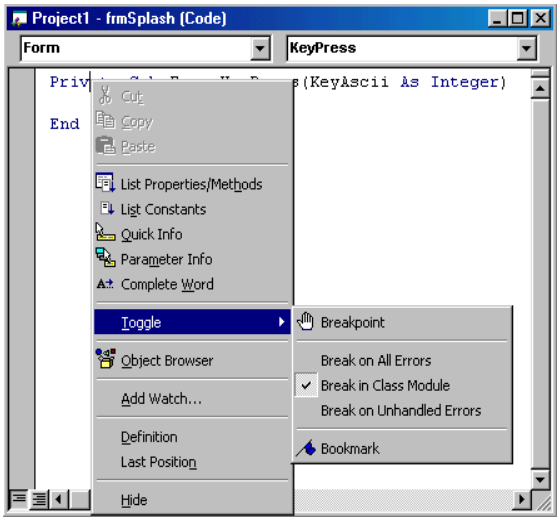


Рисунок 2.18 – Контекстне меню редактора коду

У великому проекті найзручніше працювати із **Провідником проекту**. У цьому випадку редактор коду викликається кнопкою **View Code** панелі інструментів вікна **Провідника**.

Для роботи у вікні редактора можна використовувати контекстне меню (рисунок 2.18). Воно містить команди, наведені в таблиці 2.4.

Таблиця 2.4 – Команди контекстного меню редактора коду

Команда	Призначення
Cut (Вирізати)	Вирізає виділений текст і вставляє його в буфер обміну
Copy (Копіювати)	Копіює виділений текст у буфер обміну
Paste (Вставити)	Вставляє текст із буфера обміну
List Properties/Methods (Список властивостей/методів)	Показує список властивостей і методів для зазначеного об'єкта
List Constants (Список констант)	Показує список констант
Quick Info (Швидка інформація)	Виводить синтаксис оператора, що вводиться
Parameter Info (Інформація про параметри)	Виводить список параметрів окремої функції або оператора
Complete Word (Повне слово)	Викликає список властивостей і методів. При цьому позиціювання курсору в списку здійснюється за початковими буквами введеного тексту коду. Наприклад, якщо вами введена буква "V", то курсор у списку буде встановлений на властивості Value. Обрану властивість або метод можна вставити із цього списку в текст коду подвійним клацанням миші
Toggle (Установка)	Викликає меню для установлення точок останова, переривань роботи додатка у випадку помилок, установлення закладок у вихідному тексті
Object Browser (Перегляд об'єктів)	Відкриває вікно Object Browser

Add Watch (Додати спостереження)	Відкриває діалогове вікно Add Watch
Definition (Опис)	Наводить опис зазначеного об'єкта (викликаючи вікно Object Browser), змінної або константи
Last Position (Остання позиція)	Позиціонує курсор на команді в редакторі коду, що була відредагована (переглянута) останньою
Hide (Забрати)	Закриває поточне вікно

2.2.2.8 Вікно редактора меню

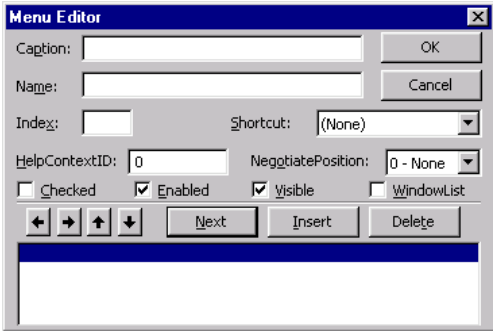


Рисунок 2.19 – Вікно редактора меню

У вікні редактора меню (рисунок 2.19) можна створювати або редагувати рядок меню для форми. Вікна ***Locals, Watches, Immediate*** призначені для відлагодження додатків і будуть розглянуті нижче**.**

3 ОСНОВНІ ОБ'ЄКТИ VB 6.0

Кожному діалоговому елементу у VB поставлений у відповідність певний набір подій, що відбуваються в період виконання програми.

Наприклад, подія Load (Завантаження) відбувається при завантаженні форми, подія Click (Клацання) викликається клацанням кнопки миші, подія DblClick (Подвійне клацання) викликається подвійним клацанням кнопки миші й т.д.

У свою чергу, кожній події ставиться у відповідність процедура її обробки – набір операторів, що виконуються при виклику процедури. Тобто подія може викликати зміну даних, що програмуються. VB містить заготовки таких процедур у полі для запису програми, що відповідає виділеній події. У полі введення тексту програми автоматично формується заголовок процедури й кінцевий оператор. Ім'я процедури формується автоматично й складається із двох частин, розділених підкресленням: ім'я виділеного діалогового елемента й ім'я виділеної події.

Приклад

Рядки **Sub Form_Load ()** та **End** визначають перший і останній оператори процедури обробки події **Load** (Завантаження).

3.1 Робота з формами (UserForm)

Об'єкт **Форма** (рисунок 3.1) має близько 50-ти властивостей.

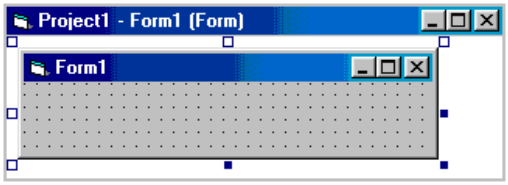


Рисунок 3.1 – Об'єкт **Форма**

Однією з найважливіших властивостей цього об'єкта є властивість **Name** (ім'я даного об'єкта).

Форма *може мати ім'я, відмінне від імені, заданого VB попередньо.* Змістовні імена полегшують роботу з ***Формами*** й іменами в програмі.

Розглянемо деякі властивості, методи й події, характерні для ***Форми*** *(таблиці 3.1-3.3)***.**

Таблиця 3.1 – Найбільш часто використовувані властивості об'єкта UserForm

Властивість	**Опис**
1	2
Caption	Текст, виведений як заголовок форми
BackColor	Визначає кольори фону форми. Найпростіший спосіб встановити цю властивість – використати Properties Window; щоб вибрати бажані кольори (якщо необхідно), можна скопіювати номер кольорів з Properties Window у свою програму
1	2
ForeColor	Те ж, що й властивість **BackColor**, але встановлює кольори, які використовують для переднього плану об'єкта форми. Звичайно – це кольори тексту
Font	Повертає посилання на об'єкт **Font**, за допомогою якого можна вибрати параметри шрифту форми або елемента управління
Enabled	Містить значення типу **Boolean**, що вказує, чи доступна форма. Якщо його значення дорівнює **False**, жоден з елементів управління форми не доступний під час виконання
ScaleMode	Встановивши цій властивості потрібне значення на етапі розроблення додатка або в програмному коді, можна отримати бажані одиниці виміру
BorderStyle	Можливість змінювання розмірів вікна
Left, Top	Координати лівого верхнього кута вікна
WindowStyle	Стан вікна у момент запуску програми
Width, Height	Ширина і висота вікна

Таблиця 3.2 – Методи об'єкта UserForm

Метод	Опис
Copy	Копіює виділений в елементі управління текст у буфер обміну Windows
Cut	Видаляє виділений в елементі управління текст і розміщує його в буфер обміну Windows
Paste	Вставляє вміст буфера обміну Windows у поточний елемент управління
Hide	Приховує UserForm, не вивантажуючи її з пам'яті, зберігаючи значення елементів управління форми й всіх змінних, оголошених у модулі класу форми
PrintForm	Виводить на використовуваний у Windows за замовчуванням принтер зображення форми, включаючи всі дані, введені в елементи управління
Repaint	Перемальовує форму, виведену на екран
Show	Виводить форму на екран. Якщо вона не завантажена, то VB спочатку її завантажує

Всі форми VB є модальними (**modal**) додатками. Це означає, що не можна виконати яку-небудь іншу дію в додатку доти, поки форма діалогу не буде закрита або схована.

Таблиця 3.3 – Події об'єкта UserForm

Подія	Опис
1	2
Activate	Ініціюється щоразу, коли вікно форми стає активним. Використовується для відновлення вмісту діалогових елементів управління, щоб відобразити будь-які зміни, які відбулися, поки вікно форми було неактивним
Click	Ініціюється щоразу, коли на формі (або будь-якій її частині, не зайнятій елементами управління) клацають мишею
DblClick	Ініціюється щоразу, коли на формі (або будь-якій її частині, не зайнятій елементами управління) двічі клацають мишею
Deactivate	Ініціюється щоразу, коли форма перестає бути активною
Initialize	Ініціюється щоразу, коли форма вперше завантажується у пам’ять за допомогою виконання оператора Load або за допомогою методу Show. Використовується для ініціалізації елементів управління форми, як тільки вони з'являться на екрані
Resize	Ініціюється при зміні розмірів форми
Terminate	Ініціюється щоразу, коли форма вивантажується з пам'яті. Використовується для здійснення будь-яких спеціальних службових завдань, які необхідно виконати перш ніж змінні форми будуть вивантажені

На додаток до методів, властивостей і подій, вбудованих в об'єкт **UserForm**, VB надає два **оператори,** які особливо корисні при роботі з об'єктами форм:

- **Load** – завантажує форму в пам'ять, але не відображає її на екрані;
- **Unload** – вивантажує форму з пам'яті та видаляє її з екрана.

3.2 Вибір і використання елементів управління

Елементи управління (**controls**) – це елементи діалогового вікна, які дають можливість користувачеві взаємодіяти із програмою. Вони містять у собі кнопки-перемикачі, текстові поля, лінійки прокручування, командні кнопки й так далі. Кожен елемент управління – це об'єкт із певними властивостями, методами й подіями. Як і для форми, що їх містить, можна встановлювати властивості елементів управління програмним шляхом або за допомогою **Properties Window**. У програмі можна привласнювати або відновлювати значення властивостей елементів управління так само, як для будь-яких інших об'єктів.

Усі елементи управління форми повинні мати унікальні імена. Ці імена варто використати при посиланнях на елемент управління у своїй програмі.

Щоразу, коли додається до форми новий елемент управління, VB привласнює йому ім'я за замовчуванням, що складається з імені типу елемента й номера. *VB забезпечує унікальність імені включенням в ім'я елемента управління числа*.

3.3 Кнопка управління (Command Button)

Command Button (Командна кнопка) – *один з найпоширеніших об'єктів у Windows-додатках. Вона використовується для вирішення всіляких задач: від найпростішого введення інформації до виведення спеціальних функцій, зв'язаних Windows-додатком*.

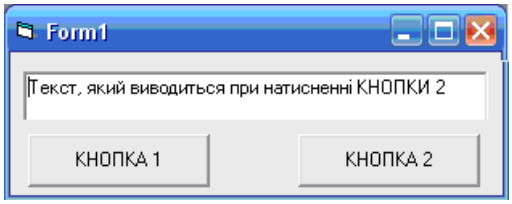


Рисунок 3.2 – Інформація в текстовому полі змінюється при натисканні кнопок

Натискання кнопки, розміщеної у формі, дозволяє виконати процедуру обробки події **Click** (рисунок 3.2).

Процедурами, пов'язаними з обробкою цієї події, може бути, наприклад, друк даних або

проведення певних обчислень. Після того як кнопка розміщена у формі й задана її назва, необхідно визначити дії, виконувані при натисканні на цю кнопку. Для цього двічі клацніть по кнопці й у вікні редактора коду, що відкрилося, задайте необхідну процедуру.

Можна використовувати до 40 властивостей.

*Дві найважливіші з них – **Name** та **Caption.***

За значенням **Name** VB відрізняє одну кнопку від іншої. Якщо треба змінити напис на кнопці – використовується властивість **Caption**. Напис може містити не більше 255 символів.

Якщо довжина напису більше ширини кнопки, автоматично здійснюється перенос напису на такий рядок. У випадку, якщо розмір кнопки не дозволяє помістити весь напис, частина напису,що не помістилася, відкидається.

Для управління зовнішнім виглядом кнопки використовується властивість **Style** (Стиль). Вона містить два значення.

За замовчуванням установлене значення **Standard** – кнопка буде містити текст, що задається властивістю **Caption.**

Можна використовувати замість текстового напису на кнопці графічне зображення, тоді властивість **Style** повинна мати значення **Graphical**.

*Для задання графічного зображення, що поміщається на кнопці, виділіть властивість **Picture** і натисніть кнопку, розташовану в правому стовпці. У результаті відкриється діалогове вікно **Load Picture,** в якому ви можете вибрати на диску файл, що містить зображення.*

Розглянемо властивості, характерні тільки для **Command Button:**

- **Cancel – *True*** – для автоматичного виклику процедури **Click** при натисканні клавіші **Esc**;
- **Default** – ***True*** – для автоматичного виклику процедури **Click** при натисканні клавіші **Enter**;
- **DisabledPicture** – рисунок, зображуваний на кнопці, коли вона не доступна (властивість **Enabled**=***False***), якщо властивість **Style=**1;
- **DownPicture** – рисунок, зображуваний на кнопці, коли вона натиснута, якщо властивість **Style=**1;
- **TabIndex** – порядковий номер у послідовності переходу (при натисканні **Tab**);
- **ToolTipText** – спливаюча підказка для кнопки.

При запуску додатка, як правило, один з наявних на формі об'єктів має бути активним, тобто обробляти певним чином інформацію, одержувану від миші або клавіатури.

У тому випадку, коли об'єкт одержує таку інформацію, говорять, що *об'єкт **має фокус*** або для нього виконується подія **GotFocus**. З іншого боку, *при **втраті фокуса*** об'єктом відбувається подія **LostFocus**.

Якщо управляюча кнопка має фокус, то вона відображається з виділеною рамкою на формі.

Одержання фокуса кнопкою, або, іншими словами, вибір її при виконанні додатку може бути реалізований декількома способами:

- при натисканні на ній лівою кнопкою миші;
- використовуючи клавіші переходу **Tab** або стрілки управління курсором, а імітувати натискання лівої кнопки миші – за допомогою клавіші "***пропуск***" або **Enter**;
- виконуючи для заданого об'єкта метод **SetFocus**;
- за допомогою клавіш ***Alt* + *підкреслена буква*** в назві кнопки.

Деякі елементи управління не можуть отримати фокус. Це, в першу чергу, графічні елементи: **Frame**, **Image**, **Label**, **Line**, **Shape** (контур)**,** а також елемент, що не відображується на екрані**, Timer**.

3.4 Мітка (Label)

Об'єкт **Label** призначений для створення текстових полів у вікні програми.

Окрім загальних властивостей, об'єкт **Label** *має ще й інші влативості (див.таблицю 3.4).*

Текст, що задається об'єктом **Label**, може мати досить великий розмір і займати кілька рядків. Максимальна кількість його символів – 65528.

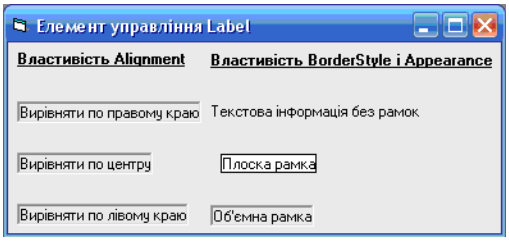


Рисунок 3.3 – Оформлення надписів

Таблиця 3.4 – Властивості Label

Властивість	Опис
Alignment	Вирівнювання тексту в межах поля
AutoSize	Проведення меж поля до границь тексту
Visible	Видимість об'єкта

WordWrap	Перенесення слів тексту в новий рядок

Текст задається властивістю **Caption**. *Виділіть дану властивість, після чого в правий стовпець властивості введіть потрібну текстову інформацію.*

У випадку, коли текст, що міститься в мітці, не вміщується в одному рядку, то, за попереднім визначенням, він переноситься на наступний рядок, якщо дозволяє задана висота елемента. В іншому випадку, та частина, що вийде за наявні межі, відображена не буде.

Для автоматичного розширення мітки на формі необхідно встановити властивість **AutoSize=*True***. Якщо необхідно розташувати текст на декількох рядках з автоматичним збільшенням її висоти, варто дозволити перенос слів установкою **WordWrap=*True***. Перенесення, встановлене таким чином, буде працювати, якщо встановлена властивість **AutoSize=*True***.

Шрифт текстової інформації визначається властивістю **Font** (Шрифт).

> *Для вибору шрифту у вікні властивостей встановіть курсор у дану властивість і натисніть кнопку із трьома крапками в правому стовпці властивості. Відкривається діалогове вікно **Вибір шрифту,** що містить три списки, що дозволяють вказати найменування, накреслення й розмір шрифту.*

Використовуючи властивості **ForeColor** і **BackColor**, можна задати колір текстової інформації й колір фону.

Властивість **BorderStyle** (Стиль рамки) визначає тип обрамлення навколо мітки, дозволяючи оформити напис у вигляді текстового поля. Для цього замість використовуваного за замовчуванням значення **None** необхідно вибрати для властивості значення **Fixed Single** (рисунок 3.3).

Властивість **Appearance** дозволяє додати тексту деяку об'ємність.

Текст, що задається об'єктом **Label**, не може бути змінений користувачем додатка. Але це не означає, що його не можна змінити при виконанні додатка, тобто програмно.

3.5 Текстове поле (TextBox)

Для відображення інформації й введення даних у формі призначений об'єкт типу **поле**, для створення якого

використовується кнопка **TextBox** (Текстове поле) на панелі елементів управління.

Зважаючи на те, що в елемента **TextBox** відсутня властивість **Caption**, у якості її заміни використовують мітку **Label** з відповідним текстом.

Інформація, виведена в текстовому полі, визначається властивістю **Text** і може задаватися в такий спосіб:

- у діалоговому вікні **Properties**;
- у процесі виконання програмного коду;
- при введенні даних у поле користувачем. У цьому випадку властивість **Locked** не повинна мати значення ***True***, тому що дані, відображувані в полі, можна тільки переглядати, але не можна редагувати.

Багаторядкові текстові поля

Якщо текст не вміщується в заданих межах текстового поля, можна дозволити перенос слів шляхом установлення властивості **MultiLine=*True***. Разом з тим, необхідно вибрати один з варіантів для смуг прокручування тексту (властивість **ScrollBar**), тому що розмірів поля може не вистачити для відображення всієї введеної інформації (рисунок 3.4).

*За замовчуванням передбачається, що текстове поле служить для введення одного рядка тексту. Властивості **MultiLine** і **ScrollBar** дозволяють настроїти об'єкт таким чином, що він буде використовуватися для введення декількох рядків або навіть великого блоку текстової інформації (таблиця 3.5).*

Таблиця 3.5 – Призначення властивостей MultiLine і ScrollBar

Властивість	**Призначення**
MultiLine	Визначає спосіб відображення поля. При встановленому значенні True текст автоматично переноситься за словами на кілька рядків. При введенні інформації в поле для переходу на новий рядок необхідно використовувати клавішу <Enter>
ScrollBar	Дана властивість використовується, якщо властивість MultiLine має значення True і розмір поля не дозволяє відобразити заданий у властивості Text текст. Властивість може приймати одне з таких значень: 0-None-смуга прокручування відсутня; 1-Horizontal-горизонтальна смуга прокручування; 2-Vertical – вертикальна смуга прокручування; 3-Both – горизонтальна й вертикальна смуги прокручування

VB дозволяє під час виконання програми управляти текстом, відображуваним у текстовому полі, за допомогою властивостей **SelStart**, **SelLength** і **SelText**.

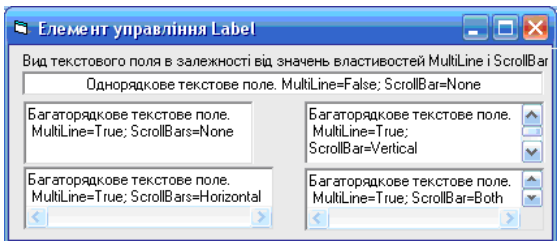


Рисунок 3.4 – Вид текстового поля залежно від значення властивостей **MultiLine** і **ScrollBar**

У випадку, коли фокус уперше переходить на текстове поле, курсор за замовчуванням встановлюється ліворуч від тексту, що розміщений на полі. У результаті введення або перегляду інформації курсор може переміщатися в межах поля. При наступному поверненні фокуса на поле курсор встановлюється в те місце, куди він був встановлений востаннє.

Для того, щоб при одержанні фокуса текстовим полем курсор перебував у заданій позиції (за попереднім визначенням – на початку тексту), використовується властивість

SelStart=<*позиція*>,

де <*позиція*> – порядковий номер символа в текстовому полі, перед яким буде розташована точка введення (при цьому нумерація символів починається з нуля).

Для точки введення можна задати не тільки позицію курсору, але й кількість символів, які будуть видалені, тобто яку частину тексту необхідно замінити першим введеним символом. У цьому випадку застосовується властивість

SelLength=<*кількість*>

За замовчуванням вона дорівнює **0**, тобто в тім місці, де курсор установлений, можна починати введення символів, не видаляючи розташованої в ньому інформації.

Властивість

SelText =<*текст*>

дозволяє задати <*текст*>, що замінить під час виконання програми видалений фрагмент.

Слід зазначити, що ці властивості доступні тільки в програмному коді і не можуть бути задані на етапі розроблення додатка.

Властивість **ToolTipText** містить текст підказки, який висвітлюється, якщо навести курсор миші.

Властивість **PasswordChar** містить символ, що буде відображатися, наприклад, при введенні пароля.

3.6 Перемикачі (OptionButton)

Перемикачі використовуються в тому випадку, якщо необхідно вибрати одну з взаємовиключних можливостей.

Value – це властивість, яка використовується в режимі конструювання і у режимі виконання.

Установлення одного перемикача в групі у режимі конструювання (присвоєння **Value** значення **True**) встановить його при відкритті форми й автоматично скидає інші перемикачі, привласнюючи аналогічним властивостям значення **False**.

За попереднім визначенням при відкритті форми буде встановлений тільки один перемикач. Перемикач має вигляд (див. рисунок 3.5), але може набувати і вигляду кнопки, якщо необхідно на даному перемикачі встановити яке-небудь зображення.

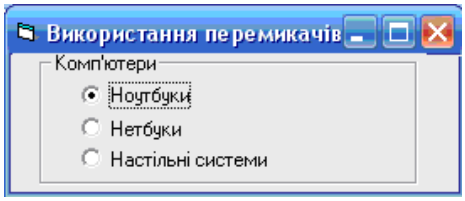


Рисунок 3.5 – Перемикач, поміщений в об'єкт-контейнер

Звичайно перемикачі групуються в рамках, але якщо використовується тільки одна група перемикачів, то їх можна групувати на формі.

Кожна логічна група має міститися в об'єкт-контейнер, наприклад, **Frame** або **PictureBox** (рисунок 3.5). У цьому випадку, для створення у формі групи перемикачів, необхідно спочатку помістити у форму рамку, а потім розмістити в ній по черзі необхідну кількість перемикачів, використовуючи для цього кнопку **OptionButton** на панелі елементів управління.

При запуску додатка на виконання, що містить групу перемикачів, перший з них установлюється за замовчуванням. Щоб під час розроблення будь-який перемикач групи зробити використовуваним за замовчуванням, виділіть необхідний перемикач і у вікні **Properties** привласніть властивості **Value** значення ***False***.

Можна перемикач зробити недоступним. Для цього необхідно значення властивості **Enabled** (Доступний) установити в ***False***. У цьому випадку перемикач виділяється сірим кольором.

Щоб вибрати перемикач у групі, можна:

- *установити курсор на необхідному перемикачі й клацнути кнопкою миші;*
- *перемістити фокус на групу перемикачів, використовуючи для цього клавішу* ***<Tab>****. Потім за допомогою клавіш-стрілок вибрати необхідний перемикач;*
- *установити в програмному коді для властивості* ***Value*** *необхідного перемикача значення* ***True****;*

▪ *натиснути клавішу швидкого доступу для відповідного перемикача.*

3.7 Списки (ListBox)

Список дозволяє користувачеві обирати один або декілька елементів з перелічених. В будь-який час у список можна додавати нові елементи або вилучати існуючі.

Для створення списків **ListBox** використовується відповідна кнопка на панелі елементів управління. Вона створює у формі список, у якому елементи розташовані в одну або декілька колонок. У випадку, якщо елементи списку не вміщуються у створеному об'єкті, то в ньому з'являються смуги прокручування, розташовані знизу й/або з правої сторони.

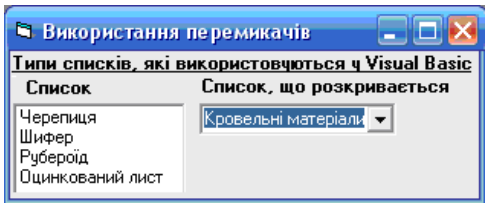


Рисунок 3.6 – Два типи списків, використовуваних у

На рисунку 3.6 показані списки, що являють собою елементи управління **ListBox** і **ComboBox**.

Основні властивості:

▪ **Name List –** здійснює формування списку. Назви пунктів списку вводяться в поле, що розміщене праворуч. Перехід на інший рядок здійснюється за допомогою комбінації клавіш **Ctrl+Enter**, якщо натиснути **Enter**, то вікно закриється. Редагування слів не припустимо;

▪ **Sorted** – за попереднім визначенням **False**. Якщо встановити **True**, то автоматично відбудеться сортування списків за алфавітом по зростанню, не враховуючи регістр символів;

▪ **Columns** – дозволяє керувати кількістю колонок у списку. Ця властивість може приймати значення:

0 –однорядковий список;

1–багаторядковий список з горизонтальним прокручуванням;

▪ **Style**:

0**– Standart** – за попереднім визначенням;

1**–Checkbox** – з'являться прапорці перед елементами списку.

Іноді для зручності потрібно, щоб з появою форми на екрані в списку за замовчуванням було виділено найбільш часто обиране з нього значення. Для установлення значення, обраного за замовчуванням, використовується властивість **Listindex,** що показує номер обраного елемента.

*При виборі першого елемента списку значення властивості дорівнює **0**, виходячи з того, що нумерація елементів*

*починається з **0**. Якщо обрано п'ятий елемент, значення властивості **ListIndex** буде дорівнювати **4**.*

Якщо у формі, що містить список **ListBox**, розташувати текстове поле, у ньому можна відобразити обране зі списку значення (рисунок 3.7).

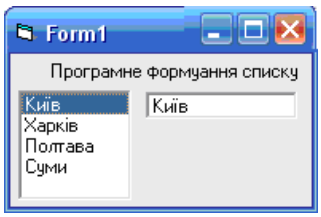


Рисунок 3.7 – Використання властивості **Listindex** для перегляду обраного зі списку елемента

Можна вибирати із списку кілька елементів. Для створення таких списків призначена властивість **Multiselect** (Множинний вибір). Вибір декількох елементів списку фіксується у властивості **Selected**. Обраному елементу списку відповідає значення **True** відповідного елемента властивості **Selected**, а іншим – **False**.

Найчастіше використовують методи:

- **Add Item** – для включення рядків у список;
- **Clear** – для очищення вікна повністю (видаляє всі рядки);
- **Remove Item** – для видалення рядків зі списків.

3.8 Комбіновані поля (ComboBox)

Використання **ListBox** пов'язане з однією потенційною проблемою, а саме вибір користувача обмежується рядками, що містяться в списках. Крім того, не можна прямо відредагувати рядок списку. Звичайні списки доцільні, якщо необхідно мати обмежений список і досить місця на формі.

Списки типу **ComboBox** сполучають можливості текстового поля й списку. Їх називають **списками, що розкриваються**, або **полями зі списком**. Списками, що розкриваються, їх називають тому, що для вибору значення зі списку, що має встановлені за замовчуванням властивості, спочатку необхідно відкрити список, натиснувши кнопку зі стрілкою, розташовану з правої сторони поля введення. Другу назву поля зі списком вони одержали тому, що цей список сполучає функції списку й поля введення. Іншими словами, можна не тільки вибирати дані зі списку, але й вводити в поле введення у верхній частині нові значення. Використання **ComboBox** дає можливість видавати великий обсяг інформації, заощаджуючи при цьому місце у формі.

Стилем оформлення списку типу **ComboBox** управляє властивість **Style** (рисунок 3.8):

0 – комбіноване поле, що розкривається. Схоже на стандартне текстове поле, зі стрілкою праворуч, при цьому користувач може вибирати рядок зі списку, а може ввести свій текст. Обране значення переноситься в текстове поле. Цей варіант зазвичай називається комбінованим полем.

1 – звичайне комбіноване поле. Являє собою різновид описаного; відмінність полягає в тому, що даний список постійно залишається відкритим. Якщо всі елементи в ньому не вміщуються, з'являється вертикальна смуга прокручування. Користувач може ввести значення в текстове поле, що розташовується у верхній частині списку, або вибрати необхідне значення, що буде перенесено в текстове поле.

2 – список, що розкривається. Набуває рис **ListBox**, зовні схожий на комбіноване поле, але користувач обмежений тільки рядками, що входять у список. Редагування не припустиме.

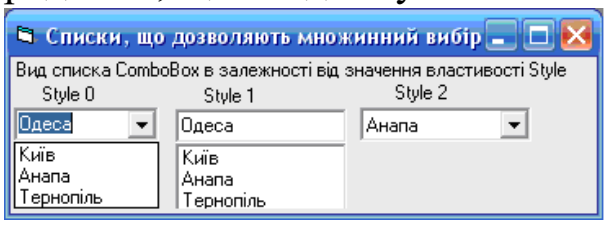


Рисунок 3.8 – Стилі оформлення списку типу **ComboBox**

Додавати, вилучати та одержувати доступ до елементів **ComboBox** під час розроблення можна так само, як і для списку **ListBox**.

Якщо для списку, що розкривається, не встановлене використовуване за замовчуванням значення, то з появою його на екрані в полі, призначеному для введення значення списку, відображається текст, що задається властивістю **Text** і який є іменем об'єкта. Якщо ви хочете, щоб це поле з появою списку на екрані було порожнім або містило заданий вами текст, виділіть властивість **Text** і в правому стовпці видаліть інформацію, залишивши поле порожнім, або введіть необхідний текст відповідно.

3.9 Смуги прокручування (VscrollBar, HscrollBar)

Якщо ви знайомі з документами програми Microsoft Word і іншими програмними продуктами, що працюють у середовищі Windows, то маєте уявлення про смуги прокручування. З ними ми також зустрічалися при роботі із багаторядковими текстовими полями й списками, у яких інформація не вміщувалася цілком у вікні перегляду.

Елементи управління **VScrollBar** і **HScrollBar** відрізняються від смуг прокручування, вбудованих у перелічені елементи, тому що існують самостійно й використовуються для управління введенням параметра, значення якого може змінюватися в деякому діапазоні (рисунок 3.9).

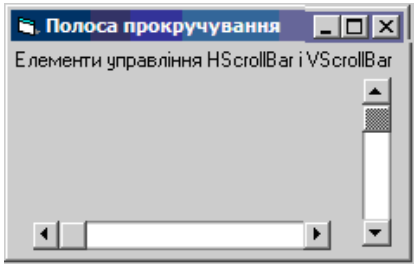


Рисунок 3.9 – Елементи управління **VScrollBar** і **HScrollBar**

Основні властивості, що характеризують елементи управління типу **VScrollBar** і **HScrollBar**, подані в таблиці 3.6.

Таблиця 3.6 – Властивості елементів управління VScrollBar і HScrollBar

Властивість	Призначення
LargeChange, SmallChange	Задають величини, на які буде зміщатися повзунок при клацанні кнопкою миші на смузі або стрілці прокручування
Min, Max	Задають діапазон, що вводиться зі допомогою смуги прокручування чисел
Value	Ціле число, що відповідає положенню повзунка на смузі прокручування

Після розміщення смуги прокручування у формі необхідно, використовуючи властивості **Min** і **Max**, задати діапазон значень, установлюваних за допомогою даного елемента управління.

Властивість **Value** (Значення) визначає поточне положення бігунка на смузі прокручування. Значення даних властивостей можуть бути тільки цілими числами в діапазоні від -32768 до +32767.

*При цьому не обов'язково, щоб значення, що задається властивістю **Min**, було менше значення властивості **Max**. Ви можете використовувати бігунок для відображення даних від більшого значення до меншого.*

Значення **Value** змінюється при переміщенні бігунка й клацанні мишею на смузі прокручування або на стрілках, розташованих по краях смуги. Для задання величини, на яку буде змінюватися значення **Value** при клацанні мишею на стрілках, що містяться по краях смуги прокручування, використовується властивість **SmallChange** (Малий зсув). За допомогою властивості **LargeChange** (Великий зсув) можна задати величину, на яку буде зміщатися бігунок при клацанні кнопкою миші на смузі прокручування. За замовчуванням обидві ці властивості мають

значення **1**. На практиці, властивість **SmallChange** використовують для більш плавної зміни значення властивості **Value**. Для властивості **LargeChange** встановлюють значення, рівне, приблизно, 10 % від діапазону зміни властивості **Value**.

Елементи управління типу **VScrollBar** і **HscrollBar** для відображення властивості **Value** використовують такі події (таблиця 3.7):

Таблиця 3.7 – Події елементів управління типу **VScrollBar** і **HScrollBar**

Подія	Призначення
Change	Подія настає після переміщення бігунка в момент відпускання кнопки миші або після клацання мишею в області смуги прокручування або на кнопках із зображеннями стрілок
Load форми	Дозволяє одержати початкове значення властивості Value при завантаженні форми
Scroll	Дозволяє одержати значення властивості Value при переміщенні бігунка до виникнення події Change

3.10 Контейнер (Frame)

Використовується для об'єднання інших елементів у групу, після чого поміщеними в нього об'єктами можна управляти як єдиним цілим.

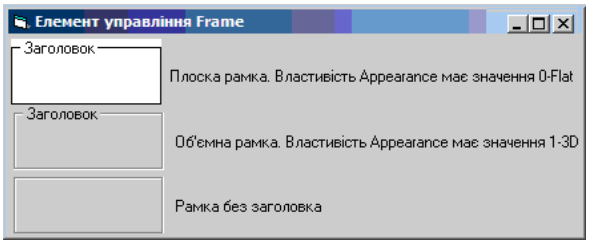


Рисунок 3.10 – Вид **Frame** залежно від його властивостей

Frame має вигляд рамки із заголовком (рисунок 3.10). Заголовок, розташовуваний у рамці, задається властивістю **Caption**.

*Якщо ви не хочете, щоб об'єкт **Frame** містив заголовок, установіть для значення цієї властивості порожній рядок.*

Об'єкт **Frame** є повноцінним контейнером і управляє загальними властивостями об'єктів, поміщених у рамку. Якщо, наприклад, зробити **Frame** невидимим, привласнивши

властивості **Visible** значення **False**, то всі об'єкти в рамці теж стануть невидимими.

*Елемент **Frame** можна використовувати для об'єднання в групу розміщених у формі й функціонально зв'язаних перемикачів. Для настроювання зовнішнього вигляду **Frame** застосовуються властивості, зазначені в таблиці 3.8.*

Таблиця 3.8 – Властивості елемента **Frame**

Властивість	**Призначення**
Appearance	Властивість може приймати значення 0-Flat або 1-3D, що задають плоский або об'ємний вигляд рамки
BorderStyle	Визначає, чи буде навколо групи присутня рамка. Може приймати значення 0-None (рамка і її заголовок відсутні) і 1-Fixed Single (рамка присутня)
Caption	Визначає текст, що розташовується в лівому верхньому куті рамки. Якщо із властивості видалити текст, то група буде об'єднана суцільною рамкою

*На рисунку 3.10 показано, який вигляд має **Frame** залежно від властивостей об'єкта.*

*Для **Frame** можна встановити різні кольори для фону й заголовка рамки, використовуючи властивості **BackColor** і **ForeColor** відповідно.*

3.11 Прапорець (CheckBox)

Використовується для створення незалежного дво- чи трипозиційного прапорця.

Прапорці дозволяють користувачеві дати відповідь на поставлене питання. У випадку позитивної відповіді користувач установлює прапорець, і він набуває вигляд квадрата, у якому розміщена галочка. При не встановленому прапорці він має вигляд порожнього квадрата, що позначає негативну відповідь на поставлене питання. Можливий ще один стан прапорця, при якому він недоступний. У цьому стані він має вигляд галочки на сірому фоні.

Прапорці можуть використовуватися у формі по одному або групами.

Щоб установити або скинути прапорець, можна використовувати швидкі клавіші. Для їхнього призначення необхідно вставити символ амперсанда (**&**) перед відповідною буквою у властивості **Caption** прапорця.

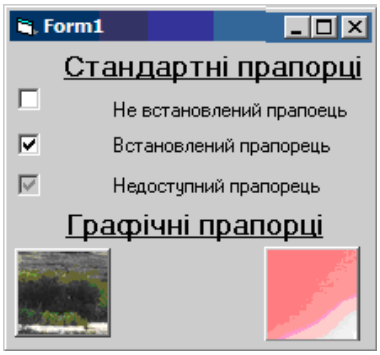


Рисунок 3.11 – Графічні прапорці мають вигляд утопленої або піднятої

Зовнішнім виглядом розміщеного у формі прапорця управляє властивість **Style**, що містить два значення. За замовчуванням установлене значення **Standard**, що припускає, що прапорець буде мати вигляд, описаний вище. При установленні значення **Graphical** прапорець буде мати вигляд утопленої або піднятої кнопки залежно від того, установлений або скинутий прапорець (рисунок 3.11).

Для задання графічного зображення на кнопці, що зображує прапорець, використовується властивість ***Picture****.*

Властивість **DownPicture** дозволяє задати графічне зображення, відображуване на кнопці при її натисканні.

Виходячи зі стану, у якому перебуває прапорець, його властивість **Value** може мати значення, вказані в таблиці 3.9:

Таблиця 3.9 – Властивості CheckBox

Властивість	**Опис**
Value	Стан прапорця :Grayded(2 – недоступний), Unchecked (0 – скинутий), Checked (1 – установлений)
MausePointer	Вигляд покажчика миші на об’єкті:Arrow (стрілка), Cross(хрест)

3.12 Лінії й контури

VB дозволяє розміщати у формі лінії, прямокутники, округлені прямокутники, кола, еліпси, використовувані для об'єднання в групу схожих за змістом об'єктів і поліпшення зовнішнього вигляду форми.

Лінія (Line)

Для додавання у форму лінії використовується кнопка **Line** на панелі елементів управління.

Установіть покажчик миші в те місце, де повинна починатися лінія, і, не відпускаючи кнопку миші, переміщайте його до одержання лінії потрібної довжини.

Властивість **BorderWidth** (Ширина границі) дозволяє задати товщину лінії. Лінія має за замовчуванням товщину **1**.

Для зміни товщини, заданої за замовчуванням, уведіть у правий стовпець властивості число, що буде визначати товщину лінії.

Для зміни розмірів лінії і її положення можна використовувати мишу й клавіатуру аналогічно тому, як це робиться для всіх елементів управління у формі. Крім цього, можна використовувати властивості **X1, X2, Y1** і **Y2**, що дозволяють змінювати ці параметри програмно. Властивості **X1** і **Y1** визначають відповідно горизонтальне й вертикальне положення лівого краю лінії, а властивості **X2** і **Y2** – правого краю.

*Властивість **BorderColor** (Колір границі) призначена для вказівки кольору лінії.*

*За допомогою властивості **BorderStyle** (Стиль границі) можна вказати стиль лінії, використовуючи значення з таблиці 3.10.*

Таблиця 3.10 – Значення властивості BorderStyle

Значення	Стиль лінії
0–Transparent	Лінія відсутня (має колір фону)
1–Solid	Тонка лінія
2–Dash	Штрихова лінія
3–Dot	Пунктирна лінія
4–Dash-Dot	Штрихпунктир
5–Dash-Dot-Dot	Штрихпунктир з подвійним штрихом
6–lnside Solid	Безперервна лінія

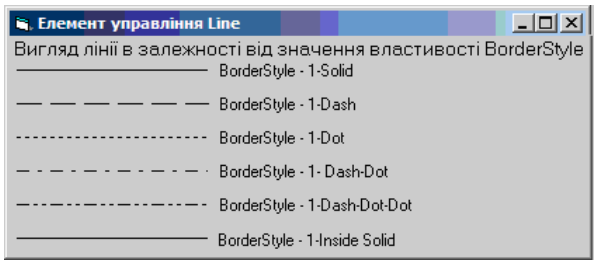


Рисунок 3.12 – Елемент управління **Line**

*Типи лінії залежно від значення властивості **BorderStyie** показані на рисунок 3.12. Для значення 0–**Transparent** лінія не наводиться, тому що її не видно у формі.*

Контури (Shape)

Для створення рамок різної форми використовується елемент управління **Shape**. Він є тільки графічним контейнером обведених контуром елементів управління, але не контейнером у справжньому значенні цього слова. Для додавання у форму контуру призначена кнопка **Shape** на панелі елементів управління. Вона дозволяє створювати у формі прямокутник, квадрат, овал, коло, прямокутник і квадрат з округленими кутами.

Щоб розмістити у формі одну з перелічених вище фігур, виконайте такі дії:

1 Натисніть кнопку **Shape** (Контур) на панелі елементів управління.

2 Встановіть покажчик миші в те місце, де повинен починатися об'єкт, і, не відпускаючи кнопку миші, перемістіть покажчик до одержання контуру необхідного розміру.

3 Для надання створеному об'єкту необхідної форми установіть одне із значень властивості **Shape**, поданих у таблиці 3.11.

Таблиця 3.11 – Значення властивості Shape

Значення	Контур
0–Rectangle	Прямокутник
1–Square	Квадрат
2–Oval	Овал
3–Circle	Коло
4–Rounded Rectangle	Прямокутник з округленими кутами
5–Rounded Square	Квадрат з округленими кутами

На рисунку 3.13 наведені стандартні контури, створювані у VB.

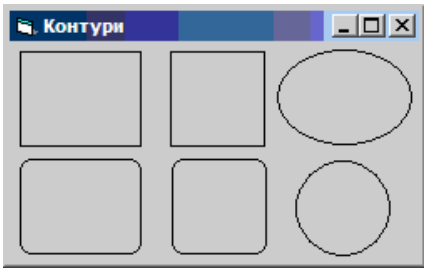


Рисунок 3.13 – Стандартні контури

Використовуючи властивість **BackStyle**, укажіть, чи буде створений об'єкт прозорим.За допомогою властивості **FillStyle** можна

задати візерунок заповнення, використовуючи значення з таблиці 3.12.

Таблиця 3.12 – Значення властивості FillStyle для контуру

Значення	Візерунок заповнення
0–Solid	Суцільне заповнення
1–Transparent	Незаповнене
2–Horizontal Line	Горизонтальне штрихування
3–Vertical Line	Вертикальне штрихування
4–Upward Diagonal	Штрихування по діагоналі зліва направо
5–Downward Diagonal	Штрихування по діагоналі справа наліво
6–Cross	Горизонтально-вертикальне штрихування
7–Diagonal Cross	Штрихування по діагоналі в обох напрямках

Властивість **FillColor** дозволяє задати колір візерунка заповнення об'єкта, а властивість **BackColor** (Колір фону) – колір фону.

Властивість **BorderStyle** призначена для задання стилю рамки об'єкта. Вона містить ті самі значення, що й для лінії.

Для надання контуру об'ємності використовуйте властивість **SpecialEffect**.

3.13 Таймер (Timer)

У VB існує елемент управління, що обробляє дані системного годинника. Цей об'єкт називається **таймером**. Його можна використовувати для виконання певних дій через заданий інтервал часу.

Для розміщення у формі таймера використовується кнопка **Timer** на панелі елементів управління форми. Об'єкт даного типу має такі властивості (таблиця 3.13):

Таблиця 3.13 – Значення властивості FillStyle для таймера

Властивість	Призначення
Interval (Інтервал)	Інтервал активізації об'єкта в мілісекундах. Може приймати значення від 0 до 64767 (від 0 до 64,8 секунди)
Enabled (Доступно)	Установлює режим роботи таймера. Якщо значення властивості дорівнює True (Істина), то таймер починає відраховувати час відразу ж після запуску форми. У противному випадку ви повинні запустити таймер по якій-небудь зовнішній події (наприклад, при натисканні на

	кнопку). Установка для властивості значення False припиняє операції таймера

Використання об'єкта-таймера розглянемо на прикладі форми, у якій через заданий інтервал часу на екран буде виводитися системний час комп'ютера. Для створення даної форми виконайте такі дії:

1 Відкрийте вікно для створення нового проекту.

2 Помістіть у форму мітку для відображення поточного системного часу. Створіть пояснювальний напис до мітки.

3 Для створення об'єкта-таймера натисніть кнопку **Timer** (Таймер) на панелі елементів управління й розташуйте його у формі.
Примітка – Розміщений у формі елемент управління Timer зображується у вигляді значка, показаного на рисунку 3.14. При запуску форми на виконання він стає невидимим користувачеві додатка.

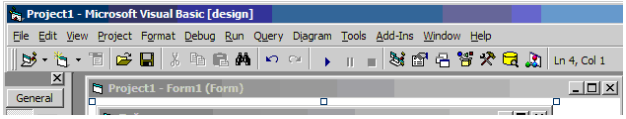
4 Визначте інтервал часу, через який необхідно робити відновлення часу у формі. Для цього скористайтеся властивістю Interval, значення якої задається в мілісекундах. Для відновлення часу до секунди введіть значення 1000.

5 Відкрийте вікно редактора коду й створіть просту процедуру, що привласнює властивості Caption мітки поточний час:

```
Private Sub Timer1 Timer()
Label1.Caption = Time
End Sub
```

6 Збережіть створену форму й запустіть проект на виконання. Ви побачите час, що оновлюється щосекунди.

Форму з розміщеними елементами керування та результати виконання проекту показано на рисунку 3.14.



3.14 Зображення (Image)

У VB для відображення у формі графіки використовуються елементи управління **Image** (Зображення) і **PictureBox** (Графічне вікно), створювані за допомогою кнопок **Image** і **PictureBox** на панелі елементів управління.

Об'єкт **Image** дозволяє розмістити у формі графічні зображення.

Для розміщення зображення у формі виконайте такі дії:

1. Натисніть кнопку **Image**.
2. Установіть покажчик миші в тім місці форми, де ви хочете розмістити зображення, і, утримуючи кнопку миші в натиснутому стані, перемістіть курсор по діагоналі так, щоб вийшла рамка необхідного розміру.
3. Відкрийте вікно властивостей **Properties** для створеного об'єкта.
4. Для задання імені графічного файлу призначена властивість **Picture** (Зображення). Виберіть дану властивість, а потім натисніть кнопку, розташовану в правому стовпці. Відкриється діалогове вікно **Load Picture**.

*Примітка – Щоб указати найменування графічного файлу, що задається об'єктом **Image**, під час виконання додатка, можна використовувати функцію **LoadPicture**.*

В якості зображень, що поміщаються в цей об'єкт, можна використовувати тільки файли певних типів.

5 Виберіть графічний файл в одному з таких форматів (таблиця 3.14):

Таблиця 3.14 – Формати графічних файлів

Опис файлу		**Тип файлу (розширення)**
Bitmap	Бітовий растровий файл	**Bmp**
	Незалежний растровий файл	**Dib**
Icon	Растрове зображення значка, має розмір 16x16 або 32x32 пікселів	**Ico**
Cursor	Растрове зображення курсору	**Cur**
Metafile	Метафайл, що являє собою зображення у вигляді закодованих ліній і образів	**Wmf**
	Розширений метафайл	**Emf**
	Растровий файл	**Gif**
	Растровий стислий файл	**jpg, jpeg**

Рисунок 3.15 – Розміщення графічного зображення у формі

6 Натисніть кнопку **Відкрити**. Зображення розміститься у формі (рисунок 3.15).

Зображення розміщується в об'єкті **Image** так само, як об'єкт у формі, тобто за допомогою властивості **Picture**. Однак, на відміну від розміщення зображення безпосередньо у формі, зображення в об'єкті **Image** пропорційно змінює свої розміри.

Для настроювання властивостей розміщеного у формі графічного об'єкта можна використовувати властивість **Stretch** (Розтягування). За замовчуванням установлене значення **False**, що вказує, що при зміні розмірів об'єкта розмір зображення залишається незмінним. Якщо для властивості **Stretch** установлене значення **True**, то при зміні розмірів об'єкта будуть відповідно змінюватися розміри зображення, що може призвести до перекручування зображення при невідповідності розмірів об'єкта й зображення.

Об'єкти **Image** розпізнають подію **Click**, що дозволяє використовувати їх як графічні кнопки управління, наприклад, при створенні панелі інструментів додатка.

При цьому необхідно враховувати, що натискання об'єкта ***Image*** *не призводить до його вдавлювання, спостережуваному при натисканні кнопки.*

Завантажувати графічне зображення в об'єкт **Image** можна в процесі розроблення додатка, а також програмно при його виконанні.

Щоб завантажити зображення при розробленні, можна використати графічне зображення, підготовлене в іншому додатку і перенесене у форму за допомогою буфера обміну Windows. Для цього:

1 Підготуйте в графічному редакторі зображення.

2 Скопіюйте його в буфер обміну Windows.

3 Перейдіть у програму VB.

4 Розмістіть у формі об'єкт **Image**, використовуючи однойменну кнопку на панелі елементів управління.

5 Виберіть елемент управління **Image**, щоб він став активний.

6 Вставте в нього зображення з буфера обміну, вибравши команду **Paste** меню **Edit** або натиснувши комбінацію клавіш **<Ctrl>+<V>.**

Для завантаження зображення в об'єкт **Image** під час виконання додатка використовується властивість **Picture** і функція **LoadPicture** такого вигляду:

Image1.Picture = LoadPicture ("ім'яФайлу")

де ім'я Файлу – це ім'я файлу із вказівкою повного шляху до нього. **Наприклад**, якщо графічний файл **Test_image.bmp** розташований у папці **\Sample** диска **C**, то функція буде мати такий вигляд:

Imagel.Picture = LoadPicture ("C:\Sample\Test_image.bmp")

Для того щоб очистити елемент управління **image** від розміщеного в ньому зображення, можна скористатися функцією **LoadPicture** такого вигляду:

Image1.Picture = LoadPicture

3.15 Графічне вікно (PictureBox)

PictureBox (Графічне вікно) має більш широкий набір властивостей і методів, ніж об'єкт **Image**.

Він може використовуватися для таких цілей:

- для відображення графічних зображень;
- як контейнер для інших елементів управління;
- у вигляді графічного вікна для виведення тексту, графічних елементів, анімації.

Для відображення графічних зображень за допомогою елемента **PictureBox** використовується властивість **Picture**, значенням якого є найменування графічного файлу. Об'єкт даного типу дозволяє відображати у формі графічні файли тих самих форматів, що й елемент управління **Image**.

Для завантаження зображення в об'єкт **PictureBox** під час виконання додатка застосовується властивість **Picture** і функція **LoadPicture** такого виду:

Picture1.Picture = LoadPicture ("ім'яФайлу")

Зображення в **PictureBox** завантажується повністю. При цьому, якщо воно більше розмірів **PictureBox**, то видно тільки

частину зображення, якщо менше, – то зображення розміщується в лівому верхньому куті. Якщо встановити для властивості **AutoSize** об'єкта **PictureBox** значення ***True***, то зображення буде пропорційно вписуватися в об'єкт, і при зміні розміру графічного зображення змінюються й розміри об'єкта. При установленні значення ***False*** усікається частина зображення, що не помістилася в об'єкт.

PictureBox можна використовувати як контейнер для об'єднання об'єктів у групи аналогічно об'єкту **Frame**. Розташовані в графічному вікні елементи будуть переміщатися разом з ним, що зручно при створенні панелей інструментів, рядка стану.

Примітка – Для створення рядка стану зручніше використовувати стандартний елемент управління ***Microsoft StatusBar*** *(Рядок стану*).

Для виведення в графічне вікно тексту призначений метод **Print**.

Програмне розміщення графічних елементів здійснюється з використанням методів **Circle** (Окружність), **Line** (Лінія), **Point** (Точка), **Pset** (Набір точок).

Метод **PaintPicture** дозволяє створювати в графічному вікні анімацію.

Примітка – Щоб об'єкт ***PictureBox*** *можна було використовувати для виведення графічних елементів і їхнього перевиведення при зміні розмірів графічного вікна, необхідно встановити значення його властивості* ***AutoRedraw*** *в* ***True****.*

СПИСОК ЛІТЕРАТУРИ

1 Visual Basic 6.0. – С.Пб.: BHV Питер, 1999. – 992с.

2 Х. Антони Синтес. Освой самостоятельно объектно-ориентированное программирование за 21 день. –М.: Вильямс, 1994.

3 Ален И. Голуб. С и С++. Правила программирования. – М.: БИНОМ, 1996. – 272 с.

4 Керниган Б., Пайк Р. Практика программирования. – С.Пб.: Невский диалект, 2001. – 381 с.

5 Браун С.Visual Basic 6. Учебный курс.– С.Пб.: Питер, 2006. – 574 с.

6 Бутенко В.М., Меркулов В.С., Казанко О.В., Чаленко О.В. Основи програмування мовами високого рівня: Навч. посіб. – Харків: УкрДАЗТ, 2009. – 206 с.

7 Волченков Н.Г. Программирование на Visual Basic 6: В 3 ч. – М.: ИНФРА-М, 2000. – 280 с.

8 Иванова Г.С. Основы программирования: Учеб. для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. – 416 с.

9 Иванова Г.С. Технология программирования: Учеб. для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002. – 320 с.

10 Ван Тассел Д. Стиль, разработка, эффективность, отладка и испытание программ. – М.: Мир, 1985. – 332 с.

11 Грэхем Иан. Объектно-ориентированные методы. Принципы и практика. – 3-е изд. – М.: Вильямс, 2004.

12 Информатика: Задачник-практикум в 2 т / Под ред. И.Г. Семакина, Е.К. Хеннера. – М.: Лаборатория базовых знаний, 2000.

13 Інформатика, комп’ютерна техніка, комп’ютерні технології: Посібник / За ред. О.І. Пушкаря. – К.: Академія, 2001.

14 Зелковиц М., Шоу А., Гэннон Дж. Принципы разработки программного обеспечения. – М.: Мир, 1982. – 368 с.

15 Максимов Н.А. Азбука программирования на Visual Basic: Практикум. – Чебоксары, 2007. – 64 с.

16 Меркулов В.С., Гончаров В.О., Бізюк І.Г., Бутенко В.М., Головко О.В. Основи алгоритмізації базових обчислювальних процесів: Навч. посіб. – Харків: УкрДАЗТ, 2008. – 163 с.

17 Вирт Н. Систематическое программирование. – М.: Мир, 1977. – 183 с.

18 Райтингер М., Муч Г. Visual Basic 6.0. – К.: BHV, 2000. – 288 с.

19 Семакин И.Г., Вараксин Г.С. Информатика: Структурированный конспект базового курса. – М.: Лаборатория базовых знаний, 2001.

20 Дал У., Дейкстра Э., Хоор К. Структурное программирование. – М.: Мир, 1973. – 247 с.

21 Філіппенко І.Г., Гончаров В.О., Меркулов В.С. Програмування інженерно-технічних задач в середовищі QBasic: Конспект лекцій з дисципліни «Обчислювальна техніка і програмування». – Харків: УкрДАЗТ. – Ч. 3. – 2007. – 134 с.

22 Дейкстра Э. Дисциплина программирования. – М.: Мир, 1978. – 275 с.

23 Йодан Э. Структурное проектирование и конструирование программ. – М.: Мир, 1979. – 415 с.

24 [http://www.alexsoft.ru Visual Basic 6.0.].

25 [http://www.ctc.msiu.ru/materials/Book/node82.html Основные концепции ООП].

26 [http://www.weblibrary.biz/c-sharp/princypy].

ДОДАТОК А

Таблиця А.1 – Команди меню File

Команда	Призначення
New Project (Новий проект)	Створює новий проект
Open Project (Відкрити проект)	Відкриває існуючий проект
Add Project (Додати проект)	Додає проект у групу для паралельної роботи над декількома проектами, копіювання форм із проекту в проект або його налагодження
Remove Project (Виключити проект)	Видаляє проект із групи проектів
Save Project (Зберегти проект)	Зберігає поточний проект, включаючи всі модулі і форми
Save Project As (Зберегти проект як)	Зберігає поточний проект, включаючи всі модулі і форми, під іншим ім'ям
Save <ім'я форми> (Зберегти форму)	Зберігає поточну форму з колишнім ім'ям
Save <ім'я форми> As (Зберегти форму як)	Зберігає форму під іншим ім'ям
Print (Друк)	Відкриває діалогове вікно Print
Print Setup (Настроювання друку)	Відкриває діалогове вікно настроювання друку
Make <ім'я проекту>.exe (Створити exe-файл проекту)	Створює виконуваний файл поточного проекту
Make Project Group (Створити групу проектів)	Створює групу проектів
Exit (Вихід)	Здійснює вихід з VB. У тому випадку, якщо у вікні програми містяться незбережені результати роботи, з'являться запити про необхідність збереження проектів, форм і інших файлів

Таблиця А.2 – Команди меню **Edit**

Команда	Призначення
1	**2**
Undo (Скасувати)	Скасовує попередню команду. Не всі команди можуть бути скасовані. Меню доступно тільки у випадку, якщо є що скасовувати
Redo (Повторити)	Відновлює попередню команду
Cut (Вирізати)	Видаляє виділений текст або об'єкт і розташовує його у Windows Clipboard (буфер)
Copy (Копіювати)	Копіює виділений текст або об'єкт, розташовує його у Windows Clipboard. Виділений текст або об'єкт залишається незмінним
Paste (Вставити)	Вставляє текст або об'єкт із Windows Clipboard у поточний модуль або форму
Remove (Перемістити)	Переміщає виділене
Delete (Видалити)	Видаляє виділене
Delete Table from Database (Видалити таблицю з бази даних)	Видаляє таблицю з бази даних
Select All (Виділити все)	Виділяє весь текст у модулі або всі об'єкти у формі
Select All Columns (Виділити всі стовпчики)	Виділяє всі стовпчики
Table: Set Primary Key (Таблиця: Установити первинний ключ)	Визначає первинний ключ таблиці
Table: Insert Column (Таблиця: Вставити поле)	Вставляє поле в таблицю
Table: Delete Column (Таблиця: Видалити поле)	Видаляє поле з таблиці
Find (Знайти)	Викликає діалогове вікно пошуку
Find Next (Знайти наступне)	Шукає наступне значення, задане для пошуку
Replace (Замінити)	Шукає значення, задане для пошуку, і заміняє його новим значенням
Go To Row: First (Перейти на запис: Перша)	Установлює покажчик на перший запис

Продовження таблиці А.2

1	2
Go To Row: Last (Перейти на запис: Остання)	Установлює покажчик на останній запис
Go To Row: Next (Перейти на запис: Наступна)	Установлює покажчик на такий запис
Go To Row: Previous (Перейти на запис: Попередня)	Установлює покажчик на попередній запис
Go To Row: Row (Перейти на запис: Запис)	Установлює покажчик на запис із заданим номером
Go To Row: New (Перейти на запис: Нова)	Вставляє новий запис
Bookmarks (Закладки)	Відкриває підменю з пунктами для переміщення, видалення або переходу до закладок, що раніше були розташовані у модулі

Таблиця А.3 – Команди меню View

Команда	Призначення
1	**2**
Code (Код-Програма)	Відкриває вікно редактора 73рограммного коду та активізує Code Window для відображення вихідного коду VB, асоційованого з обраним модулем або формою
Object (Об'єкт)	Відкриває вікно конструктора форм
Object Browser (Браузер об'єктів)	Відкриває вікно браузера об'єктів
Immediate Window (Вікно безпосереднього виконання)	Відкриває вікно, призначене для уведення й безпосереднього виконання команд
Locals Window (Вікно Локальних змінних)	Відкриває вікно, призначене для перегляду значень змінних
Watch Window (Вікно Спостереження)	Відкриває вікно перегляду значень виразів (контрольні значення)
Project Explorer (Провідник проекту)	Відкриває вікно провідника проектів
Properties Window (Вікно властивостей)	Відкриває вікно властивостей об'єктів Properties

Продовження таблиці А.3

1	2
Form Layout Window (Вікно макета форми)	Відкриває вікно макета форми
Toolbox (Панель інструментів)	Відображає панель елементів управління
Data View Window (Вікно перегляду даних)	Відкриває вікно перегляду даних
Color Palette (Колірна палітра)	Відображає колірну палітру
Toolbars (Панелі інструментів)	Відкриває меню, призначене для відображення у вікні програми VB панелей: Debug, Editor, Form Edit, Standard

Таблиця А.4 – Команди меню **Project**

Команда	Призначення
Add Form (Додати форму)	Додає у вікно проекту форму
Add MDI Form (Додати форму вікна **MDI**)	Додає MDI-Форму
Add Module (Додати модуль)	Додає програмний модуль
Add Class Module (Додати клас)	Додає користувальницький клас
Add User Control (Додати елемент управління)	Додає користувальницький елемент управління
Add Property Page (Додати сторінку властивостей)	Додає стандартну форму настроювання властивостей
Add WebClass (Додати **Web**-Клас)	Додає Web-Клас
Add Data Report (Додати звіт)	Додає в проект звіт
Add DHTML Page (Додати сторінку **DHTML**)	Додає сторінку DHTML
Add Data Environment (Додати середовище даних)	Додає середовище бази даних
Add File (Додати файл)	Додає файл
Remove (Виключити)	Виключає файл із проекту
References (Посилання)	Викликає вікно включення посилань проекту на бібліотеки
Components (Компоненти)	Відкриває діалогове вікно Components, що дозволяє додати на панель елементів управління компоненти для їхнього подальшого включення в проект додатка
Project Properties (Властивості проекту)	Відкриває вікно властивостей проекту Project Properties

Таблиця А.5 – Команди меню Format

Команда	**Призначення**
Align (Вирівняти)	Відкриває підменю команд, що дозволяють вирівнювати обрані об'єкти у формі по відношенню один до одного. Тут можна вирівнювати об'єкти по верхній/нижній, правій/лівій границях, по центру або середині створюваного об'єкта
Make Same Size (Установити подібний розмір)	Відкриває меню, що містить команди, що управляють розмірами об'єктів у формі, змінювати розмір виділених об'єктів до розміру зазначеного об'єкта
Horizontal Spacing (Горизонтальний інтервал)	Відкриває підменю команд, що дозволяють встановлювати горизонтальний інтервал для обраних об'єктів. Тут можна встановлювати рівномірний горизонтальний інтервал, зменшувати або збільшувати його або видаляти будь-який горизонтальний інтервал між об'єктами
Vertical Spacing (Вертикальний інтервал)	Відкриває підменю команд, що дозволяють встановлювати вертикальний інтервал для обраних об'єктів. Тут можна встановлювати рівномірний вертикальний інтервал, зменшувати або збільшувати його або видаляти будь-який вертикальний інтервал між об'єктами
Center in Form (Центрування у формі)	Центрує об'єкти у формі. Відкриває підменю команд, що дозволяють змінювати положення обраних об'єктів, щоб вони були центровані у формі, горизонтально або вертикально. Відкриває підменю команд, що дозволяють автоматично розташовувати командні кнопки у формі в ряд з рівним інтервалом по нижньому або правому краю форми
Order (Порядок)	Направляє обраний об'єкт у нижній або верхній шар форми. Відкриває підменю команд, що дозволяють змінювати упорядкування зверху вниз (називане *z-order)* об'єктів у формі, що перекриваються. Команда Order може забезпечити, наприклад, появу текстового вікна завжди поверх графічного об'єкта у формі
Size to Grid (Вирівняти розмір по сітці**)**	Одночасно змінює ширину і висоту об'єкта до найближчих міток сітки. При розробці форм Редактор VB відображає у формі сітку, щоб було легше розташовувати і змінювати розміри об'єктів у формі

Таблиця А.6 – Команди меню **Debug**

Команда	**Призначення**
Step Into (Крок із заходом у процедури)	Здійснює покрокове виконання процедури (по одному оператору щоразу), включаючи також викликувані нею процедури
Step Over (Крок без заходу в процедури)	Здійснює покрокове виконання процедури без трасування викликуваних нею процедур
Step Out (Крок з виходом із процедури)	Виконання програми здійснюється до виходу з поточної процедури
Add Watch (Додати вікно **Watch-**додати контрольне значення)	Відкриває вікно **Add Watch** (Додати спостереження) для введення виразів, за значеннями яких буде вестися спостереження під час виконання вихідного коду VB
Edit Watch (Відкрити вікно **Edit Watch-**змінити контрольне значення**)**	Відкриває діалогове вікно **Edit Watch** для редагування або видалення виразів, за значеннями яких ведеться спостереження
Quick Watch (Відкрити вікно **Quick Watch)**	Відкриває вікно **Quick Watch** для перегляду значення обраного виразу
Toggle Breakpoint (Установити точку останова)	Установлює точку останова, де необхідно зупинити виконання коду
Clear All Breakpoints (Скасувати всі точки останова)	Видаляє всі точки останова
Set Next Statement (Визначити наступний оператор)	Установлює наступний оператор і змінює звичайне виконання коду шляхом вказівки вручну наступного
Show Next Statement (Показати наступний оператор)	Показує наступний оператор
Run to Cursor (Виконати до поточної позиції)	Виконує оператори вихідного коду VB, починаючи від оператора, що виконується в даний момент, до поточної позиції курсору

Таблиця А.7 – Команди меню **Run**

Команда	**Призначення**
Start (Запустити)	Запускає додаток на виконання
Start with Full Compile (Запустити з повною компіляцією)	Запускає додаток на виконання з повною компіляцією
Break (Призупинити)	Зупиняє виконання додатка
End (Кінець)	Припиняє виконання додатка
Restart (Запустити знову)	Перезапускає додаток

Таблиця А.8 – Команди меню **Query**

Команда	**Призначення**
1	**2**
Run (Запустити)	Виконує запит
Clear Results (Очистити результати)	Очищає результати запиту
Verify SQL Syntax (Перевірити синтаксис **SQL**)	Перевіряє синтаксис запиту
Group By (Групувати)	Групує дані в запиті
Change Type: Select (Модифікація: Вибрати)	Створює запит для вибірки даних
Change Type: Insert (Модифікація: Вставити)	Створює запит, призначений для додавання даних у таблицю, використовуючи значення інших таблиць
Change Type: Insert Values (Модифікація: Вставити значення)	Створює запит, призначений для додавання даних у таблицю, використовуючи задані значення
Change Type: Update (Модифікація: Змінити)	Створює запит, призначений для зміни даних
Change Type: Delete (Модифікація: Видалити)	Створює запит, призначений для видалення даних
Change Type: Make Table (Модифікація: Створити таблицю)	Створює нову таблицю на основі результатів вибірки
Add to Output (Додати поля)	Додає поля в список полів формованої вибірки
Sort Ascending (Сортування по зростанню)	Призначає сортування по зростанню
Sort Descending (Сортування по убуванню)	Призначає сортування по убуванню

Продовження таблиці А.8

1	2
Remove Filter (Видалити фільтр)	Видаляє фільтр
Select All Rows From <Table A > (Вибрати всі записи з першої таблиці)	Створює зовнішнє з'єднання, що вибирає всі записи з першої таблиці, навіть якщо вони відсутні в другій таблиці
Select All Rows From <Table B > (Вибрати всі записи із другої таблиці)	Створює зовнішнє з'єднання, що вибирає всі записи із другої таблиці, навіть якщо вони відсутні в першій таблиці

Таблиця А.9 – Команди меню **Diagram**

Команда	**Призначення**
New Text Annotation (Новий текст пояснення)	Додає пояснювальний текст
Set Text Font (Установити шрифт)	Установлює шрифт
Add Related Tables (Додати зв'язану таблицю)	Додає зв'язану таблицю
Show Relationship Labels (Показати зв'язку)	Показує найменування зв'язків
Modify Custom View (Змінити подання)	Змінює вид відображення таблиць у діаграмі
View Page Break (Показує посторінкову розбивку)	Показує розбивку на сторінки
Recalculate Page Break (Перелічити посторінкову розбивку)	Перелічує розбивку на сторінки
Arrange Selection (Вирівнювати обрані)	Вирівнює обрані таблиці
Arrange Tables (Вирівнювати всі таблиці)	Вирівнює всі таблиці
Autosize Selected Tables (Авторозмір виділених таблиць)	Автоматично встановлює розмір обраних таблиць

*Таблиця А.10 – Команди меню **Tools***

Команда	**Призначення**
Add Procedure (Додати процедуру)	Додає процедуру
Procedure Attributes (Атрибути процедури)	Установлює атрибути процедури
Menu Editors (Редактор меню)	Викликає вікно редактора меню
Options (Параметри)	Відкриває діалогове вікно **Options** для настроювання параметрів програми (проекту) VB
SourseSafe (Сховище)	Управляє сховищем проектів

Таблиця А.11 – Команди меню **Add-Ins**

Команда	**Призначення**
Visual Data Manager (Менеджер баз даних)	Запускає менеджера управління базою даних
Add-In Manager (Менеджер надбудов)	Відкриває діалогове вікно Add-In Manager, що дозволяє додати надбудову у VB

Таблиця А.12 – Команди меню **Window**

Команда	**Призначення**
Split (Розділити)	Розділяє вікно
Tile Horizontally (Горизонтально)	Розміщає вікна горизонтально
Tile Vertically (Вертикально)	Розміщає вікна вертикально
Cascade (Каскадом)	Розміщає вікна каскадом
Arrange Icons (Упорядкування значків)	Змінює розміщення значків вікон

Таблиця А.13 – Команди меню **Help**

Команда	**Призначення**
Context (Зміст)	Викликає довідкову систему з активною вкладкою **Зміст**
Index (Покажчик)	Викликає довідкову систему з активною вкладкою **Індекс**
Search (Пошук)	Викликає довідкову систему з активною вкладкою **Пошук**
Technical Support (Технічна підтримка)	Викликає довідкову систему з інформацією про службу технічної підтримки **Microsoft**
Microsoft on the Web (Microsoft у Web)	Викликає сторінку **Microsoft** в **Internet**
About Visual Basic (про Visual Basic)	Викликає довідкову інформацію про пакет **Visual Basic 6**