Бондаренко О.О, Ковшун М.І.,
Пилипчук О.П., Шестопалов Є.А.

Інформатика

Третій рік – єдиний курс

11 клас

Рівень стандарту
Академічний рівень (половина)

Навчальний посібник
(використовується з «Робочим зошитом»)

Шепетівка
«Аспект»
2011

УДК.004.451 (07)
ББК.32.973.26-018.2я7
Б52

Пробний випуск для апробації в умовах навчального процесу.

Зауваження та пропозиції щодо змісту навчального матеріалу посібника надсилайте на адресу: aspekt@aspekt.in.ua

Після випробування в різних регіонах України та обробки зауважень виправлене і доповнене видання посібника буде подано на розгляд експертної комісії МОН України для одержання відповідного грифу.

Бондаренко О.О., Ковшун М.І., Пилипчук О.П., Шестопалов Є.А.

Б52 Інформатика. Третій рік – єдиний курс. 11 клас. Рівень стандарту, Навчальний посібник / – Шепетівка: «Аспект», 2011 – 192 с.

ISBN 978-966-2017-34-2

Навчальний посібник рекомендується для учнів 11-х класів загальноосвітніх навчальних закладів усіх профілів рівня стандарту та академічного рівня (половина), які розпочали вивчення інформатики з 9-го класу.

Навчальний посібник відповідає відповідає вимогам програми «Інформатика. Єдиний базовий курс. 9-11 класи» (автори: Завадський І.О., Дорошенко Ю.О., Пилипчук О.П., Шестопалов Є.А.).

Розрахований на використання з робочим зошитом «Інформатика. 11 клас. Рівень стандарту. Академічний рівень (половина)» /Бондаренко О.О., Ковшун М.І., Пилипчук О.П., Шестопалов Є.А. – «Аспект», 2011.

УДК.004.451 (07)
ББК.32.973.26-018.2я7

ISBN 978-966-2017-34-2

Передмова

Навчальний посібник розрахований на учнів 11-х класів загальноосвітніх навчальних закладів усіх профілів рівня стандарту і частково академічного рівня, які почали вивчати інформатику з 9-го класу.

Поданий у посібнику навчальний матеріал для 11-го класу відповідає чинній програмі «Інформатика. Єдиний базовий курс. 9–11 класи» (автори Завадський І.О., Дорошенко Ю.О., Пилипчук О.П., Шестопалов Є.А.)

Основною структурною особливістю єдиного курсу інформатики є уніфікація змісту навчального матеріалу та вимог до навчальних досягнень рівня стандарту та академічного рівня.

1. Базові поняття програмування – 11 год.;
2. Основи програмування – 16 год.;
3. Растрова комп'ютерна графіка та анімація – 5 год.;
4. Бази даних. Системи керування базами даних – 8 год.;
5. Тривимірна комп'ютерна графіка та анімація – 8 год.;
6. Автоматизоване створення та публікація веб-ресурсів – 7 год.;
7. Мова гіпертекстової розмітки – 4 год.;
8. Колективна розробка проекту – 4 год.

Із наведеного списку навчальних тем академічного рівня, на вивчення яких відводиться 2 год. на тиждень, рівень стандарту (1 год. на тиждень) передбачає вивчення тем 1, 3, 4 і 6.

Для академічного рівня до цих тем додаються теми 2, 5, 7 і 8.

Такий підхід дозволяє застосовувати єдині методики викладання та навчально-методичне забезпечення для шкіл всіх профілів і рівнів.

Кожен параграф посібника розрахований на вивчення протягом одного уроку і має структуру, що відповідає санітарним нормам: теоретичний матеріал – 20 хв.; робота за комп'ютером – 25 хв.

Для поточного закріплення нового матеріалу та вироблення навичок на кожному уроці передбачено виконання практичних робіт за комп'ютером тривалістю до 25 хв. Тематичні роботи виконуються на окремих уроках після формування і закріплення теоретичних знань та практичних навичок, здобутих протягом кількох уроків, які складають розділ.

Практичні та тематичні роботи вміщені в робочий зошит «Інформатика. 11 клас. Рівень стандарту. Академічний рівень (половина)» / Бондаренко О.О., Ковшун М.І., Пилипчук О.П., Шестопалов Є.А. – Шепетівка: «Аспект», 2011. – 52 с.

Зміст

1. Базові поняття програмування

1.1. Принципи роботи у середовищі візуальної розробки програм

В цьому розділі ви познайомитесь з основами проектування та розробки програм для комп'ютера. Програмування – це процес створення комп'ютерних програм за допомогою мов програмування. Програмування поєднує в собі елементи мистецтва, математики й інженерної науки.

Програма та мова програмування

Програма *– впорядкована послідовність команд для комп'ютера, виконання якої реалізує алгоритм розв'язування певної задачі.*

Команди в програмі (програмному коді) записуються мовою програмування.

Мова програмування *– це система позначень, яка використовується для запису алгоритмів для реалізації (виконання) їх за допомогою комп'ютера.*

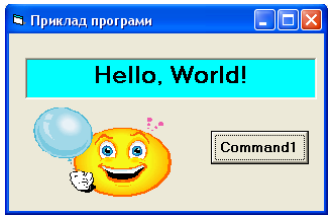
Класифікація мов програмування

Таблиця 1.1

Машинний код	Набір двійкових кодів для роботи центрального процесора певного типу
Мова асемблера	Мова для запису машинних кодів процесора зрозумілими людині символами – мнемоніками
Мова програмування високого рівня	Мова, що максимально наближена до людської мови, використовує для запису команд звичайні слова або їх скорочення (як правило, англійською мовою). Програма мовою високого рівня дозволяє формулювати завдання для комп'ютера у звичній для людини формі
Візуальне програмування	Системи програмування, які використовують графічний інтерфейс. Завдяки цьому програма створюється шляхом проектування макету в графічному вигляді з використанням готових елементів керування

Приклад 1.1. Приклад програми на мові **Visual Basic 6.0**.

```
Privat Sub Command1_Click()
    Label1.Caption = "Hello, World!"
End Sub
```



На малюнку наведено можливий вигляд вікна програми, де після натискання кнопки виконується наведений програмний код.

Середовище програмування Visual Basic

Система програмування *– це комп'ютерна система, призначена для створення програм.*

До складу системи програмування входять: текстовий редактор для введення та редагування тексту програми, компілятор, налагоджувач програми тощо. Призначення цих елементів розглянемо далі.

Інтегроване середовище програмування **Visual Basic IDE** (*Integrated Development Environment*) включає набір меню, панелей і вікон, службові програми, довідкову систему, що у сукупності утворюють «робоче місце» програміста.

IDE *– це «робоче місце» програміста, засобами якого створюються програми.*

Середовище програмування **Visual Basic IDE** є інтегрованим, тому що воно дозволяє виконувати всі дії, необхідні при розробці програмного продукту: проектування і опис складових частин програми, редагування програмного коду, компіляцію усіх елементів програми у виконуваний файл, відлагодження програми.

Запуск Visual Basic

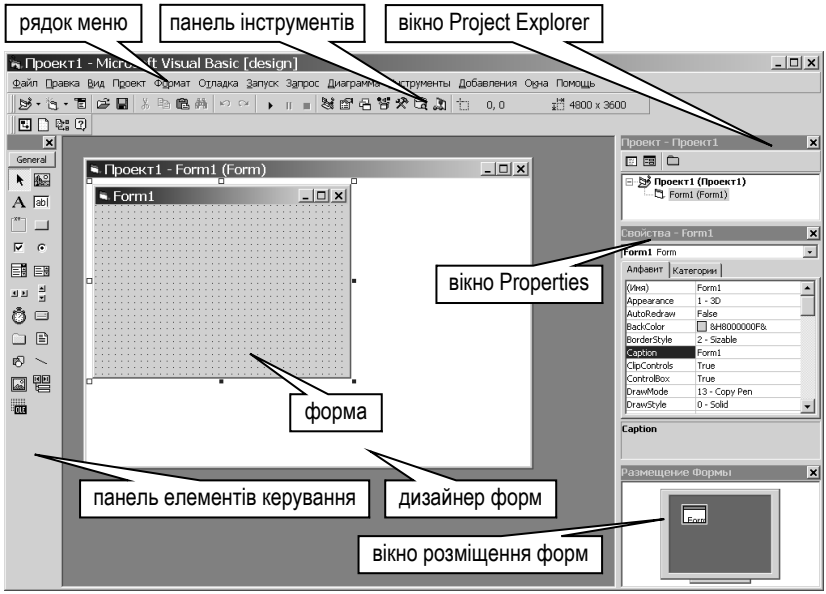
Для запуску **Visual Basic**:

- клацніть на кнопці Пуск на панелі задач Windows;
- в меню, що відкрилося, вкажіть на рядок *Программы*, потім перейдіть на рядок *Microsoft Visual Studio 6.0*;
- клацніть на рядку *Microsoft Visual Basic 6.0*. З'явиться діалогове вікно *New Project* (*Новий проект*), де потрібно вказати тип створюваного проекту. Щоб створити стандартну прикладну Windows-програму (застосунок), яка відкривається і працює у вікні, виберіть *Standard EXE*;
- натисніть кнопку *Open* (*Відкрити*).

Після виконання цих дій ми потрапляємо в середовище розробки **Visual Basic IDE**. При створенні нового проекту на екрані з'являється порожня форма – заготовка вікна майбутньої програми.

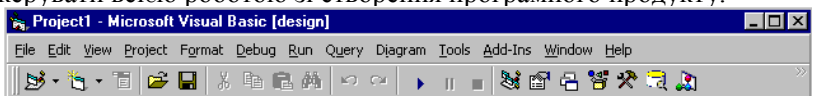
Основні об'єкти середовища IDE

Запустіть Visual Basic. Зверніть увагу на вікна, що з'явилися на екрані. ***Головне вікно*** використовується для керування створенням проекту і запуску розроблюваних програм.



У рядку заголовка головного вікна вказується ім'я проекту і поточний режим роботи Visual Basic: [*design*] (розробка), [*break*] (переривання), [*run*] (виконання).

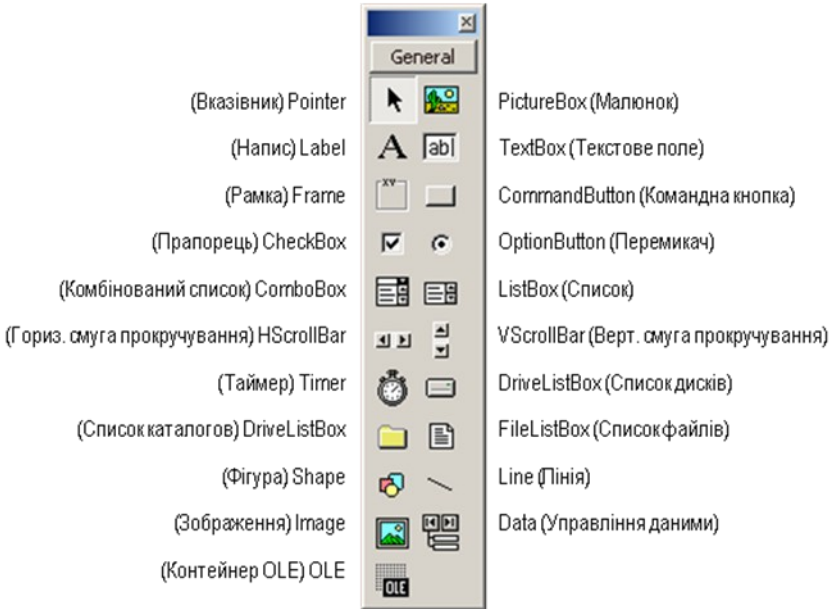
Під рядком заголовка знаходиться ***Головне меню***, необхідне для доступу до різних функцій Visual Basic. Меню дозволяє керувати всією роботою зі створення програмного продукту.



Під головним меню знаходиться ***Панель інструментів*** з кнопками, які призначені для швидкого доступу до деяких команд меню. Якщо ви помістите вказівник на кнопку панелі інструментів, то приблизно через секунду з'явиться спливаюча підказка про призначення даної кнопки.

Дизайнер форм (*Form Designer*). ***Форма*** (*form*) є основним елементом керування при розробці проекту в Visual Basic. Форми – основні будівельні блоки програм на Visual Basic – представлені у вікні дизайнера форм. З вікна *Form* починається розробка вашої програми. Воно призначене для редагування форм, тобто додавання та вилучення з них різних елементів керування.

Кожна відкрита форма має своє вікно дизайнера форм, яке у середовищі розробки зазвичай розташоване у лівому верхньому куті робочого поля головного вікна. Його можна переміщати в межах робочого поля, збільшувати, зменшувати.



Елементи керування (*Controls*), розміщені на формі, забезпечують зручну взаємодію користувача з комп'ютером. Дії користувача спричиняють ***події*** (*events*) елементів керування, програмна обробка яких дозволяє керувати роботою комп'ютера.

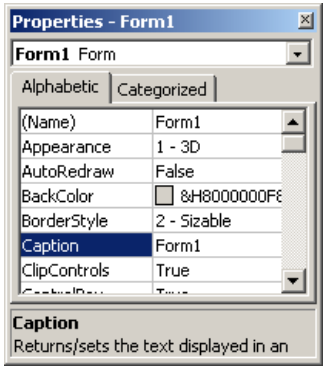
Якщо на екрані не видно вікна дизайнера форм, необхідно вибрати команду меню View / Object.

Панель елементів керування (*Toolbox*). Панель *ToolBox* використовується для додавання на форми проекту різних елементів.

Тут вибирають об’єкти, які потрібно помістити на форму. Наприклад, щоб намалювати лінію, слід клацнути кнопку *Line*. Потім помістити вказівник на те місце форми, де повинна починатися лінія, і, натиснувши ліву кнопку, перетягти вказівник до кінцевої точки лінії, після чого відпустити кнопку.

Інтерфейс користувача *– це сукупність засобів взаємодії програми з користувачем.* ***Компоненти*** *Visual Basic – це готові елементи інтерфейсу, які ви можете застосовувати у своїх програмах.*

У вікні ***Провідник проекту*** (*Project Explorer*) відображаються ті проекти, над якими ви працюєте в даний момент, а також структура кожного з них.



Вікно властивостей (*Properties Window*) призначене для перегляду і встановлення початкових значень деяких параметрів (властивостей) елементів керування. Список, що розкривається, у верхній частині вікна містить назви усіх елементів керування, розташованих на формі, з якою ви працюєте. Перелік властивостей обраного об’єкта (елемента керування) може бути поданий у двох виглядах: *Alphabetic* (За алфавітом) і *Categorized* (За категоріями).

Один з об’єктів завжди є «поточним об’єктом»: саме його властивості перелічені у вікні властивостей. Об’єкт стає поточним, коли на ньому клацнути. Розпізнати це найчастіше можна за обрамленням чи за яким-небудь іншим маркуванням.

Для настроювання властивостей об’єкта призначене вікно властивостей Properties.

Якщо вікно властивостей відсутнє на екрані, то необхідно викликати команду меню *View* ⇨ *Properties Window* або натиснути ***F4***.

Вікно розміщення форм (*Form Layout*) призначене для точного позиціювання форми на екрані при виконанні програми. Щоб задати положення форми, її зображення у вікні розміщення форм за допомогою миші переміщують в потрібне місце.

Поняття проекту

У **Visual Basic** ***проект*** – це сукупність файлів, що створюються в процесі розробки програми (див. таблицю 1.2). Крім того, проектом називають і саму програму у процесі її розробки.

У файлі проекту (наприклад, *Project1.vbp*) міститься інформація про файли програми: список усіх використовуваних програмою файлів, назва проекту, конфігурація *IDE* для роботи над даним проектом і т.п. Проект має модульну структуру. Модулі бувають трьох типів: модуль форми, стандартний модуль та модуль класу. Ми в даному курсі будем створювати проекти, що найчастіше складатимуться з однієї форми, і будуть містити тільки модуль форми. Файл модуля форми (наприклад, *Form1.frm*) містить оголошення змінних, констант, типів даних, процедур обробки подій. Більшість файлів проекту не потребують «ручного» редагування, бо їх вміст генерується автоматично.

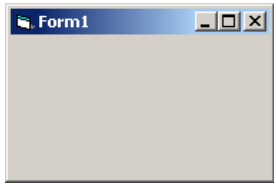
Виконання програми

Запустити програму на виконання можна такими способами:

1 спосіб. Відкрити меню *Run,* вибрати команду *Start*.

2 спосіб. Натиснути на панелі інструментів кнопку ▶.

Приклад 1.2. Запустіть програму на виконання − на екрані з'явиться стандартне вікно *Windows*-програми. Вікно порожнє, оскільки ми ще нічого на форму не поміщали, і має заголовок "Form1". Зверніть увагу: проект перейшов у режим виконання [*run*]. Форма дещо змінилась: зникли крапки, вона втратила зв'язок з головним вікном Visual Basic і поводиться, як незалежна програма. Щоб переконатися, згорніть на панель задач головне вікно Visual Basic − вікно проекту залишиться на екрані. Його можна пересувати по всьому екрану, вхопившись за заголовок, і на панелі задач *Windows* з'явиться відповідна кнопка. Вікно належним чином реагує на клацання кнопок керування у правому верхньому куті і дозволяє змінювати розміри.



Зупиніть виконання проекту (кнопка ■). Visual Basic вийде з режиму [*run*] і повернеться в режим [*design*].

Збереження проекту

Щоб зберегти щойно створений проект, або оновити вже існуючий зберігши зроблені доповнення, у меню *File* виберіть

команду *Save Project*. Якщо проект зберігається вперше, необхідно виконати такі кроки:

- створити папку для проекту;
- зберегти файл форми в папці проекту;
- зберегти файл проекту в папці проекту.

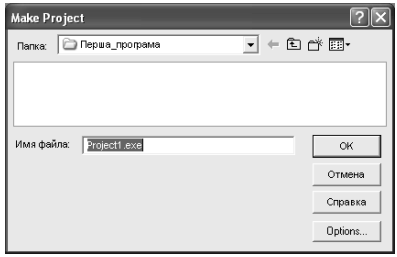
Збережіть проект, викликавши команду *File* ⇨ *Save Project*. У діалоговому вікні *Save File As*, що відкрилося, відкрийте або створіть загальну папку для VB-проектів, а в ній створіть нову папку для вашого проекту. Збережіть у папку файл форми (*Form1.frm*) і файл проекту (*Project1.vbp*).

Створення EXE-файлу

Компілятор *– програма, яка перекладає програму з мови програмування високого рівня на мову машинних кодів.*

Відкомпільовану програму можна зберегти для подальшого використання. Файл, що містить отриманий при компіляції програмний код, називається виконуваним файлом. Такий файл може мати розширення **.exe* або **.com*.

Щоб створити виконуваний файл проекту, необхідно виконати таку послідовність операцій:



- в меню *File* вибрати команду *Make*, поряд з якою вказана назва відкритого проекту з розширенням EXE;
- у діалоговому вікні *Make-Project* указати ім’я для створюваного EXE-файлу і папку, в якій його буде збережено;
- якщо є необхідність змінити деякі параметри даного файлу (номер версії, заголовок та значок програми), натиснути кнопку *Options* і внести зміни;
- натиснути *OK*.

Елемент керування Label (Напис)

Label (Напис) – це елемент керування, що застосовується для відображення тексту, недоступного для безпосереднього редагування користувачем: заголовків, підписів інших елементів керування тощо.

Для створення об’єкту *Напис* потрібно виконати такі дії:

1) вибрати компонент *Label* лівою кнопкою миші на панелі компонентів;
2) навести вказівник миші на форму і перетягуванням створити об'єкт *Напис*.

На панелі компонентів

На формі

Розташування об'єкта на формі змінюють, перетягуючи його мишею, а розміри об'єкта можна змінити за допомогою маркерів.

Напис містить текст, який можна прочитати під час роботи програми.

Кожен елемент керування при додаванні на форму отримує унікальне ім'я (властивість ***Name***). **Visual Basic** автоматично надає властивості ***Name*** значення, яке містить назву типу елемента керування з порядковим номером в кінці. Наприклад, для першого елемента *Label* значення властивості *Name* дорівнює *Label1,* для другого – *Label2* і т. д. Ім'я використовується для позначення елемента керування в програмному коді і може бути змінене тільки у вікні *Properties* на етапі розробки.

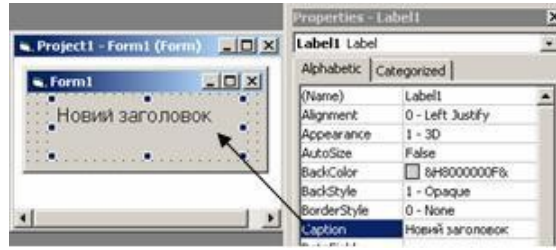
Програмісти, здебільшого, змінюють ім'я так, щоб воно вказувало на призначення елемента. При цьому часто використовують ***префікси***, які позначають належність об'єкта до певного типу.

Якщо елемент управління має змістовне ім'я, то легко визначити його роль у програмі.

Таким чином, ім'я складається з префікса, який ідентифікує тип об'єкта, і зручної назви, що описує призначення об'єкта в програмі:

Ім'я елемента = Префікс + Опис призначення

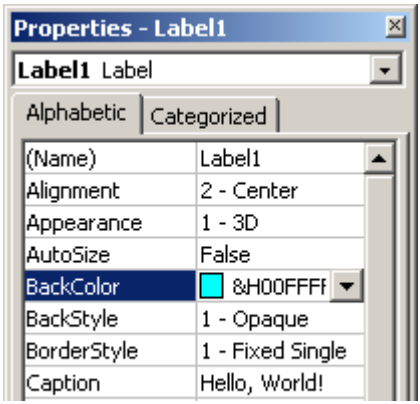
Для написів застосовують префікс *lbl*. Наприклад, якщо напис містить коментар, то замість імені *Label1* краще взяти *lblComment*.



Caption (Заголовок) – це основна властивість елемента керування *Label*, яка містить текст напису. Значення властивості *Caption* можна змінити як під час розробки вигляду форми у вікні *Properties*, так і в ході виконання проекту, запрограмувавши присвоєння властивості нового значення в програмному коді (детальніше − далі).

Щоб змінити значення властивості у вікні *Properties,* необхідно виконати такі дії:

1) виділити елемент керування на формі;
2) у вікні *Properties* в лівій частині таблиці вибрати властивість *Caption*;
3) клацнути у правій частині таблиці напроти назви властивості і ввести текст.



Приклад 1.3. Властивостям елементу керування *Label1* надані значення:

- властивості *Caption* (Заголовок) надано значення «Hello, World!»;
- для властивості *Alignment* (Вирівнювання) вибрано значення *2-Center*.
- напис виділено рамкою: для властивості *BorderStyle* (Тип межі) вибрано *1 – Fixed Single*.

- параметри шрифту змінено на *Arial, 16, жирний*. Для виклику вікна *Шрифт* потрібно активізувати рядок властивості *Font* і клацнути в ньому кнопку ...;
- для властивості *BackColor* (Колір фону) на вкладці *Palette* вибрано блакитний колір (*FFFF00*&).

Питання для самоконтролю

1. *Що таке мова програмування?*
2. *Чому середовище програмування Visual Basic IDE є інтегрованим?*
3. *Перелічіть складові системи програмування.*
4. *Як запустити систему програмування Visual Basic?*
5. *Що таке форма?*
6. *Як додати елемент керування на форму?*
7. *Як змінити розташування об'єкта на формі і його розміри?*
8. *Як змінити значення певної властивості об'єкта?*
9. *Як запустити проект на виконання?*
10. *Опишіть послідовність додавання елементів на форму і зміну властивостей для елементу Напис.*
11. *Що таке проект?*
12. *Які файли входять до складу проекту?*
13. *Опишіть послідовність дій при збереженні проекту.*

14. Для чого потрібно створювати EXE-файл для існуючого проекту?
15. Опишіть послідовність операцій, необхідну для створення виконуваного файлу.

1.2. Процедури опрацювання подій

Поняття об'єкта у програмуванні

Технологія роботи у середовищі **Visual Basic** базується на ідеях об'єктно-орієнтованого та візуального проектування інтерфейсу користувача. Програміст створює певну сукупність об'єктів та описує їх взаємодію. Кожен об'єкт має набір властивостей та може виконувати певні дії. Деякі об'єкти є візуальними, тобто зображуються на екрані. Прикладами візуальних об'єктів є відомі вам елементи керування у вікні: кнопки, списки, текстові поля тощо.

*Щоб об'єкт виконав якусь дію, потрібно для нього викликати відповідний фрагмент програми − **метод**.*

Отже, структурною одиницею при розробці інтерфейсу користувача в середовищі **Visual Basic** є візуальний об'єкт із певним набором властивостей і методів, який називається ***елементом керування***. Автоматизація розробки інтерфейсу користувача досягається завдяки можливості переносити елемент на форму (у програму) з панелі компонентів і змінювати його властивості, не вносячи вручну змін до програмного коду. Програміст має змогу змінювати значення властивостей даного об'єкта і викликати різні його методи. Також є можливість програмування реакції комп'ютера на різні події, які можуть виникати внаслідок деяких дій користувача. При виконанні програми користувач ініціює певні події і тим самим керує виконанням програми.

Приклад 1.4. Розглянемо взаємодію двох об'єктів: людини та телефонного апарату. Обробляючи подію «надходження сигналу з телефонної лінії» апарат «вмикає» дзвінок. Дзвінок телефону − це подія, на яку реагує людина: піднімає трубку. Щоб подзвонити комусь, людина застосовує метод «набрати номер». З телефоном при цьому відбувається послідовно

декілька подій «натискання кнопки». А властивості характеризують апарат: «колір», «висота», «кількість кнопок» тощо.

Атрибути об'єкта

Будь-яка написана на **Visual Basic** *Windows*-програма – це набір об'єктів (*Objects*). Так само як, скажемо, комп'ютер – набір знімних модулів − плат, приводів і т.д. Об'єкт характеризується певними атрибутами, які поділяються на три категорії:

- події (*Events*), на які об'єкт може реагувати, за умови, що програміст напише програмний код обробки події;
- методи – являють собою окремі дії, які об'єкт здатний виконувати;
- властивості (*Properties*) – характеристики об'єкта.

Кожен об'єкт у Visual Basic, наприклад, форма чи елемент керування, має свій власний набір властивостей, що його описують. Проте деякі властивості притаманні багатьом об'єктам.

Таблиця 1.2

Деякі спільні властивості елементів керування Visual Basic	
Властивість	**Опис**
Height	Висота елемента керування. Вимірюють у спеціальних одиницях − твіпах (див. далі).
Width	Ширина елемента керування (у твіпах)
Left	Відстань (у твіпах) від елемента керування до лівого краю його контейнера (див. далі)
Top	Відстань (у твіпах) від елемента керування до верхнього краю його контейнера
Name	Ім'я, яке використовується для посилань на елемент керування в програмі. Не може змінюватися під час виконання програми
Enabled	Логічна (*True*/*False*, тобто *Так/Ні*) властивість, яка визначає, чи може користувач працювати з цим елементом керування
Visible	Логічна (*True*/*False*, тобто *Так/Ні*) властивість, яка визначає видимість елемента керування під час виконання програми

Подія – це те, що відбувається в програмі або за її межами. Подією є будь-який вплив на елемент керування в активному вікні від миші чи клавіатури. Наприклад, коли користувач тисне на кнопку, відбувається відразу кілька подій: натискаються кнопка миші та командна кнопка *CommandButton*, відпускається кнопка

миші. Ці три дії відповідають подіям *MouseDown*, *Click*, *MouseUp*. Події, які відбуваються внаслідок дій користувача (наприклад, при переміщенні миші, натисканні клавіші на клавіатурі, клацання в текстовому полі), існують для багатьох елементів керування.

Методи являють собою фрагменти програмного коду, які вбудовані безпосередньо в елемент керування і виконують ту чи іншу задачу. Хоча різні об'єкти мають різні методи, деякі з методів притаманні багатьом об'єктам. Наприклад, спільними для багатьох елементів керування Visual Basic є методи *Move* (при виклику переміщує об'єкт), *Drag* (обробляє операції на зразок «перетягнути і відпустити»).

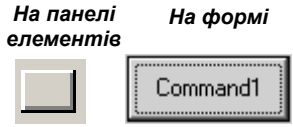
Система програмування **Visual Basic**, як і інші подібні системи, дозволяє формувати візуальну складову проекту (інтерфейс) без написання коду, а простими засобами графічного редагування (компонування). Не випадково цей процес називають візуальним конструюванням інтерфейсу користувача.

Але сформована візуальна складова повинна ще певним чином реагувати на ті чи інші події. До таких можна віднести сигнал від миші чи клавіатури, зміну стану елемента керування, одержання повідомлення від іншої *Windows*-програми і т.д.

Елемент керування Command Button (Командна кнопка)

Командна кнопка – один з найрозповсюдженіших елементів керування. Командна кнопка призначена для запуску чи закінчення виконання проектом певних дій, які викликаються натисканням на кнопку.

Вигляд кнопки на панелі ToolBox і на формі показано на малюнку.



Помістіть на форму елемент керування *CommandButton*. Даний об'єкт отримає ім'я *Command1*. Виділіть об'єкт *Command1* і перегляньте список його властивостей у вікні *Properties.* При розробці інтерфейсу здебільшого змінюють властивості *Name* (трибуквений префікс для імені командної кнопки – ***cmd***), *Caption*, встановлюють потрібні значення властивостей *Enabled* та *Visible*.

Приклад 1.5. Змініть для об'єкта *Command1* значення властивості *Name* на *cmdHello*. Надайте властивості *Caption* значення «Привітатися». Запустіть проект на виконання.

В режимі [*run*] на кнопку можна натискати мишкою, але при цьому нічого не відбувається, адже ми не написали програму, яка б

«пояснила» комп'ютеру, що потрібно робити при натисканні на кнопку.

Подійне програмування

У Visual Basic керування відбувається за допомогою подій, тобто коли в проекті відбувається подія, то виконується процедура опрацювання події (***event procedure***). Значна частина написаного вами програмного коду буде виконуватися у відповідь на дії користувача, тобто при виникненні подій. Такий тип програмування передбачає, що ви повинні знати, коли відбуваються події, і вміти створювати програмний код, який реагує на них.

Для кожного об'єкта існує набір стандартних подій, що можуть виникнути при роботі програми, і для кожної з них може бути написана процедура, яка обробляє подію.

Коли відбувається деяка подія, **Windows** посилає програмі повідомлення. Програма повинна визначити, яка подія стоїть за цим повідомленням, і виконати відповідні дії. Процедури подій повідомляють комп'ютеру, що робити у відповідь на подію. Якщо в програмі немає процедури для даної події, вона ігнорується.

Процедури опрацювання подій вміщують у вікні коду.

Code Window (Вікно коду)

Щоб написати процедуру обробки події для певного об'єкта, перейдіть у вікно редагування коду форми, виконавши одну з дій:

- двічі клацнути мишею на об'єкті;
- вибрати мишею об'єкт і натиснути ***F7***.

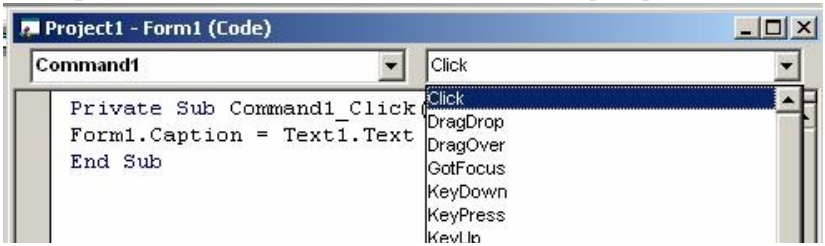
В основній частині вікна система підготувала оброблювач цієї події, оформлений у вигляді процедури.

Якщо ж програмний код було створено раніше, то перейти до нього можна, вибравши команду меню *View* ⇨ *Code* або клацнувши у вікні проекту кнопку *View Code* (див. мал.).

У верхній частині вікна коду знаходяться два списки, що розкриваються, – список об'єктів (*Object list*) і список процедур (*Procedure list*). Список об'єктів містить імена всіх елементів, що є на формі. У списку процедур перелічені всі події, на які може реагувати цей об'єкт.

При виборі зі списків об'єкта і події IDE автоматично створює шаблон процедури обробки цієї події або показує її код, якщо вона створена раніше.

Приклад 1.6. Клацніть на кнопці, що розкриває список



процедур, у верхньому правому полі вікна коду. Розгляньте інші доступні події для елемента керування *CommandButton*.

Структура процедури обробки подій

Оператор *– команда комп'ютеру виконати певну дію, записана за правилами мови програмування.*

Процедура – це набір операторів, представлений у вигляді іменованого блоку коду, який можна викликати з будь-якої частини програми за його ім'ям. Такий код може змінювати властивості об'єктів, отримувати дані від користувача, переміщати об'єкти у формі, обчислювати значення за формулою або записувати дані в базу даних тощо. Тут розглядаються процедури особливого виду – процедури обробки подій.

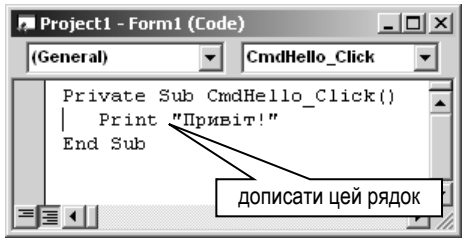
Загальна структура процедури обробки подій така:

Private Sub НазваЕлемента_НазваПодії()
[програмний код]
End Sub

Рядок заголовка говорить про те, що ми працюємо з ***Private*** (тобто доступною тільки для даної форми), ***Sub****routine* – стандартною підпрограмою – процедурою обробки подій. Ця процедура виконується, коли для елемента керування *НазваЕлемента* відбувається подія *НазваПодії*. Безпосередньо після заголовка записують код процедури обробки подій – послідовність операторів. Комп'ютер виконує спочатку перший оператор, потім другий і всі наступні оператори один за одним, до оператора завершення процедури *End Sub*.

Якщо двічі клацнути на командній кнопці, створюється процедура обробки події *Click*. Ім'я процедури обробки події складається з імені об'єкта та імені події, зв'язаних символом підкреслення (*Command1_Click*( )). Між заголовком і *End Sub* необхідно написати оператори, що будуть виконані при клацанні кнопки.

Приклад 1.7. Продовжимо роботу над проектом з прикладу 1.5. Нехай клацання кнопки повинне призводити до виведення на форму тексту «Привіт!». Цю дію виконує оператор *Print*. Запрограмуйте реакцію на натискання кнопки «Привітатися»:



1) двічі клацніть на ній;
2) у вікні програмного коду внесіть зміни до процедури *Private Sub CmdHello_Click()*, відповідно до малюнка.

Запустіть проект на виконання і проаналізуйте дію кнопки.

Події для форми

У цьому курсі ми розглянемо тільки дві події для форми – *Click* і *Load*.

Подія *Click* відбувається, коли користувач клацає на формі лівою кнопкою миші. Подія *Load* відбувається, коли форма завантажується у пам'ять комп'ютера. Це зручний момент для встановлення початкових значень різних властивостей елементів керування та інших параметрів проекту.

Усі імена процедур обробки подій для форми мають формат: ***Form***_*EventName*.

Незалежно від того, яке значення властивості Name ви встановлюєте для форми, ім'я процедури обробки події міститиме слово Form.

Оператор присвоєння

Оператор присвоєння – основний оператор більшості мов програмування, використовується для надання змінній потрібного значення. У мові **Basic** він має вигляд:

Змінна = НовеЗначення, де:

- символ «=» якраз і позначає оператор присвоєння, тобто, дію «присвоїти (надати) значення»;
- *Змінна* – ім'я змінної або властивості елемента керування;

- *НовеЗначення* – константа, змінна або вираз того ж самого типу, що і *Змінна*. Детальніше типи даних будуть розглянуті далі.

Оператор присвоєння діє таким чином: змінній, вказаній в лівій частині оператора присвоєння, присвоюється значення виразу, записаного у правій частині. Ліворуч від оператора присвоєння може бути вказане тільки одне ім'я змінної або властивості елемента керування.

VarName =NewValue		**Приклади**
▪ Ім'я змінної (символьне позначення адреси комірки пам'яті, куди треба помістити результат) ▪ Властивість деякого об'єкта	▪ Константа	A = 6 cmdBlue.Visible = False
	▪ Вираз	Sum = Sum+5
	▪ Ім'я змінної	A = B

Приклад 1.8. Нехай значення змінної *A* дорівнює 3. Тоді після виконання вказівки присвоєння

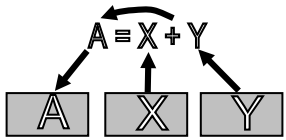
A =A + 1

змінна *A* отримає значення 4.

Приклад 1.9. Нехай *A*=10, *X*=2, *Y*=3. Тоді після виконання вказівки присвоєння

A = X + Y

змінна *A* отримає значення 5.



Попереднє значення змінної A (число 10) буде втрачене.

Приклад 1.10. У вікні коду знаходиться процедура опрацювання події *Click* для елемента керування *Command1* (командна кнопка). Внаслідок виклику процедури (тобто, після клацання на командній кнопці) текст напису *Label1* стає заголовком форми.

Цю дію позначає оператор присвоєння, який властивості *Caption* форми надає значення властивості *Caption* елемента *Label1*.

```
Private Sub Command1_Click()
    Form1.Caption = Label1.Caption
End Sub
```

Більшість властивостей форм і елементів керування є змінними. Їх значення можна задати в режимі розробки і змінювати в ході виконання програми за допомогою оператора присвоєння.

Взаємозв'язок властивостей, методів і подій

Хоча властивості, методи і події – це зовсім різні речі, дуже часто вони виявляються взаємозалежними. Наприклад, якщо елемент керування переміщується за допомогою методу *Move* (зокрема, у відповідь на подію), змінюються властивості *Top* і *Left* елемента керування.

Багатьом властивостям можна присвоювати нові значення. Деякі методи потребують при виклику вхідних параметрів.

Основна відмінність між методами й властивостями полягає в тім, що із властивостями можна працювати як під час розробки проекту, так і під час виконання програми, тоді як методи доступні тільки при виконанні.

Взаємозв'язок властивостей, методів і подій означає, що багато дій можна виконувати декількома способами, використовуючи відповідний програмний код роботи з подіями і методами об'єкта.

Приклад 1.11. Переміщення кнопки *CommandButton*

Переміщення кнопки за допомогою властивостей	**Переміщення кнопки за допомогою методу Move**
Command1.Left = 100 *Command1.Top = 100*	*Command1.Move 100,100*

Приклад 1.12. Відображення і приховування форми на екрані

Робимо форму видимою, змінивши властивість	**Відображаємо форму за допомогою методу**
Form1.Visible = True	*Form1.Show*
Робимо форму невидимою, змінивши властивість	**Приховуємо форму за допомогою методу**
Form1.Visible = False	*Form1.Hide*

Питання для самоконтролю

1. *Що таке властивості елемента керування? Наведіть приклади.*
2. *Опишіть послідовність зміни значень властивостей.*
3. *Що таке методи елемента керування?*
4. *Що таке події? Наведіть приклади подій.*
5. *Опишіть послідовність додавання на форму і зміни властивостей елементу Командна кнопка.*
6. *Як перейти до вікна програмного коду? Опишіть структуру вікна програмного коду.*
7. *Імена яких об'єктів містить список Object list у верхній частині програмного коду?*
8. *Перелічіть та опишіть властивості, спільні для багатьох елементів керування.*
9. *Поясніть структуру процедури обробки події.*

10. *Як внести зміни до програмного коду для програмування реакції на натискання командної кнопки?*
11. *Для чого, на вашу думку, змінюють значення властивості Name елементів керування, що розміщені на формі? Який префікс використовують в іменах командних кнопок (CommandButton)?*
12. *Які події елемента Form вам відомі?*
13. *Поясніть синтаксис і схему дії оператора присвоєння.*
14. *Яким буде результат виконання оператора Label1.Text = Form1.Caption?*

1.3. Етапи розв'язування задач на комп'ютері

Етапи розв'язування задачі на комп'ютері

Найбільш загальними складовими комп'ютерної програми є дані, логіка роботи та інтерфейс користувача.

Дійсно, кожна з програм обробляє певні *дані*. Деякі з них користувач вводить під час виконання програми, користуючись *інтерфейсом користувача*. Обробка даних підпорядкована певному алгоритму, який, по суті, описує *логіку роботи* програми.

Тому, розв'язування прикладної задачі на комп'ютері з використанням програмування включає такі етапи:

1) постановка задачі;
2) проектування інтерфейсу;
3) розробка алгоритму;
4) написання програмного коду;
5) тестування й налагодження програми;
6) аналіз результатів.

І етап. Постановка задачі. Розв'язування практичної задачі починається з її формулювання таким чином, щоб чітко визначити умови задачі:

- Що дано?
- Які дані допустимі?
- Які результати, в якому вигляді повинні бути отримані?

Для чіткого визначення переліку початкових даних і результатів виконання програми зручно використовувати таку форму:

Дано: <Перелік початкових даних>

Потрібно: <Перелік потрібних результатів>

Зв'язок: <Система рівнянь або тверджень, що зв'язують вхідні та шукані дані>

При <Умови допустимості початкових даних>

II етап. Проектування інтерфейсу. Перед розробкою інтерфейсу форми поставте перед собою питання:

- Які елементи керування необхідні для введення інформації?
- Яку інформацію буде обробляти комп'ютер?
- Які елементи керування необхідні для виведення результатів виконання програми?

Розробка інтерфейсу проекту **Visual Basic** включає два кроки:

1) розміщення елементів керування на формі;
2) налаштування властивостей елементів керування.

Дружній інтерфейс користувача *(зручне розташування елементів керування на формі та продумана їх взаємодія), прикладом якого є інтерфейс Windows, буде гарною основою для вашого проекту.*

III етап. Складання алгоритму. Для того, щоб комп'ютер виконав потрібні обчислення і повідомив нам правильний результат, ми повинні скласти для нього чітку інструкцію, задати послідовність дій, яку необхідно виконати для розв'язування задачі. Цю інструкцію називають алгоритмом розв'язування задачі.

IV етап. Складання програмного коду за розробленим алгоритмом. Програмування (складання програми) – кодування складеного алгоритму однією з мов програмування. Створення програмного коду мовою **Visual Basic** зводиться до написання процедур обробки подій для елементів керування та, за потреби, інших процедур.

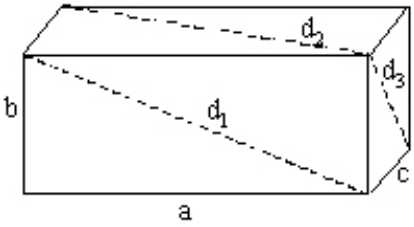
V етап. Тестування і налагодження програми. На даному етапі проводиться перевірка правильності роботи програми за допомогою тестів і виправлення виявлених помилок. Тест – це набір спеціально підібраних вихідних даних і результатів роботи програми, очікуваних при цих даних. Процес тестування включає: запуск програми; введення тестового набору даних; порівняння результатів, наведених у тестовому прикладі, з результатами, отриманими після виконання програми.

VI етап. Аналіз результатів. На завершальному етапі програма виконується з даними, що входять до розв'язуваної задачі. Після остаточного виконання програми проводиться аналіз

результатів. У випадку невірогідності результатів можлива зміна самого підходу до розв'язання задачі (алгоритму) і повернення до етапу постановки задачі для її корегування та уточнення.

Приклад 1.10. Через круглий ілюмінатор корабля, що затонув, потрібно витягти скриню з коштовностями. Чи вдасться це зробити?

Звичайно, при такому формулюванні задачі відповісти на поставлене питання неможливо. Тому уточнимо постановку задачі, з'ясувавши, які дані потрібні для розв'язування задачі і які значення вважатимемо результатом. Ілюмінатор корабля має форму кола. Припустимо, що скриня має форму паралелепіпеда.



Нехай D – діаметр ілюмінатора,
a, b, c – розміри скрині,
d_1, d_2, d_3 – діагоналі бічних поверхонь скрині.

Скриню можна просовувати через ілюмінатор однією з трьох бічних граней, отже, достатньо, щоб діаметр ілюмінатора виявився більшим хоча б від однієї з трьох діагоналей граней скрині. Для розв'язування задачі необхідно перевірити три умови. Введемо допоміжні змінні P_1, P_2, P_3, які одержують значення 1 або 0 в залежності від виконання відповідної умови.

Дано: R – радіус ілюмінатора,
a, b, c – розміри скрині,

Потрібно: висновок про те, чи є діагональ, менша від діаметра.

Зв'язок: $d_1 = \sqrt{a^2 + b^2}$. Якщо $D > d_1$, то P_1=1.

$d_2 = \sqrt{a^2 + c^2}$. Якщо $D > d_2$, то P_2=1.

$d_3 = \sqrt{c^2 + b^2}$. Якщо $D > d_3$, то P_3=1.

Якщо $P_1 + P_2 + P_3 = 0$, то робимо висновок: коштовності недоступні; в іншому випадку, скриню можна дістати.

При $R>0$; $a>0$; $b>0$; $c>0$.

Постановка задачі може ускладнюватися і доповнюватися. Наприклад, може знадобитися врахувати вагу скрині – чи зможе водолаз підняти її. Якщо передбачається застосування підйомного механізму, треба враховувати товщину тросів, якими обв'яжуть скриню тощо.

Поняття алгоритму

Алгоритм – це скінчена послідовність вказівок, яка однозначно описує процес розв'язування задачі певного типу.

Кожен алгоритм припускає наявність деяких вихідних даних і приводить за обмежений час до певних результатів.

Поняття алгоритму є в інформатиці фундаментальним, як, наприклад, «точка» і «площина» – у геометрії, «простір» і «час» – у фізиці).

Слово «алгоритм» походить від латинської форми написання імені арабського математика Аль-Хорезмі (800-847 рр.), що сформулював правила чотирьох арифметичних дій над числами.

З курсу математики добре відомі алгоритми виконання арифметичних операцій над багатоцифровими числами; алгоритми знаходження коренів лінійних і квадратних рівнянь; алгоритми поділу відрізка на *N* рівних частин, побудови трикутника за заданими його елементами та ін.

Приклад 1.11. Алгоритм побудови графіка функції виду $y=|f(|x|)|$ може бути поданий у такий спосіб:

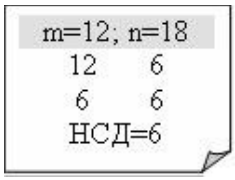
1) побудувати графік функції $y=f(x)$ для значень $x\geq 0$;
2) побудовану лінію симетрично відобразити відносно осі *Oy*;
3) частину побудованого графіка, що знаходиться в нижній півплощині, симетрично відобразити відносно осі *Ox*;
4) частину графіка, що знаходиться в нижній півплощині, стерти.

У повсякденному житті людина зустрічається з різними алгоритмами, спрямованими як на обчислення певного значення, так і не зв'язаними з обчислювальними процесами, а такими, що визначають різноманітні послідовності дій.

Приклад 1.12. Числові алгоритми створюються для розв'язування обчислювальних задач.

Алгоритм Евкліда. Алгоритм знаходження найбільшого спільного дільника двох чисел *m* і *n* − *НСД(m,n)* − був описаний в III столітті до н.е. в класичному трактаті «Початки» грецького математика Евкліда.

Розв'язок можна одержати шляхом послідовного віднімання меншого числа від більшого доти, поки $m\neq n$.



- *Крок 1.* Якщо $m=n$, то перейти до кроку 5.
- *Крок 2.* Визначити більше з чисел.

- *Крок 3*. Відняти від більшого числа менше. Отриманою різницею замінити більше число.
- *Крок 4*. Перейти до кроку 1.
- *Крок 5*. *НСД(m,n)=m*.

Приклад 1.13. Алгоритми ігрових задач.

Для багатьох ігор, результат яких залежить не від випадкового збігу обставин, а від кмітливості гравця і попереднього розрахунку, існують алгоритми виграшу (виграшні стратегії).

Гра Баше. Є 11 предметів. За один хід гравець може взяти 1, 2 або 3 предмети. Програє той, хто змушений взяти останній предмет.

Алгоритм виграшу для 1-го гравця:

- *1 хід*: узяти 2 предмети.
- *2 хід і далі*: брати стільки предметів, щоб кількість предметів, узятих разом із суперником за черговий хід, становила 4.

Властивості алгоритму

Розробка алгоритму – найважливіший етап розв'язування задачі. Від якості алгоритму залежать правильність результатів, ефективність використання часу та ресурсів комп'ютера.

Алгоритм повинен мати такі властивості:

Масовість – придатність для обробки великої кількості варіантів вхідних даних. Наприклад, алгоритм обчислення площі трапеції за заданими основами та висотою повинен давати правильний результат при будь-яких коректних вхідних даних, алгоритм знаходження коренів квадратного рівняння повинен бути придатним для розв'язування будь-якого рівняння виду $ax^2+bx+c=0$.

Визначеність (однозначність) – однозначність тлумачення правил виконання дій і порядку їхнього виконання. Наприклад, розпорядження «Порівняти числа *A* і *B*» може бути витлумачене порізному, тому в алгоритмі неприпустиме.

Дискретність означає, що алгоритм повинен складатися з окремих завершених дій. Розпорядження повинні мати форму «виконати», «зробити», а не «виконувати», «робити».

Результативність означає, що виконання послідовності вказівок алгоритму повинно приводити до цілком конкретного результату. Наприклад, алгоритм розв'язування квадратного рівняння повинен містити перевірку випадку, коли коренів не існує, і передбачати інформування про відсутність коренів.

Формальність означає, що будь-який виконавець, здатний сприймати і виконувати вказівки алгоритму (навіть не розуміючи їх змісту), діючи за алгоритмом, може виконати поставлене завдання.

Як відомо, автомати правильно розв'язують багато задач за заданими їм алгоритмами, хоча суті задач, безумовно, автомати розуміти не можуть.

Сукупність всіх команд, які даний виконавець може виконувати, називають його ***системою команд****.*

Кожен алгоритм розрахований на певного виконавця, тому повинен містити тільки ті команди, які входять до системи команд виконавця.

Скінченість – потенційна виконуваність алгоритму. Алгоритм повинен складатися зі скінченого числа кроків, кожний з яких вимагає для свого виконання скінченого проміжку часу.

Приклад 1.14. Нехай деякий учень задумав натуральне число *X*. Запропонуємо йому виконати алгоритм:

1) помножити задумане число на три;
2) від добутку відняти одиницю;
3) отриману різницю помножити на 4;
4) від добутку відняти 5;
5) повідомити результат (*Y*).

Якщо ми знаємо результат виконання алгоритму – *Y*, ми легко відгадаємо задумане число *X*, якщо скористаємось алгоритмом відгадування:

1) додайте до *Y* число 9;
2) суму поділіть на 12;
3) повідомте результат (*X*).

Зрозуміло, що дії, виконані над числом *X*, описує формула:

$(3x - 1) \cdot 4 - 5 = y$.

Отже, $x = (y + 9) / 12$.

Алгоритм відгадування числа *X* має властивість *формальності*, тобто якщо інший учень правильно виконає дії алгоритму, він отримає потрібний результат – відгадає число *X*, навіть не знаючи, обчислення за якою формулою цей алгоритм реалізує.

Форми подання алгоритмів

Існують різні форми подання алгоритмів – словесна, формульна, словесно-формульна, графічна, у вигляді програмного коду та інші – залежно від того, на якого виконавця орієнтований алгоритм.

У словесній формі можна подати багато обчислювальних алгоритмів, наприклад, правила виконання арифметичних дій над

багатоцифровими числами, обчислення коренів квадратного рівняння, довжини кола, площі трикутника, алгоритм Евкліда для знаходження найбільшого спільного дільника двох чисел та ін.

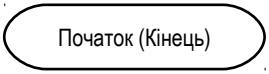
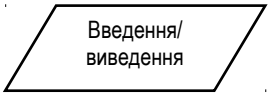
Обчислювальні алгоритми можна задавати й у вигляді формул. Так, алгоритм обчислення суми членів нескінченно спадної геометричної прогресії можна подати у вигляді формули $S=\frac{b_1}{1-q}$, а алгоритм обчислення площі прямокутного трикутника – у вигляді $S=\frac{a\cdot b}{2}$. Записуючи алгоритми, часто комбінують словесне і формульне подання вказівок.

Для графічного подання вказівок використовують блок-схеми.

Блок-схема алгоритму – графічне зображення алгоритму у вигляді сукупності з'єднаних блоків.

Графічне зображення вказівок

Таблиця 1.4

Назва блоку	Опис дії
	Позначає початок та завершення алгоритму
	Позначає дію, яку потрібно виконати. Прямокутником може бути позначена як вказівка виконати окрему дію (додати два числа, накреслити лінію), так і послідовність логічно пов'язаних дій (виконати розрахунки за заданими формулами, намалювати малюнок, відтворити мелодію), тобто певний процес.
	Позначає введення вхідної інформації та виведення проміжної і результуючої інформації.
	Позначає перевірку значення логічного виразу, деякої умови. Логічний вираз може набувати одного з двох значень – *TRUE* (істина, так) або *FALSE* (хибність, ні).

Базові алгоритмічні структури

Різні типи задач вимагають різних підходів до їх розв'язування і, відповідно, описуються різними алгоритмами. Розмаїтість цих алгоритмів велика, але для опису послідовності виконання дій при

складанні будь-якого алгоритму використовуються подібні блоки, які називають базовими структурами.

Логічна структура будь-якого алгоритму може бути подана комбінацією трьох базових структур: слідування, розгалуження, повторення (цикл).

Принцип структурного програмування полягає в тому, що на будь-якому етапі проектування алгоритму порядок виконання дій задається однією з базових структур:

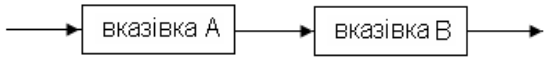
- слідування,
- розгалуження,
- повторення.

Найважливішою особливістю базових структур алгоритмів є те, що кожна з них має єдиний вхід і єдиний вихід.

Кожен прямокутник на блок-схемі алгоритму може бути замінений сукупністю простіших вказівок, що у свою чергу являють собою базові структури алгоритмів чи їх комбінації. При конструюванні алгоритму вихід кожної базової алгоритмічної структури приєднується до входу наступної. Як наслідок, алгоритм в цілому та кожна його частина являє собою ланцюжок базових алгоритмічних структур, що має важливе значення для аналізу алгоритмів, їхнього розуміння, корегування, доказу правильності

Слідування

Базова структура «Слідування» утворюється діями, що виконуються одна за одною, без пропусків або повторень. Алгоритми, у яких використовується тільки структура «Слідування», називаються ***лінійними***.



У структуру «Слідування» можуть бути організовані вказівки ввести початкові значення змінних, обчислити нові значення змінних, вивести результати обчислень тощо.

Приклад 1.15. ***Задача***. Скільки комп'ютерів можна встановити в комп'ютерному класі, довжина якого a м, ширина b м, якщо згідно з санітарними нормами на 1 ПК повинно припадати 6 м^2?

Математична постановка задачі:

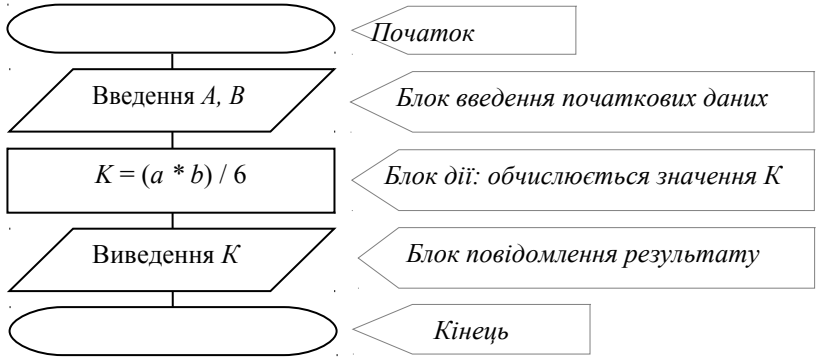
Дано: a – довжина класу (м), b − ширина класу (м).

Потрібно: K – кількість комп'ютерів.

Зв'язок: $K = \dfrac{ab}{6}$.

При $a>0$; $b>0$.

Блок-схема алгоритму:



З базовими структурами «Розгалуження» і «Повторення» ви ознайомитесь далі. Алгоритми розв'язування складних задач, здебільшого, містять у собі базові структури всіх трьох типів.

Питання для самоконтролю

1. *Перерахуйте етапи розв'язування задачі з використанням комп'ютера.*
2. *Опишіть зміст кожного з етапів розв'язування задачі.*
3. *З яких кроків складається етап проектування інтерфейсу?*
4. *Що таке алгоритм? Назвіть основні вимоги до алгоритмів і поясніть суть кожної з них.*
5. *Нехай маємо такий алгоритм:*
   - *прочитати перше число – a1;*
   - *прочитати друге число – a2;*
   - *поділити число a1 на a2;*
   - *записати результат.*

   Чи має даний алгоритм властивості масовості та однозначності?
6. *Деяке задане число, більше одиниці, необхідно зменшити до одиниці шляхом ділення на два:*
   - *поділити число на два;*
   - *за результат прийняти частку (нове число);*
   - *якщо частка не дорівнює 1, перейти на перший крок; інакше – записати результат.*

   Чи має даний алгоритм властивість скінченності?
7. *Нехай маємо такий алгоритм:*
   - *взяти 0,5 кг борошна;*
   - *взяти склянку цукру;*
   - *взяти 1 г лимонної кислоти;*
   - *взяти 4 яєчні жовтки;*
   - *взяти склянку кефіру;*
   - *спекти пиріг.*

   Чи має даний алгоритм властивості формальності та однозначності?

8. *Виконайте алгоритм Евкліда для чисел: а) 75 і 120; б) 75 і 84. Які властивості має алгоритм Евкліда?*
9. *Прямокутник, довжини сторін A і B якого задовольняють умову a/b=b/(a-b), називається «золотим». Складіть алгоритм для перевірки, чи є даний прямокутник «золотим». Яким може бути результат виконання складеного вами алгоритму?*

Скласти блок-схеми алгоритмів для розв'язання задач:

10. *Пішохід пройшов S_1 км за час T_1 годин. Яку відстань пройде пішохід за час T_2?*
11. *Ширина шпалер 70 см. Скільки метрів шпалер треба купити для ремонту кімнати довжиною a м, шириною b м та висотою h м? Наявністю вікон та дверей знехтувати.*
12. *Скільки грамів фарби буде потрібно для фарбування стола, якщо на фарбування 1 м^2 потрібно x грамів фарби?*

1.4. Основи програмування мовою Visual Basic

Складові мови програмування

Основними складовими будь-якої мови програмування є алфавіт, синтаксис і семантика.

До ***алфавіту*** мови програмування, як правило, входять: літери латинського алфавіту, цифри, знаки арифметичних операцій, спеціальні символи, ключові (зарезервовані) слова мови.

У мові **Visual Basic** при створенні програм можуть використовуватися такі символи:

- літери латинського алфавіту A..Z, a..z;
- цифри 0..9;
- знаки арифметичних операцій, спеціальні символи: + – * / \^ = < > ( ) . , : ; ‘ « @ $ # & _;
- комбінації символів: <=, >=, <>;
- ключові (зарезервовані) слова, що мають однаковий зміст у будь-якій програмі на **Visual Basic**, наприклад: *Dim*, *As*, *If*, *While* тощо.

Синтаксис мови – сукупність правил побудови команд мови програмування.

Семантика мови – сукупність правил виконання комп'ютером команд, записаних мовою програмування.

З синтаксисом та семантикою команд **Visual Basic** ви будете ознайомлюватися по мірі вивчення мови програмування.

Опис величин

Величина – це об'єкт, який має стале або змінне значення.

Основними характеристиками величини є тип, вид і значення. Деяким величинам надають імена.

Тип величини – це множина допустимих значень величини. Тип визначає обсяг пам'яті, необхідний для збереження величини, та операції, які можна над нею виконувати.

Вид величини визначає спосіб використання величини в алгоритмі. Величина може бути константою або змінною. ***Змінні*** – це іменовані величини, значення яких може змінюватися в ході виконання програми. ***Константи*** – це просто числа чи набори символів. Іноді константам теж надають імена.

Ім'я (ідентифікатор) величини – це назва змінної або константи, що обирається програмістом. Ідентифікатори мають також інші елементи програми: процедури, функції, типи даних.

Обмеження на запис ідентифікаторів:

- допустимі лише латинські літери, цифри та знак підкреслення (зокрема, не допускаються крапки та пропуски);
- імена починаються тільки з літери або знаку підкреслення;
- найбільша довжина імені – 255 символів,
- недопустимі збіги імен з ключовими словами.

У числових константах між цілою і дробовою частинами десяткового дробу ставлять крапку.

Приклад 1.16. Програму набагато легше зрозуміти, якщо всюди, де потрібне число $\pi \approx 3{,}14159$, написано *Pi*, а не цифри 3.14159. Якщо на початку процедури написати: *const Pi = 3.14159*, то при виконанні команди *Print 2 * Pi* комп'ютер замінить ім'я *Pi* вказаним числом і виведе на формі *6.28318*.

Значення змінної може багаторазово змінюватися в процесі виконання програми. Для цього застосовують оператор присвоєння.

Команди	Значення змінних	
	a	b
a=5	5	не визначене
b=8	**5**	**8**
a=a+b	13	8
b=a−b	13	5
a=a−b	**8**	**5**

Приклад 1.17. Розглянемо наведену в таблиці послідовність команд. Як бачимо, використовуючи операції додавання й віднімання та оператор присвоєння вдалося поміняти місцями значення двох змінних.

Робота зі змінними

Змінні можуть зберігати практично будь-які дані: числа, рядки тексту, візуальні об'єкти тощо.

У табл. 1.5 наведено опис найчастіше використовуваних в **Visual Basic** типів даних (крім візуальних об'єктів).

Таблиця 1.5

Типи даних			
Тип даних	**Інформація, що зберігається**	**Обсяг займаної пам'яті (байт)**	**Діапазон значень**
Integer	Цілі числа	2	від –32768 до 32767
Byte	Цілі числа	1	від 0 до 255
Long	Цілі числа	4	від − 2 147 483 648 до 2 147 483 647
Single	Дійсні числа	4	від $-3{,}4\cdot10^{38}$ до $+3{,}4\cdot10^{38}$
Double	Дійсні числа	8	від $-1{,}8\cdot10^{308}$ до $+1{,}8\cdot10^{308}$
String	Текстова інформація	1 на кожен символ	близько 4 млрд. символів; константи записують в лапках
Boolean	Логічні (булеві) значення	2	True (так), False (ні)
Date	Інформація про дату і час	8	від 1.01.100 р. до 31.12.9999 р
Variant	Значення кожного з вищеназваних типів	16+1 на кожен символ	Не визначений

Типи властивостей

Кожна з властивостей об'єктів також належить до певного типу. При налаштуванні об'єктів у вікні властивостей (у режимі розробки), **Visual Basic** автоматично пропонує відповідний тип даних.

Щоб запрограмувати зміну властивості у процедурі обробки подій з використанням оператора присвоєння, необхідно знати тип властивості, тому що значення властивості можна змінити тільки на значення того ж типу. При цьому в лівій частині оператора присвоєння використовується «запис через крапку»:

Ім'яЕлементу.Ім'яВластивості = Значення

де *Ім'яЕлементу* – це ім'я елементу керування; *Ім'яВластивості* – ім'я властивості, яку слід змінити; ***Значення*** – нове значення, яке присвоюється цій властивості.

Таблиця 1.6

Типи властивостей			
Тип	**Властивості даного типу**	**Приклад використання**	**Пояснення**
Integer	*Top*, *Left*, *Height*, *Width*	*Form1.Width = 4000*	Ширина (*Width*) форми *Form1* стане рівною 4000
Long	*BackColor*, *ForeColor*	*Form1.BackColor = vbBlue*	Колір фону (*BackColor*) форми, набуде значення константи, що має ім'я *vbBlue* (див. табл. 1.15)
Boolean	*Visible*	*Form1.Visible = False*	Форма стане невидимою
String	*Caption*	*Form1.Caption = "Про програму"*	Заголовок форми з ім'ям *Form1* набуде значення «Про програму»

Оголошення змінних

Після вибору імен і типів змінних треба помістити інформацію по це в проект, тобто потрібно оголосити змінні. Оператор, за допомогою якого оголошується змінна з ім'ям *Ім'яЗмінної* типу *ТипЗмінної*, записується так:

Dim Ім'яЗмінної As ТипЗмінної

де *Dim* – це ключове слово, скорочення від *Dimension* (величина, розмірність), *As* − з англійської «як, у якості».

При запуску проекту на виконання компілятор створить всі оголошені змінні, тобто виділить пам'ять для збереження даних в обсязі, який відповідає вказаному типу.

Приклад 1.18. За таким описом:

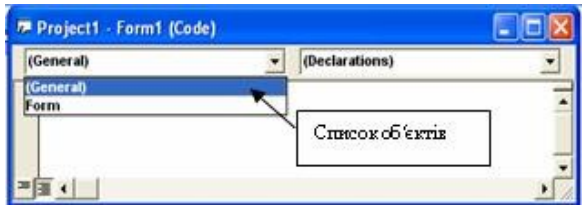
Dim A As Single, B As Integer

буде виділено 4 байти для змінної *A* та 2 байти для змінної *B*.

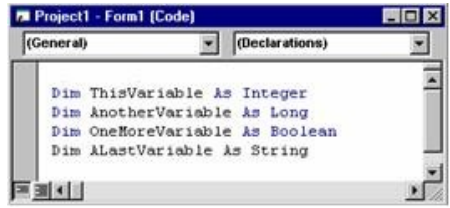
Після виконання оператора *Dim* комірки пам'яті комп'ютера, що виділені під змінні *A* і *B*, є порожніми (*Empty*) − змінні ще не набули значень.

Де ж розміщуються оголошення змінних? Створіть новий проект у **Visual Basic**, перейдіть у вікно коду та відкрийте список об'єктів. Там ви побачите два записи: *(General)* і *Form*. Виберіть *(General) – Загальні*. Вікно коду буде виглядати так:

Так ви відкриєте частину коду, що називається ***розділом загальних оголошень*** (*general declarations*). Якраз тут і розміщують оголошення змінних. Змінні, оголошені в розділі загальних оголошень, є ***глобальними***. Вони можуть використовуватися в будь-якій процедурі обробки подій і зберігають свої значення доти, поки виконується код форми.



Якщо змінна потрібна тільки в коді певної процедури обробки подій, оператор *Dim* вміщують після заголовку даної процедури. Змінні, оголошені в тілі процедури, є ***локальними***. Вони втрачають свої значення при виході з даної процедури.



Математичні оператори

Visual Basic підтримує ряд математичних операцій, які використовують у виразах.

Таблиця 1.7

Математичні операції і відповідні їм символи операторів Visual Basic			
Операція	**Символ оператора**	**Приклад**	**Результат**
Додавання	+	*Res = 15+3*	Res = 18
Віднімання	-	*A = Res – 10*	A = 8
Множення	*	*A = A*2*	A = 16
Ділення	/	*Res = 2.3 / 2*	Res = 1.15
Цілочисельне ділення	\	*Res = 9 \ 2*	Res = 4
Ділення по модулю (обчислення остачі)	*Mod*	*Res = 11 mod 3*	Res = 2
Піднесення до степеня	^	*A = 2^3*	A = 8

Порядок виконання (пріоритет) математичних операцій:

1) піднесення до степеня (^);
2) множення (*) і ділення (/), цілочисельне ділення (\), обчислення остачі від цілочисельного ділення (*Mod*);
3) додавання (+) і віднімання (–).

Якщо операції мають однаковий пріоритет, то вони виконуються зліва направо по черзі.

Приклад 1.22. В результаті виконання оператора присвоєння

*A = 24/2*3*

змінна *A* одержить значення 36.

При необхідності порядок дій регулюється дужками. В результаті виконання оператора присвоєння

*A = 24/(2*3)*

змінна *A* одержить значення 4.

Крім констант та змінних у виразах можуть застосовуватись математичні функції. Наприклад, функцію добування квадратного кореня з аргументу *x* позначають *Sqr(x)*. Детальніше ця та інші функції будуть описані далі.

Логічні оператори

Крім математичних у мові Visual Basic можна обчислювати значення логічних виразів, які мають тип *Boolean*, тобто набувають значення *True* (Істина) або *False* (Хибність). Значення таких виразів можна присвоювати змінним та властивостям логічного типу.

Таблиця 1.8

Логічний оператор	Операція	Логічний вираз	Значення
=	Дорівнює	8=9	False
>	Більше	8>9	False
<	Менше	8<9	True
>=	Більше або дорівнює	5>=5	True
<=	Менше або дорівнює	5<=2	False
<>	Не дорівнює	2<>5	True

Приклад 1.23. Робота з логічними значеннями

```
Dim x As Integer, y As Integer, A As Boolean
    x = 5 : y = 2
    A = x > y   ' змінна A стане рівна True
    A = x < y   ' змінна A стане рівна False
```

Як видно з прикладу, записуючи два оператори в одному рядку, їх відокремлюють двокрапкою

Для побудови складніших логічних виразів використовують стандартні логічні операції, які повертають *True* чи *False* в залежності від значень аргументів (див. табл. 1.9).

Порядок виконання (пріоритет) логічних операцій:

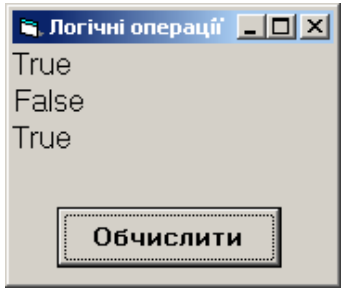
1 – *Not*; 2 – *And*; 3 – *Or*.

Таблиця 1.9

Таблиця істинності для логічних операцій						
A	B	A **AND** B	A **OR** B	A **XOR** B	A	**NOT** A
False	False	False	False	False	False	True
False	True	False	True	True		
True	False	False	True	True	True	False
True	True	True	True	False		
Коментар		*A **AND** B=true, якщо й A, і B істинні*	*A **OR** B=false, якщо й A, і B хибні*	*A **XOR** B=false, якщо A= B*	*Якщо A=True, то NOT A=false*	

Приклад 1.24. Використання логічних операцій

```
Dim x As Integer, y As Integer, z As Integer
Dim A As Boolean
Private Sub Command1_Click()
x = 1:    y = 2:    z = 3
      A = x < y And y < z
      Print A
      A = x > y Or y > z
      Print A
      A = x > y Xor y < z
      Print A
End Sub
```



Елемент керування TextBox

TextBox (текстове поле) являє собою вікно, призначене для введення і виведення тексту під час виконання програми (*run mode*).

Події для елемента керування TextBox. Основна властивість елемента керування *TextBox* – це властивість *Text*. Її значення може змінитись при уведенні в поле тексту з клавіатури або внаслідок присвоєння. З елементом *TextBox* пов'язані дві події, за допомогою яких можна керувати значенням властивості *Text*:

Change (Змінити) – подія відбувається, коли змінюється властивість *Text*;

LostFocus (Втрата фокуса) – подія відбувається, коли користувач закінчує роботу з текстовим полем і викликає подію на іншому елементі керування.

В обробниках цих подій організовують перевірку коректності уведених даних, змінюють стан інших елементів керування тощо.

Властивості текстового поля. Основна властивість − ***Text*** − зберігає рядок тексту, який відображає текстове поле. Щоб текстове поле відразу після запуску проекту було порожнім, у вікні *Properties* слід видалити значення цієї властивості.

Властивість ***Font***, як і в елементу керування *Label*, визначає шрифт, яким відображається текст. Від значення властивості ***Alignment*** залежить горизонтальне вирівнювання тексту: 0 − за лівим краєм, 1 − за правим краєм, 2 − по центру.

Спільна з елементом керування *Label* властивість ***BorderStyle*** визначає тип межі. В елемента керування *Label* початкове значення цієї властивості − *None* (відсутня), а в елемента керування *TextBox* встановлюється значення *Fixed Single* (фіксована одинарна). Кольори залежать від значень властивостей ***BackColor*** (колір фону) і ***ForeColor*** (колір тексту).

Якщо властивості ***MultiLine*** (багаторядковість) надати значення *True*, то текст буде виводитись в декілька рядків. При цьому можуть знадобитись смуги прокручування, наявність та кількість яких залежить від значення властивості ***ScrollBars*** (0 – смуг немає, 1 − горизонтальна, 2 – вертикальна, 3 – обидві).

Як бачимо, елемент керування *TextBox* легко пристосувати для виконання різних задач.

Перетворення типів у Visual Basic

Досить часто у програмах використовують таку логіку роботи. Для **введення** користувачем різних символів, у тому числі і цифр, використовують текстове поле (*TextBox*). Потім комп'ютер виконує математичні **обчислення**, а результат **виводить** за допомогою елемента керування *TextBox* або *Label*.

В цій ситуації виникає проблема: математичні операції можна виконувати тільки з числами (наприклад, значеннями типу *integer*), а властивість *Text* елементу керування *TextBox* має рядковий тип (*string*). Потрібно мати спосіб перетворення рядків у числа і навпаки, чисел у рядки.

Для розв'язання цієї проблеми існують вбудовані функції мови Basic: функція ***Val*** і функція ***Str*** (табл. 1.8).

Функція *– це процедура, яка в результаті виклику повертає деяке значення.*

Завдяки цьому виклики функцій можна використовувати у виразах.

Таблиця 1.10

Функція	Призначення	Формат функції	Приклад використання
Val	повертає числове значення, записане в текстовому рядку	*Val(YourString)*	*YourNumber = Val("23")* Змінна YourNumber отримує числове значення 23.
Str	повертає рядок, що зображає вказане число.	*Str(YourNumber)*	*YourString = Str(23)* Змінна YourString отримує рядкове значення “23”

Приклад 1.25. Додавання чисел *A* і *B*, значення яких вводиться в текстові поля *Text1* і *Text2*. Значення суми *C* виводиться в текстове поле *Text3*.

A = Val (Text1.Text)
B = Val (Text2.Text)
C = A + B
Text3.Text = Str (C)

Питання для самоконтролю

1. *Назвіть основні характеристики величини.*
2. *Які імена недопустимі в якості ідентифікаторів та чому: summa; W1; Priklad 1; (SUM); A-4; Priklad_1; Dim; 9A?*
3. *Що таке тип величини? Назвіть основні типи даних у мові Visual Basic.*
4. *Що таке вид величини? Назвіть і охарактеризуйте види величин.*
5. *Як відбувається оголошення змінних у мові Visual Basic? Які способи існують для оголошення змінних?*
6. *Запишіть оператори оголошення змінних A цілого типу, B логічного типу, C рядкового типу.*
7. *Вкажіть типи констант: а) 3; б) 300; в) 5.4; г) True; д)”парта”; е)”324”.*
8. *Який синтаксис має оператор присвоєння? Опишіть схему його виконання.*
9. *Чому дорівнює значення X після виконання послідовності присвоювань:*
   а) y = 2; x = y;
   б) x = 8; x = x+2;
   в) x = 5; x = -x;
   г) x=10; x=x +3?
10. *Запишіть оператори присвоєння, що реалізують такі дії:*
    а) змінній S присвоїти значення суми змінних A і B;
    б) обчислити значення добутку змінних A і B і результат присвоїти змінній C;
    в) подвоїти значення змінної A;
    г) змінити знак змінної t.
11. *Поясніть дію операторів:*
    а) Form1.Top = Form1.Top + 300;
    б) Form1.Top = 300.

12. *Для чого використовується елемент керування TextBox? Назвіть основні властивості елемента керування TextBox.*
13. *Як виконати перетворення рядкового значення в числове і навпаки? В яких випадках це необхідно?*
14. *Запишіть оператор присвоєння змінній A дійсного типу значення, яке вводиться в текстове поле Text1.*
15. *Опишіть призначення функцій Val та Str.*

1.5. Організація введення і виведення даних

Описану вище логіку роботи комп'ютерної програми можна узагальнити. Більшість програм передбачають введення (***I****nput*) даних користувачем, після чого над ними будуть виконані деякі операції (***P****rocessing*), а результат операцій виведений (***O****utput*) у тому чи іншому вигляді.

Принцип IPO обробки інформації:
введення ⇨ опрацювання ⇨ виведення.

Розглянемо детальніше засоби введення та виведення даних. Нехай змінну *A* оголошено в такий спосіб: *Dim A As Single*. Присвоїти змінній *A* значення можна:

- за допомогою оператора присвоєння;
  A = 3.25
- шляхом введення значення у текстове поле і присвоєння змінній значення властивості *Text*;
  A = Val (Text1.Text)
- за допомогою функції InputBox.
  A = InputBox ("A=?")

Функція InputBox

Для введення даних зручно використовувати функцію *InputBox*. Функція *InputBox* відображає вікно діалогу і очікує, поки користувач введе дані в текстове поле і натисне кнопку. Після цього функція повертає введений текст у вигляді рядка (*String*). Стандартними елементами вікна є кнопки *OK* (підтвердження дії) і *Cancel* (відміна дії). Формат використання функції *InputBox* такий:

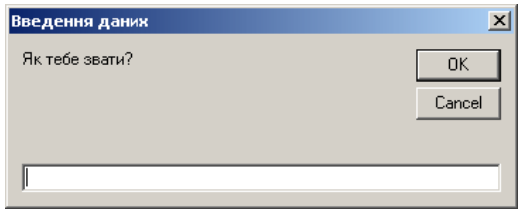
Ім'яЗмінної = InputBox (***підказка***, [***заголовок***]), де:

- *Ім'яЗмінної* – змінна, яка отримує значення, повернуте функцією; перетворення типу повернутого значення до типу змінної відбувається автоматично;
- *підказка* – повідомлення, яке буде виведене у діалоговому вікні функції;
- *заголовок* – заголовок вікна (необов'язковий параметр).

Приклад 1.26. За допомогою функції *InputBox* можна організувати такий діалог з користувачем:

```
Dim D As String
D = InputBox("Як тебе звати?")
Text1.Text ="Привіт, " &  D
```

Тут символом & позначено операцію об'єднання (конкатенації) рядків (детальніше − далі).



Для виведення в попередніх пунктах ми застосовували елементи керування *TextBox*, *Label*. Але не завжди зручно створювати окремий елемент для виведення кожного повідомлення або кожного результату опрацювання даних. **Visual Basic** дозволяє виводити повідомлення в будь-який момент виконання програми за допомогою діалогового вікна ***MsgBox*** та виводити результати роботи програми безпосередньо на форму за допомогою методу ***Print***.

Діалогове вікно MsgBox

Функція *MsgBox* дозволяє вивести на екран діалогове вікно, придатне для відображення будь-якого повідомлення.

Тут і далі у квадратних дужках вказані назви не обов'язкових елементів.

Функція має такий формат:

***MsgBox підказка*, [*кнопки*], [*заголовок*]**, де:

- *підказка* – рядкова константа або змінна для відображення в діалоговому вікні;
- *кнопки* – числовий вираз, який визначає кількість і типи кнопок або малюнок у вікні. Якщо параметр не вказано, то використовується значення 0 (кнопка *ОК*);
- *заголовок* – рядкове значення, яке задає заголовок вікна.

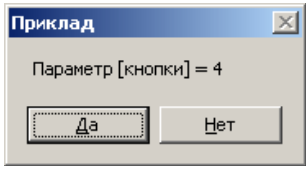
Якщо один з параметрів (крім останнього) відсутній, коми, що відокремлюють його від інших, слід зберігати.

Таблиця 1.11.

Основні константи діалогу *MsgBox*			
Значення	**Опис**	**Значення**	**Опис**
0	Кнопка *OK*	16	Знак помилки
1	Кнопки *OK* і *Cancel*	32	Знак питання
3	Кнопки *Yes*, *No* і *Cancel*	48	Знак оклику
4	Кнопки *Yes* і *No*	64	Знак інформації

Приклад 1.27. При використанні вказаних значень параметрів діалогове вікно має наведений на малюнку вигляд:

MsgBox "Параметр [кнопки] = 4", 4, "Приклад"



Застосування методу Print

Для виведення на формі або на малюнку тексту застосовується метод ***Print***. Синтаксис методу:

[*об'єкт*.]*Print* [*список виведення*],

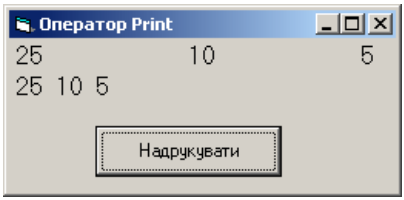
де *об'єкт* – це форма або елемент керування типу *PictureBox* (детальніше про нього − далі), а *список виведення* – перелік текстових рядків або виразів.

Якщо друкування здійснюється на поточній формі, як це було в попередніх прикладах, ім'я об'єкта не вказують.

Якщо в списку виведення декілька значень, їх розділяють комою або крапкою з комою. Значення, відокремлене від попереднього комою, буде надруковане в наступній колонці (ширина колонки 14 символів). Якщо ж числові значення відокремлюються крапкою з комою, то перед і після них додаються пропуски. Рядкові значення в такому разі друкуються без пропусків.

Приклад 1.28.

A = 25 : B = 10 : C = 5
Print A, B, C
Print A; B; C;



Розділювач, поставлений в кінці оператора, впливає на наступні оператори *Print*. Якщо ж розділювача немає, то наступний оператор *Print* виводить дані з нового рядка.

Відокремлювати комою чи крапкою з комою можна і порожні значення.

Як ви вже знаєте, вигляд тексту на формі залежить від установок властивості *Font*. Формат доступу до параметрів шрифту з програмного коду має такий вигляд:

Font.параметр = значення

Приклад 1.29. Наведені оператори друкують рядок «ABC» шрифтом *Arial*, розміром 20 пунктів, накреслення напівжирне:

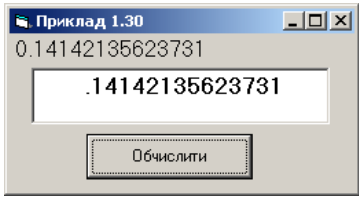
```
Font.Size = 20
Font.Name = "Arial"
Font.Bold = True
Print "ABC"
```

Виведення значень змінних типу Single

При виведенні значень змінних типу *Single* можна одержати значення, яке незручне для читання.

Приклад 1.30. Випробуйте процедуру, в якій обчислюється і виводиться значення $\sqrt{2}/10$:

```
Private Sub Command1_Click()
    a = 2
    Print Sqr(a) / 10
    Text1.Text = Str(Sqr(a) / 10)
End Sub
```



Як бачимо, оператор *Print* виводить забагато десяткових цифр (припустимо, що за умовою задачі така точність обчислень нас не цікавить), а в текстовому полі ще й не показано цифру нуль. Для управління виведенням числових значень існує функція ***Format(),*** яка має такий синтаксис:

Format (***вираз***, ***шаблон***),

де *вираз* – числовий вираз; *шаблон* – рядок-шаблон, який визначає вигляд рядка-результату.

Параметр *шаблон* беруть в лапки, оскільки він являє собою рядок символів.

Для створення шаблона використовуються символи:

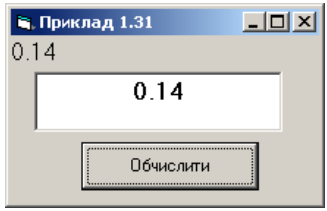
- # – відображення значущих цифр;
- 0 – відображення необхідної кількості цифр. Замість відсутніх значущих цифр будуть виведені нулі.

Приклад 1.31. Якщо у процедурі з прикладу 1.30 додати виведення значення виразу за форматом, отримаємо наведений на малюнку результат:

Print Format(Sqr(a) / 10, "#0.00")

Text1.Text = Format(Sqr(a) / 10, "#0.00")

Зверніть увагу: додаткове перетворення числового значення в рядкове при виведенні в текстове поле не потрібне.



Приклад 1.32. Щоб значення змінної *Res* було виведене з трьома цифрами після коми, використовують виклик функції *Format(Res, "#0.000")*.

Якщо *Res=235.6*, то буде отримано рядок 235.600.

Якщо *Res=0.6789564*, то буде отримано рядок 0.679.

Коментарі

Коментарі – це пояснення до рядків вашого коду, які допомагають зрозуміти, як працює програма. Компілятор їх ігнорує. З коментарями простіше згодом вносити зміни в програму.

Відсутність коментарів – ознака недбалості або невисокої кваліфікації програміста!

Для розміщення коментаря в рядку застосовується апостроф ('): всі символи від апострофа і до кінця рядка вважаються коментарем. Коментар може займати цілий рядок, наприклад:

' Змінити колір форми на червоний

Form1.BackColor = vbRed

або бути поміщеним в одному рядку з оператором:

Form1.BackColor = vbRed ' Змінити колір форми на червоний

Питання для самоконтролю

1. *Як задати значення змінної з використанням функції InputBox?*
2. *Поясніть синтаксис функції MsgBox. Які значення повертає функція MsgBox?*
3. *Наведіть приклади ситуацій, у яких доцільно використовувати діалогове вікно MsgBox.*
4. *Поясніть призначення кожного параметру в операторі*
   MsgBox Str(V) & " грн", , "Загальна сума"
   Чому в списку параметрів стоїть дві коми поспіль?
5. *Поясніть правила використання методу Print.*
6. *Які можливості існують для позиціонування виведення значень на екран при застосуванні методу Print?*
7. *Як змінити параметри шрифту при виведенні тексту?*

8. *Яка функція використовується для керування виведенням числових значень?*
9. *В ході виконання програми змінна А отримала значення 12.567. Яке значення буде виведене в текстове поле Text1 після виконання такого оператора:*
   а) Text1.Text = Format(A, "#0.00");
   б) Text1.Text = Format(A, "#0.0000")?
10. *Для чого у програмний код вміщують коментарі?*
11. *Як додати коментар в рядка програмного коду?*
12. *На початку рядка програмного коду стоїть апостроф. Що це означає?*

1.6. Стандартні функції мови Basic. Таймер

Стандартні математичні функції

У **Visual Basic** є низка функцій для розв'язування математичних задач. Їх можна використовувати безпосередньо при обчисленні значень будь-яких виразів.

У таблиці 1.12 наведено синтаксис і короткий опис математичних функцій, які використовуються найчастіше.

Таблиця 1.12

Запис на Visual Basic	Призначення	Тип результату
Abs(X)	$\|X\|$	збігається з типом аргументу *X*
Atn(X)	*arctg X*	*Double*
Cint(X)	зводить значення *X* до типу *Integer*	*Integer*
Cos(X)	*cos X* (аргумент в радіанах)	*Double*
Exp(X)	e^X	*Double*
Fix(X)	відкидання дробової частини числа *X*	*Integer*
Int(X)	ціла частина числа *X* – найбільше ціле число, що не перевищує *X*	*Integer*
Round(X,a)	округлення числа *X* з точністю до *a* цифр після коми	*Single*
Rnd(X)	генератор випадкових чисел	*Single*
Log(X)	*ln X*	*Double*
Sgn(X)	знакова функція $=\begin{cases} 1 \text{ при } X>0 \\ 0 \text{ при } X=0 \\ -1 \text{ при } X<0 \end{cases}$	*Variant*
Sin(X)	*sin X* (аргумент в радіанах)	*Double*
Sqr(X)	$\sqrt{X}$	*Double*
Tan(X)	*tg X* (аргумент в радіанах)	*Double*

Аргумент в усіх тригонометричних функціях задається в радіанах, а не в градусах. При необхідності переведення значення градусів в радіани слід використовувати формулу:

радіани = градуси $\times \pi$ /180.

Функція *Rnd(X)* – генератор випадкових чисел – повертає довільне число з інтервалу (0;1), при цьому аргумент *X* можна не вказувати. Для отримання при кожному запуску програми різних послідовностей випадкових чисел, необхідно перед першим викликом функції *Rnd* викликати процедуру *Randomize* з параметром *Timer* (запуск генератора випадкових чисел).

Приклади використання випадкових чисел: ігрові програми, в яких потрібне щоразу інше розміщення певних елементів (карт, фішок) або непередбачуваний хід комп'ютера; імітація різних природних процесів, що мають випадковий характер; підпрограми генерування паролів у системах реєстрації користувачів тощо.

Приклад 1.33. Для одержання випадкових цілих чисел з заданого діапазону використовують такий прийом: множать значення *Rnd* на кількість цілих чисел в діапазоні, беруть від отриманого добутку цілу частину і додають перше з чисел діапазону. Наприклад, щоб одержати випадкове ціле число *X*, що належить проміжку від 0 до 99, користуються операторами:

```
Randomize Timer
X = Int (100 * Rnd)
```

А число з проміжку від 15 до 75 отримують операторами:

```
Randomize Timer
X = 15 + Int (61 * Rnd)
```

Приклад 1.34. Дійсне значення перетворюють в ціле за допомогою стандартних функцій *Fix(X)* та *Int(X)*.

Нехай $X = 4.65$. Тоді *Fix (X)* =4; *Int (X)* = 4.

Нехай $X = -4.1$. Тоді *Fix (X)* = −4; *Int (X)* = −5.

Правила запису арифметичних виразів:

1. Вираз повинен бути записаний у вигляді рядка символів, так як записують, наприклад, формули в електронних таблицях.
2. Не можна опускати знак операції множення.
3. Порядок виконання операцій одного пріоритету регулюється дужками.
4. Аргументи функцій записуються в круглих дужках.

Приклад 1.35.

Арифметичний вираз	Запис виразу у програмі
$\frac{2x-5}{3+x}$	*(2*x-5)/(3+x)*
$b+\sqrt{a}$	*b+sqr(a)*
$\cos^2 x$	*Cos(x)^2*
$\cos x^2$	*Cos(x^2)*

Функції для роботи з датою та часом

Приклад 1.36. Присвоєння значень змінним типу *Date*:

BirthDay = #29.10.2008#

EndOfTime = #8:30#

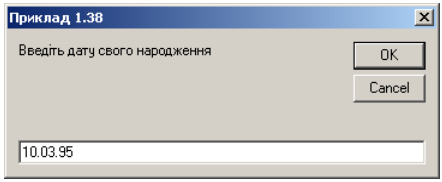
У таблиці 1.13 наведено синтаксис і короткий опис деяких функцій для роботи з датою та часом.

Таблиця 1.13

Запис на Visual Basic	Призначення	Тип результату
Date	Значення поточної дати	*Date*
Time	Значення поточного часу	*Date*
Now	Значення поточної дати і поточного часу	*Date*
Hour (time)	Кількість годин в часі *time*	*Integer*
Minute (time)	Кількість хвилин в часі *time*	*Integer*
Second (time)	Кількість секунд в часі *time*	*Integer*
WeekDay (date)	Номер дня тижня, який указано в аргументі *date*	*Integer*
WeekDayName (n)	Назва дня тижня, який має номер *n*	*String*
DateDiff (S, D1, D2), де *S* – параметр, який може приймати значення: *"d"* – доби; *"h"* – години; *"n"* – хвилини; *"s"* – секунди.	Кількість одиниць часу *S*, що пройшла з часу *D1* до часу *D2*	*Integer*

Приклад 1.37. Відображення поточної дати у текстовому полі:

Text1.Text = Date



Приклад 1.38. В полі *Text1* буде виведено кількість днів, яку прожив користувач.

D = InputBox("Введіть дату свого народження")

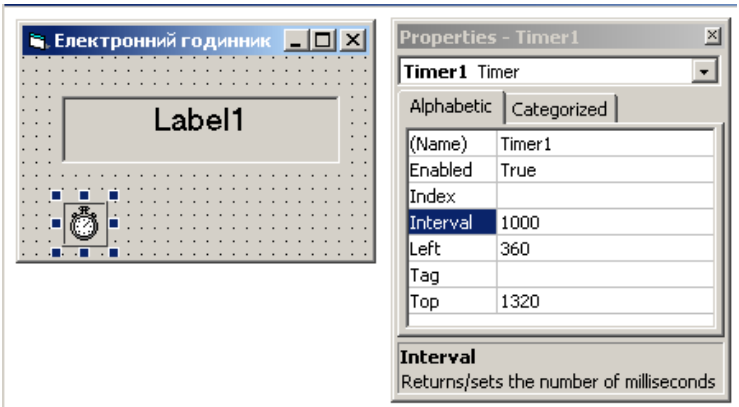
Text1.Text =DateDiff ("d", D, Now)

При введенні значень змінних типу Date символи «#» писати не треба. Число, місяць та рік відокремлюють крапками.

Елемент керування Timer

Елемент керування *Timer* дозволяє виконувати через певні проміжки часу деякі операції, які не залежать від дій користувача під час роботи програми. Для вмикання таймеру потрібно його властивості *Enabled* надати значення *True*. Час між подіями, які викликає об'єкт *Timer*, визначається властивістю *Interval*, якій можна надавати значення в діапазоні [0; 65535] (мілісекунд), тобто подія *Timer* не може виконуватись рідше, ніж 1 раз у 66 секунд.

Приклад 1.39. Щоб створити на формі «електронний годинник», який відображатиме поточний час, оновлюючись кожну секунду, потрібно помістити на форму елемент Таймер і властивості *Interval* надати значення *1000* (1 секунда). Для



відображення показників часу на формі розмістити елемент *Label.* Далі слід запрограмувати обробку події «Спрацювання таймера». Для цього двічі клацають об'єкт *Timer1* і у процедурі обробки події, що має назву *Timer1_Timer*, вписують рядок:

Label1.Caption = Str(Time)

Призначення даної процедури – змінити значення властивості *Caption* об'єкта *Label1*.

Об'єкт типу Timer не відображається при роботі програми.

Функції для роботи з даними рядкового типу

Велика кількість програм призначені для роботи з текстами: текстові редактори, перекладачі, поштові клієнти, браузери тощо. Для опрацювання текстової інформації у **Visual Basic** існує тип даних *String*. Як ви вже знаєте, значеннями типу *String* є рядки символів, а записуючи рядкові константи, їх беруть у лапки, наприклад: "Добрий ранок", "5a+c" (див. табл. 1.5, 1.6). Найпростіші дії з рядками вже зустрічались вище під час організації виведення на екран.

Рядок є послідовністю символів. Нумерація символів у рядку починається з 1.

Символи в тексті програми, взяті в лапки, сприймаються **Visual Basic** як рядкові значення. Символи, які не взяті в лапки, розглядаються як змінні, ключові слова тощо.

Приклад 1.40. Присвоєння значень рядковим змінним:

Dim A As String, B As String

A = "31 грудня"

B = "75"

Якщо потрібно обмежити довжину рядка конкретною кількістю символів, використовують опис виду:

Dim Ім'яЗмінної As String * довжина

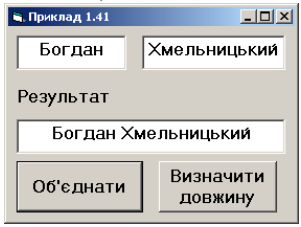
де *довжина* – кількість символів у рядку. Отже, щоб створити рядкову змінну довжиною 10 символів, її оголошують таким чином:

*Dim A As String * 10*

Конкатенація рядків

Конкатенацією двох або більше рядків називають їх об'єднання в один рядок. Цю операцію позначають символом **&**. Операція **&** дозволяє об'єднувати в рядок як змінні, так і константи рядкового типу.

Знак **&** слід відокремлювати пропуском від розміщеного перед ним імені змінної.



Приклад 1.41. Для об'єднання двох рядків, які введені в текстові поля *Text1* і *Text2*, з додаванням пропуску між ними,

в обробник події для кнопки з написом «Об'єднати» (див. мал.) вміщують такі рядки:

A = Text1.Text & " " & Text2.Text

Text3.Text = A

або

Text3.Text = Text1.Text & " " & Text2.Text

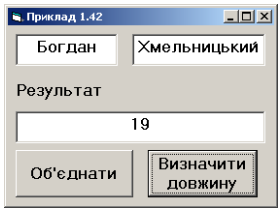
Визначення довжини рядка

Функція ***Len()*** повертає число символів, які містить рядок. Синтаксис функції:

Len (рядок),

де *Len* – ім'я функції; *рядок* – рядкова змінна, константа або вираз, довжину значення якого потрібно визначити.

Приклад 1.42. При натисканні кнопки «Визначити довжину» мають виконатися такі команди:



K = Len(A)

Text3.Text = Str(K)

або

Text3.Text = Str(Len(A))

Одержання підрядка за допомогою функції Mid()

Підрядком називають довільну частину рядка. Функція ***Mid(strS, i, N)*** повертає *N* символів рядка *strS*, починаючи з і-го символу.

Приклад 1.43. Після виконання команд:

A ="Visual Basic"

B =Mid (A, 5, 6)

Змінна B отримає значення *«al Bas»*.

Значення, яке задає початкову позицію, повинне бути не менше 1. Якщо не вказана кількість символів, які слід повернути, функція *Mid()* поверне частину вихідного рядка, починаючи з вказаної початкової позиції і до кінця.

Результат, повернутий функцією Mid(), можна вивести або присвоїти змінній, а початковий рядок залишається без змін.

Приклад 1.44.

Dim A As String, B As String

A = "програмування"

B = Mid (A, 1, 2) & Mid (A, 6, 1) & Mid (A, 11,3)

Питання для самоконтролю

1. *Які математичні функції призначені для перетворення значень дійсного типу в значення цілого типу?*
2. *Поясніть правила запису арифметичних виразів у програмі.*
3. *Як отримати випадкові числа? Наведіть приклади задач, для яких потрібні випадкові числа.*
4. *Запишіть оператор присвоєння, який реалізує таку дію: змінній B типу Single присвоїти значення дробової частини дійсного числа A.*
5. *X=3,567. Чому дорівнює Fix(X), Int(X), Round(X,1)?*
6. *X=–3,567. Чому дорівнює Fix(X), Int(X), Round(X,1)?*
7. *Запишіть на Visual Basic такі формули:*

   а) $\frac{x-5}{2x}$; *б)* x^5 ; *в)* $\sqrt[3]{1+x}$; *г)* $\cos^2 x$.

8. *Які функції призначені для роботи з датою та часом?*
9. *Які символи використовуються для присвоєння значень змінним типу Date?*
10. *Як виконують об'єднання рядків?*
11. *За допомогою яких засобів можна виконати копіювання підрядка із заданого рядка? Поясніть на прикладі.*
12. *Для чого використовується елемент керування Timer? Опишіть схему його застосування.*

1.7. Графічні елементи керування

Додавання зображень на форму

Для графічного оформлення форми застосовують елементи управління ***Image*** і ***PictureBox***. Елемент управління *Image* служить для розміщення на формі графічного зображення. Елемент управління *PictureBox* є свого роду вікном, можливо, з малюнком на фоні, тобто є, перш за все, контейнером для розміщення інших об'єктів.

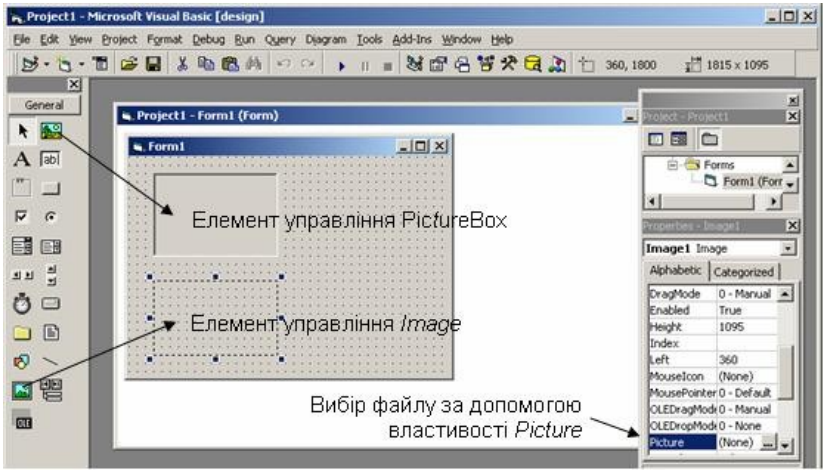
Кожен з елементів додається на форму стандартним способом: необхідно або двічі клацнути потрібний елемент, або, виділивши його піктограму на панелі інструментів, намалювати елемент на формі за допомогою миші.

Відмінність в зовнішньому вигляді елементів після додання на форму викликане тим, що властивість *BorderStyle* для елемента *PictureBox* спочатку має значення 1 – *Fixed Single*, а для елемента *Image* – значення *0 – None*.

Зображення для обох елементів керування визначається властивістю *Picture*, значенням якої є ім'я графічного файлу. Для вибору файлу у вікні *Properties* у рядку *Picture* треба натиснути кнопку ... . Після цього з'явиться стандартне вікно для вибору

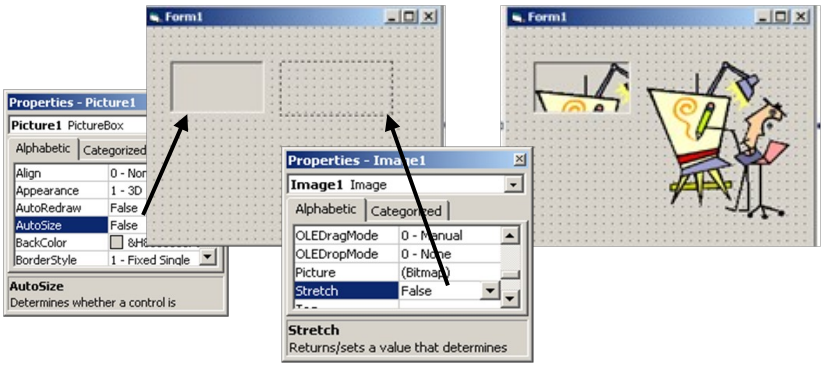
файлу. Підтримуються всі основні формати графічних файлів, такі як *.bmp, .gif, .jpg* та інші.

Елемент управління *PictureBox* має властивість логічного типу *AutoSize*, що дозволяє або забороняє автоматично змінювати розміри елемента управління відповідно до розмірів зображення.



Елемент управління *Image* має властивість логічного типу *Stretch*, яка керує зміною розмірів зображення відповідно до розмірів елемента управління.

Значення *True* властивостям *AutoSize* або *Stretch* необхідно надати перед тим, як завантажувати зображення.



Приклад 1.45. Якщо згадані властивості автомасштабування елементів керування *PictureBox* і *Image* мають значення *False*, а розміри малюнка перевищують розміри елементів, то при

завантаженні зображення отримаємо такий результат: в *PictureBox* відобразиться тільки частина малюнка, а розміри *Image* збільшаться до розмірів зображення (див. мал.).

Завантаження зображення під час виконання програми

Часто виникає потреба змінювати зображення під час виконання програми. Таку можливість забезпечує функція ***LoadPicture()***. Синтаксис функції такий:

VarPicture = LoadPicture (FileName),

де *VarPicture* – змінна або властивість для збереження малюнка; *FileName* – рядкова змінна або константа, яка містить шлях до графічного файлу на диску.

Якщо графічний файл знаходиться в поточному каталозі (у папці проекту), достатньо вказати лише його ім'я.

Проект встановлює зв'язки з файлами в папці проекту в момент відкриття, тому графічні файли потрібно додати до папки проекту до відкриття (або створення нового) проекту.

Приклад 1.46. Наступний оператор ставить у відповідність об'єкту *Image1* малюнок з колекції *Clipart*:

Image1.Picture = LoadPicture ("C:\Program Files\Microsoft Office\Clipart\Pub60cor\ BL00261_.Wmf")

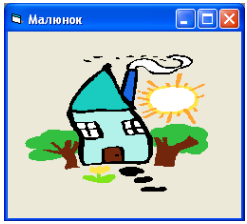
Якщо функція *LoadPicture()* буде викликана без зазначення шляху до файлу, то відбудеться очищення даного елемента від малюнка. Подібного ефекту можна досягти, викликавши для елемента методу *Cls*.

Приклад 1.47. Такі рядки коду реалізують однакові дії – очищують вміст елемента *Image1*:

Image1.Picture = LoadPicture ()

або

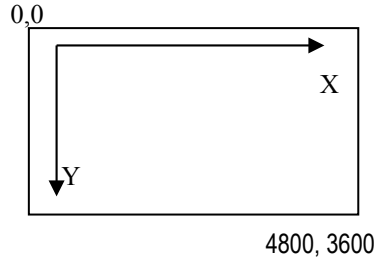
Image1.Cls



Система координат

Для визначення положення на формі графічного елемента використовують координати. Стандартна система координат **Visual Basic** передбачає використання вже відомих вам одиниць вимірювання – твіпів. Один ***твіп*** дорівнює 1/20 пункту чи 1/1440 дюйма. Будь-яка точка на формі задається парою чисел *(X, Y)*. Вертикальна координата *Y* зростає з верху до низу, а горизонтальна

X – зліва направо. Поточні розміри форми зазначені у правій частині панелі інструментів *ToolBar*. Початкові розміри форми становлять 4800×3600 твіпів. Один твіп – це дуже маленька відстань: погляньте на форму в режимі [*design*] – вона вся замальована чорними крапками. Відстань від крапки до крапки – 120 твіпів!



Елементи керування Shape і Line

Панель *Toolbox* містить два елементи керування: *Line* (Лінія), що має вигляд відрізка прямої певного кольору, товщини і стилю, та *Shape* (Фігура), що може мати вигляд прямокутника, круга або еліпса. Ці елементи використовуються для створення графічних композицій на формі.

Використання елемента керування Shape

На панелі компонентів виберіть компонент ***Shape*** і намалюйте його на формі перетягуванням. Потім налаштуйте властивості фігури:

1. Виберіть форму фігури зі списку властивості *Shape*:

Допустимі значення властивості *Shape*			
0	Прямокутник	3	Круг
1	Квадрат	4	Округлений прямокутник
2	Овал	5	Округлений квадрат

2. Задайте заповнення фігури. Для властивості ***FillStyle*** (Спосіб заповнення) виберіть зі списку потрібне значення:

Допустимі значення властивості *FillStyle*			
0	Суцільне	4	Діагональ вгору
1	Прозоре	5	Діагональ вниз
2	Горизонтальні лінії	6	Клітинки
3	Вертикальні лінії	7	Діагональні клітинки

Колір зафарбування задається властивістю ***FillColor*** (Колір заповнення). Розкрийте список і виберіть на вкладці *Palette* (Палітра) потрібний колір.

3. Задайте спосіб взаємодії заповнення і тла. Якщо властивість *Backstyle*=0, то простір між лініями заповнення залишиться

прозорим. При *Backstyle*=1 прозорі ділянки заповняться кольором. вказаним у властивості *BackColor*.

4. Для точного задання розмірів фігури використовуйте властивості *Height* (Висота) і *Width* (Ширина).

5. Для точного розміщення фігури на формі використовуйте властивості *Left* (Відстань від лівого краю форми) і *Top* (Відстань від верхнього краю форми).

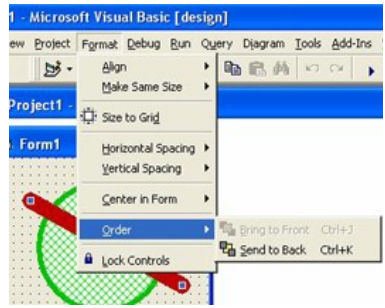
Використання елемента керування Line

На панелі компонентів виберіть компонент *Line*  . Помістіть вказівник на те місце форми, де повинна починатися лінія, і, натиснувши ліву кнопку, перетягніть вказівник до кінцевої точки лінії. Відпустіть кнопку мишки. На формі з'явиться відрізок прямої. Кінці відрізка можна переміщати за маркери, а увесь відрізок − перетягуючи за будь-яку іншу точку.

Приклад 1.48. Для даних об'єктів *Shape* і *Line* в режимі [*design*] у вікні *Properties* встановлені такі значення властивостей:

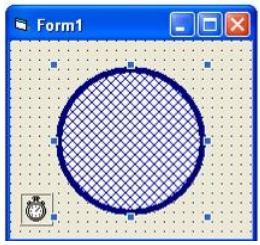
Властивість	Об'єкт *Shape*	Об'єкт *Line*
Shape	3 – Circle	–
BorderWidth	5	15
BorderColor	&H0000C000&	&H000000C0&
X1, Y1	–	480, 600
X2, Y2	–	2600, 2000
BackStyle	1 – Opaque	–
FillStyle	7 – Diagonal Cross	–
FillColor	&H0000FF00&	–

Щоб змінити порядок розташування об'єктів на формі, необхідно виділити об'єкт, який треба перемістити, і вибрати команду *Format* ⇨ *Order* ⇨ *Send To Back* (Перемістити назад) або *Bring To Front* (Перемістити вперед).



Значення властивостей об'єктів *Shape* і *Line* можна змінювати програмним способом.

Приклад 1.49. При спрацьовуванні таймера, випадковим чином змінюються значення властивостей об'єкта *Shape*. Діапазон для генерації випадкового числа обирається в залежності від діапазону допустимих значень тієї чи іншої властивості (наприклад, тип фігури визначається значенням від 0 до 5, отже, інтервал для випадкового числа обмежується числом 6):




```
Private Sub Timer1_Timer()
    Randomize Timer
    Shape1.Shape = Int(6 * Rnd)
    Shape1.FillStyle = Int(8 * Rnd)
    Shape1.Width = Int(2000 * Rnd)
End Sub
```

Задання кольорів

Як ви вже знаєте, при виведенні зображення на екран (у телевізорах, комп'ютерних моніторах тощо) кожен колір – це «суміш» у певній пропорції 3-х базових кольорів – червоного, зеленого і синього.

У програмному коді колір малювання задають за допомогою функції *RGB*, в аргументах якої окремо вказується ступінь насиченості кожної складової (у межах від 0 до 255).

Таблиця 1.14

Основні кольори							
Колір	**R(ed)**	**G(reen)**	**B(lue)**	**Колір**	**R**	**G**	**B**
чорний	0	0	0	жовтий	255	255	0
білий	255	255	255	фіолетовий	255	0	255
червоний	255	0	0	коричневий	205	155	135
зелений	0	255	0	жовтогарячий	255	128	0
синій	0	0	255	сірий	128	128	128

Приклад 1.50. Контур фігури *Shape1* стає коричневим:

Shape1.BorderColor = RGB(205, 155, 135)

Фігура *Shape1* отримує випадковий колір тла:

*Shape1.BackColor = RGB(Int(256*Rnd), Int(256*Rnd), Int(256*Rnd))*

Поряд з використанням функції *RGB* можна, як вже було згадано, застосовувати колірні константи, описані в табл.1.15 (див. приклад в табл.1.6).

Таблиця 1.15

Колірні константи					
Константа	**Значення**	**Опис**	**Константа**	**Значення**	**Опис**
vbBlack	0	чорний	vbMagenta	&HFF00FF	яскраво-червоний
vbBlue	&HFF0000	синій	vbRed	&HFF	червоний
vbCyan	&HFFFF00	блакитний	vbWhite	&HFFFFFF	білий
vbGreen	&HFF00	зелений	vbYellow	&HFFFF	жовтий

Питання для самоконтролю

1. *Які елементи управління використовують для графічного оформлення форми?*
2. *Які дії потрібно виконати, щоб помістити на форму зображення, яке зберігається на диску в графічному файлі?*
3. *Поясніть порядок використання елементів управління Image і PictureBox.*
4. *Назвіть три відмінності між елементами Image і PictureBox.*
5. *Як змінити зображення для елемента Image програмним способом?*
6. *Як змінити розміри зображення програмним способом?*
7. *Запишіть оператор, який надає висоті елемента Image1 значення 200 твіпів.*
8. *Запишіть оператор, який реалізує збільшення відстані від лівого краю до елемента Image1 на 200 твіпів.*
9. *Запишіть оператор, який реалізує додавання зображення до елемента Picture1 з файлу Малюнок1.bmp, який знаходиться на диску C: в папці TEMP.*
10. *Опишіть порядок застосування елементів управління Line та Shape.*
11. *Які властивості притаманні фігурі (Shape)?*
12. *Запишіть оператори, які встановлять суцільне заповнення червоним кольором для фігури Figura1.*

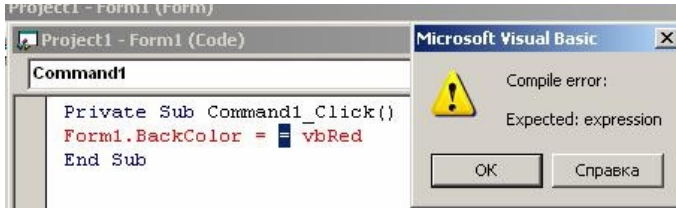
1.8. Налагодження програмного коду

Синтаксичні помилки в програмному коді

Як би ретельно ви не писали текст програми, ви не уникнете помилок в програмному коді. Помилки в написанні або розміщенні ключових слів називаються синтаксичними. Більшість синтаксичних помилок середовище **Visual Basic** дозволяє виправити при наборі програмного коду. Коли ви переводите курсор з рядка, **Visual Basic** вважає, що ви закінчили роботу з рядком, перевіряє його на наявність синтаксичних помилок, і, якщо їх не знаходить, дещо змінює вигляд рядка, приводячи його до стандартного.

Приклад 1.51. Якщо ви робите спробу перевести курсор з рядка, в якому є помилка, підозріле місце буде виділене. Помилку в операторі

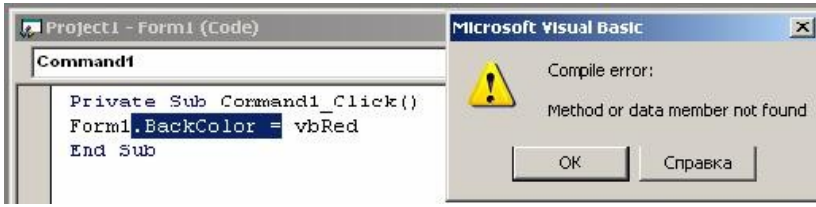
Form1.BackColor = = vbRed



Visual Basic помітить ще в режимі розробки (див. мал.). Натисніть *OK* і виправте помилку.

*Процес усунення помилок називається **налагодженням** програмного коду.*
*Програма, призначена для налагодження програмного коду, називається **налагоджувачем**.*

Приклад 1.52. Налагоджувач **Visual Basic** не завжди правильно виявляє помилки. В даному випадку в імені властивості *BackColor* літера «В» є українською, але система повідомляє, що



для форми даного метода або властивості не існує.

Умовно до синтаксичних можна віднести помилки, які виявляються при компіляції програми (*compile error*). Ці помилки найчастіше виникають, якщо програміст порушує структуру запису складних операторів, або припускає неповний запис програмних блоків операторів.

Приклад 1.53. В даному фрагменті в запису оператора *If Then…Else…End If* відсутній рядок *End If.*

При виправленні помилки, для оперативного одержання довідкової інформації стосовно певного об'єкта, стане в нагоді довідкова система **Visual Basic**.

Інтерактивна довідка

Інтерактивна – тобто діалогова, оперативна, здатна вести діалог і давати підказки в типових ситуаціях.

Щоб звернутись до довідкової системи **Visual Basic** клацніть на слові *Help* у головному меню, потім – на рядку *Contents* (Зміст).

Головною особливістю інтерактивної довідкової системи **Visual Basic** є те, що вона контекстно-залежна.

Приклад 1.54. Виділіть форму і перейдіть у вікно властивостей. Прокручуванням знайдіть властивість *BackColor* і клацніть на ній. Властивість буде виділено. Натисніть клавішу ***F1*** – з'явиться довідкова інформація саме про властивість *BackColor*. Довідкова система розпізнала ваш запит.

Довідка надається за словом, виділеним у вікні властивостей, або за тим, на якому знаходиться курсор у вікні коду. Якщо ж запропонована автоматично інформація не допомогла вирішити проблему, користувач може продовжити роботу з довідковою системою у режимі, подібному до довідкових систем інших програм (**Word**, **PowerPoint** тощо).

Виявлення неоголошених змінних (Option Explicit)

Приклад 1.55. Припустимо, що програміст помилився, набираючи програмний код для надання значень змінним *Wctil* і *Hctil*. В операторі присвоєння він написав *Hcnil* (замість *Hctil*). Система сприймає це як введення нової змінної, значення якої не визначене. Тому в результаті виконання даного коду буде надруковано:

```
Private Sub Command1_Click()
    Wctil = 5:   Hctil = 6
    Pl = Wctil * Hcnil
    Print "Площа = "; Pl
End Sub
```

Площа = 0

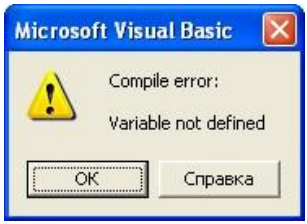
На відміну від інших мов, **Visual Basic** не вимагає оголошувати кожну змінну перед її використанням. Якщо змінна не оголошена, використовується тип даних *Variant* (довільний). Це призводить до зайвої витрати ресурсів пам'яті та непередбачуваних результатів, в зв'язку з використанням значень, заданих за мовчазною згодою.

Щоб уникнути помилкових ситуацій під час роботи зі змінними в програмі, налагодьте **Visual Basic IDE** на необхідність явного оголошення змінних. Для цього:

1) викличте команду меню *Tools / Options*;
2) відкрийте вкладку *Editor* і клацніть на прапорці *Require Variable Declaration* (Вимагати оголошення змінної);
3) натисніть *OK*.

З цього моменту для кожного створюваного проекту у кожен створюваний модуль, у розділ *Declarations,* буде додаватися оператор *Option Explicit*, який свідчить про обов'язковість оголошення всіх змінних в даному модулі (ця опція набере сили з моменту відкриття наступного проекту).

Створіть новий проект і спробуйте ввести з клавіатури оголошення змінних у вікні коду. Зверніть увагу, що у випадку використання в тексті програми неоголошеної змінної при виконанні проекту буде генеруватися помилка. Розглянутий спосіб оголошення змінних називається явним.



Приклад 1.56. Режим явного оголошення змінних допоможе вам запобігти випадкових помилок при наборі. Якщо ви при відсутності оператора *Option Explicit* напишете такий оператор:

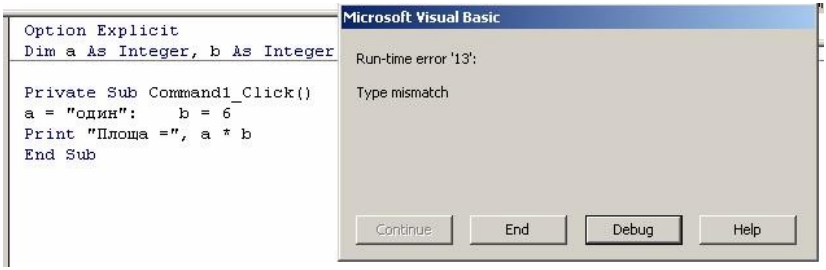
*Form1.BackColor = vbRed**d***

то **Visual Basic** не помітить цієї помилки і зафарбує форму в чорний колір, оскільки буде створено змінну *vbRed**d*** типу *Variant* з числовим значенням 0. А в режимі обов'язкового оголошення змінних *vbRed**d*** буде сприйнято як ім'я неоголошеної змінної, і при виконанні проекту з'явиться повідомлення про помилку.

Помилки виконання

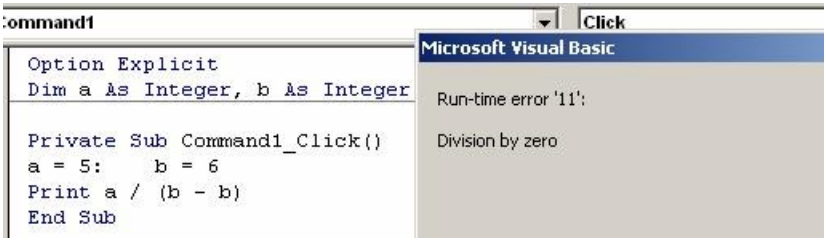
Але не всі помилки в програмі є синтаксичними. Деякі з них виявляються вже при виконанні програми, коли робиться спроба опрацювати неприпустимі дані. Такі помилки називають помилками виконання (*run-time error*).

Приклад 1.57. У наведеному програмному коді робиться



спроба присвоїти змінній, яку оголошену як цілу, значення типу *String*. Тому виконання програми переривається і виводиться повідомлення про невідповідність типів (*Type mismatch*). Якщо натиснути *Debug*, то рядок, в якому виявлено цю помилку, буде позначений жовтим кольором.

Приклад 1.58. В даному випадку відбувається спроба виконати операцію з неприпустимими даними. Виконання програми переривається, виводиться повідомлення *Division by zero* (Ділення на нуль).



Приклад 1.59. На стадії виконання **Visual Basic** помічає такі помилки, як *Sqr(–25)*, і реагує повідомленням про «Неправильний виклик процедури чи неправильний параметр процедури». У нашому випадку це, звичайно, неприпустиме значення параметра (–25). Після натискання на кнопку *Debug* система переходить у режим переривання і підсвічує рядок з помилкою.

Логічні помилки

Найскладніше виправити помилки, викликані неправильною логікою програми або помилками, які припущені при розробці алгоритму. Такі помилки називаються логічними. Програма може бути скомпільована успішно, але результати її виконання виявляються хибними. Так, якщо ви, бажаючи збільшити число *a* на 1, замість *a*=*a*+1 пишете *a*=*a*+2, то, зрозуміло, ні при компіляції, ні при запуску це виявлено не буде. Найпростіший спосіб виявити логічну помилку – це запустити програму в покроковому режимі, і проаналізувати значення змінних на кожному кроці.

Приклад 1.60. Проект «Виконання програми в покроковому режимі»

1. Створіть новий проект. Помістіть на форму командну кнопку, а в вікні програмного коду наберіть таку програму:

Private Sub Command1_Click()
*a = 2 * 3 + 4*

b = a
y = a + b + 1
Print a, b, y, b + y
End Sub

2. Запустіть програму. На формі оператором *Print* буде надрукований рядок чисел:

10 10 21 31

3. Виберіть команду *View* ⇨ *Immediate Window.* Вікно *Immediate* зручно використовувати для перевірки значень змінних в процесі виконання програми. Для виведення значень у вікно *Immediate* призначений оператор ***Debug.Print.***

4. Змініть 5-й рядок програмного коду таким чином:

Debug.Print a, b, y, b + y

Виконайте програму. 4 числа з'явилися у вікні *Immediate*. Тобто ми можемо відстежувати проміжні значення змінних без перевантаження форми текстовими полями або результатами роботи операторів *Print*.

5. Комп'ютер виконує програму дуже швидко. Це добре, якщо результати обчислень нас влаштовують. Але, якщо програма видає хибні результати, бажано відстежити, як після виконання кожного оператора змінюються значення змінних в пам'яті – це допомагає знайти помилку в логіці програми. Для цього і призначений покроковий режим виконання програми.

Запустіть проект на виконання не кнопкою *Start*, як ми звикли, а клавішею *F8* на клавіатурі. Це «гаряча клавіша» для команди *Debug* ⇨ *Step Into.* Проект починає виконуватись. Натисніть кнопку *Command1*. Замість того, щоб повністю виконатися і показати результат, програма зупиняється на першому рядку процедури, а саме, на *Private Sub Command1_Click(),* на ознаку чого цей рядок підсвічено жовтим кольором.

Поцікавтесь, чому під час зупинки дорівнюють значення змінних у пам'яті комп'ютера? Для того, щоб дізнатися про це, наводьте курсор миші на позначення змінної в тексті процедури у вікні коду. Як і слід очікувати, з'являється підказка на зразок «*a=Empty*» − змінні ще не набули значень.

*При натисканні на **F8** налагоджувач виконує черговий оператор програми і підсвічує той оператор, який буде виконаний наступним.*

Натисніть **F8**. Жовта смуга переходить на рядок a = 2 * 3 + 4.

F8. Виконується оператор a = 2 * 3 + 4, підсвічується наступний оператор. Перевірте, чому зараз дорівнюють a, b, y.

F8. Виконується оператор b = a, підсвічується наступний рядок. Перевірте, яких значень набули a, b, y.

Зверніть увагу: режим роботи змінився. У заголовку головного вікна **Visual Basic** ви бачите слово [*break*]. Це означає, що система зараз перебуває у режимі переривання, тобто режимі роботи з зупинками. Натискаючи на *F8*, ви наказуєте **Visual Basic** зупинятися після виконання кожного оператора. Тому такий режим зветься *покроковим режимом* виконання програми.

F8. Виконується оператор y = a+b+1, підсвічується наступний рядок.

F8. Виконується оператор Debug.Print a, b, y, b + y, підсвічується рядок End Sub. У вікні *Immediate* з'явилися 4 числа.

F8. Жовта смуга зникає, тому що процедуру виконано. Можна знову натискати на кнопку *Command1*.

6. В будь-який момент покрокового виконання програми ви маєте змогу замість ***F8*** натиснути *Start*, яка в такому випадку має напис *Continue*, і виконання продовжиться в звичайному режимі.

Питання для самоконтролю

1. *Що буде надруковано при виконанні таких фрагментів програм:*
   *а) a=100 : a=10*a+1 : Debug.Print a;*
   б) a=100 : a= -a : Debug.Print a;
   в) a=10 : b=25 : a=b-a : b=a-b : Debug.Print a, b;
   *г) a = (2^2+1) * (20- (2^2)^2) - 11 : b=11 \ (a-4) : Debug.Print a^2 + b – 1.*
2. *Які помилки називаються синтаксичними? Як система сигналізує про синтаксичні помилки?*
3. *Які помилки належать до помилок виконання?*
4. *Які помилки вважаються логічними?*
5. *Як змінити конфігурацію IDE для автоматичного включення режиму явного оголошення змінних?*
6. *Як здійснити друк значень у вікні Immediate?*
7. *Поясніть схему виконання програми в покроковому режимі.*
   В чому полягають помилки в наведених фрагментах програмних кодів? До якого виду належить кожна з помилок?
8. *Dim a As Integer*
   a = "25 kg" : Print a
   Form1.BackColor = Red
9. *Dim a As Single, x As Single*
   *a = Sqr(Abs(x^2 – 1) / (1+2*x)*
10. *Dim a As Single, b As Single*
    a = 4
    Print a / b
11. *Dim a As Single, b As Single*

```
    Print Sqr(a -  b)
    a = 4 :  b = a*2
12. Dim a As Single, b As Single, c As Single
    a = 3 :  b = 4
    c = Sqr(a^2 -  b^2) 'обчислення гіпотенузи прямокутного трикутника
```

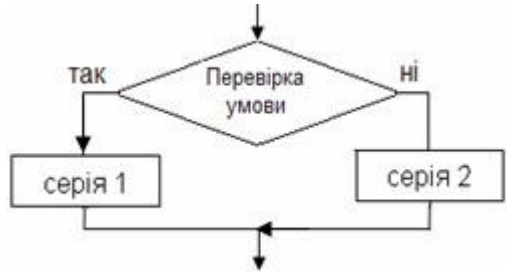
1.9. Вказівка розгалуження

Розгалуження

Розгалуження – це така форма організації дій, при якій, в залежності від виконання або невиконання певної умови, виконується одна з двох послідовностей (серій) дій.

Можна навести багато прикладів вибору дії в залежності від результату виконання умови зі звичайного життя і науки. Наприклад:

- якщо день робочий, то йдемо в школу, інакше будемо відпочивати;
- якщо в рівностороннього чотирикутника кути прямі, то його називають квадратом, інакше його називають ромбом;
- якщо удар пружний, то повна механічна енергія зберігається, інакше − змінюється.



Таку структуру називають повним розгалуженням (структура «ЯКЩО – ТО – ІНАКШЕ»).

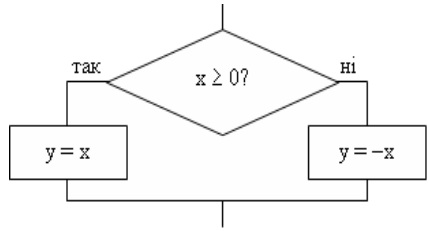
Умова – це твердження, яке може бути істинним чи хибним. Якщо твердження істинне, то вважається, що умова виконана. Якщо умова є істинною, то виконуються команди серії 1 (на блок-схемі − гілка «Так»), якщо ж умова не виконується (умова хибна), – команди серії 2 (гілка «Ні»). Після виконання однієї з серій команд виконавець переходить до наступної після розгалуження команди.

Приклад 1.61. Потрібно побудувати алгоритм обчислення значення функції $y = |x|$. Вона задається співвідношенням

$$y = \begin{cases} x, \text{ якщо } x \geq 0, \\ -x, \text{ якщо } x < 0. \end{cases}$$

При розв’язуванні цієї задачі потрібно виконати такі дії:

1) перевірити, умову чи є $x \geq 0$;
2) якщо $x \geq 0$, то присвоїти y значення x ($y = x$);
3) якщо $x < 0$, то присвоїти y значення, протилежне x ($y = -x$).



Оператор If...Then...Else...End If

Для перевірки істинності умов і організації розгалуження у програмах мовою Visual Basic призначені умовні оператори *If...Then* та *If...Then...Else...End If.*

Оператор If використовує логічні змінні або вирази для запису умови й організації розгалуження в програмі.

Алгоритмічній конструкції «Повне розгалуження» відповідає умовний оператор *If...Then...Else...End If*

Синтаксис оператора:

If <умова> Then
<серія операторів 1>
Else
<серія операторів 2>
End If

або

If <умова> Then
< серія операторів 1>
Else : < серія операторів 2>
End If

Якщо результатом перевірки умови є значення *True*, то виконується блок дій *<оператор 1>*, який знаходиться після ключового слова *Then*. Якщо перевірка умови дала результат *False*, виконується блок дій *<оператор 2>*, вказаний після ключового слова *Else*. В другому з наведених варіантів може використовуватися як один оператор (його можна записати у тому ж рядку, що й *Else*, після знаку «:»), так і декілька (при цьому кожен оператор, починаючи з другого, записується в окремому рядку або теж через двокрапку).

Приклад 1.62. Змінну *A* треба збільшити на 1, якщо її значення ділиться на 3, і зменшити на 1 у протилежному випадку

```
If A Mod 3 = 0 Then
    A = A+1
Else : A = A–1
End If
```

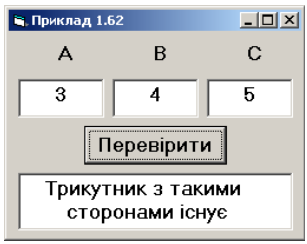
Приклад 1.63. Приклади умов, побудованих з використанням логічних операцій:

Not A <=3 – рівносильна виразу *A > 3*.

Age >= 10 And Age <= 18 – істинна тоді і тільки тоді, коли значення *Age* знаходиться в проміжку від 10 до 18 ($10 \le A \le 18$).

Age < 10 Or Age > 18 – істинна для всіх значень *Age*, які не належать проміжку від 10 до 18.

Приклад 1.64. Фрагмент програми для перевірки існування трикутника із сторонами *a, b, c*. З математики відома умова існування трикутника з певними довжинами сторін: сума двох будь-яких сторін повинна бути більшою за третю.

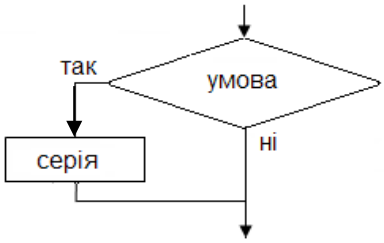



```
A = Val(Text1.Text)
B = Val(Text2.Text)
C = Val(Text3.Text)
If A < B + C And B < A + C And C < A + B Then
    Label4.Caption = "Трикутник з такими сторонами існує"
Else
    Label4.Caption = "Трикутника з такими сторонами не існує"
End If
```

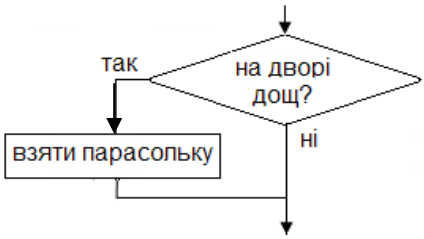
Конструкція If...Then

Досить часто при невиконанні умови не потрібно виконувати ніяких дій. Тоді використовується скорочена форма розгалуження − неповне розгалуження (структура «ЯКЩО-ТО»):

Блок-схема структури "ЯКЩО-ТО"



Приклад 1.65.



Оператор *If...Then* призначений для виконання деякої послідовності дій у тому випадку, якщо вказана в ньому умова є істинною.

Синтаксис оператора:

If <умова> Then <оператор>

Оператор *If* (Якщо) перевіряє істинність зазначеної умови. Якщо умова приймає значення *True* (Істина), програма виконає дію, зазначену в частині <оператор>. Якщо ж умова приймає значення *False* (Хибність), то команди будуть проігноровані і управління передається наступному після *If* оператору.

Значення змінних, що задіяні в логічному виразі, який використовується в якості умови, повинні бути визначені до початку оператора розгалуження.

Приклад 1.66. Нехай потрібно збільшити значення змінної *A* на одиницю, якщо її поточне значення менше 5. Відповідний оператор розгалуження має вигляд:

If A<5 Then A = A+1

Оператор *A = A*+1 виконається тільки в тому випадку, коли істинна умова *A*<5:

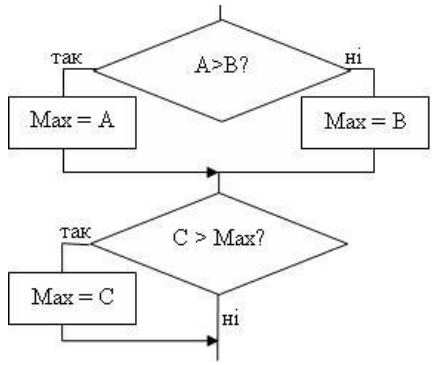
Початкове значення A	Значення умови	Оператор A = A+1	Значення A після виконання оператора If...Then
1	True	виконується	2
5	False	пропускається	5
10	False	пропускається	10

Якщо ж у випадку істинності умови потрібно виконати послідовність дій, оператор *If...Then* записується за таким форматом:

Синтаксис	Приклад

If <умова> Then
<оператор 1>
...
<оператор N>
End If
	If A<5 Then
	A = A+1
	*B = B*2*
	End If

Ключове слово *End If* вказує, де закінчується умовний оператор. Якщо умова хибна, управління передається оператору, записаному після *End If*.



Приклад 1.67. Дано три числа *A, B, C*. Визначити найбільше з цих чисел. Алгоритм пошуку найбільшого з трьох чисел *A, B, C*

реалізується за допомогою послідовного виконання повного та неповного розгалужень.

```
If A > B Then
    Max = A
Else: Max = B
End If
If C > Max Then Max = C
Text4.Text = Str(Max)
```

Елемент CheckBox (прапорець)

Елемент *CheckBox* застосовується, якщо користувачеві необхідно вибрати потрібні серед незалежних параметрів або характеристик. У панелі інструментів елемент керування *CheckBox* має вигляд: 

Під час роботи з елементом *CheckBox* основною властивістю, яка впливає на його відображення, є *Value* (Значення).

Таблиця 1.16

Значення властивості Value		
Значення	**Константа**	**Стан прапорця**
0	*vbUnchecked*	скинутий
1	*vbChecked*	встановлений
2	*vbGrayed*	недоступний

При кожній зміні користувачем стану прапорця, тобто при його встановленні або скиданні, для даного елемента відбувається подія *Click*. Вона виникає або після клацання на ньому лівою кнопкою миші, або після встановлення фокуса клавішею *Tab* з подальшим натисканням клавіші *Пропуск*.

Програмування прапорця. Після подвійного клацання елементу *CheckBox* на формі в режимі розробки, система **Visual Basic** створить процедуру обробки клацання *Check1_Click()*.

Подія *Click* (Клацання) відбувається як при встановленні, так і при скиданні прапорця. Щоб з'ясувати, встановлений прапорець чи скинутий, перевіряють значення властивості *Value* (табл. 2.5). Це роблять у процедурі *Check1_Click()* за такою схемою:

```
If Check1.Value = 1 Then
    <активізація параметрів, які пов'язані з прапорцем >
Else
    <відключення параметрів, які пов'язані з прапорцем >
End If
```

Приклад 1.68. Створення проекту, в якому користувач зможе вибирати різні значення властивості Font для елементу Label.

Властивості *Caption* об'єкта *Label1* надане значення *«Текст»*.

На формі розміщені три окремі елементи *CheckBox*, які мають імена *Check1, Check2, Check3*. Значення заголовків (*Caption*) змінені згідно з малюнком.

В залежності від стану прапорців застосовуються або відміняються параметри форматування шрифту для об'єкта *Label1*.

Властивості елемента *Label*, які відповідають за форматування тексту напису наведені в таблиці:

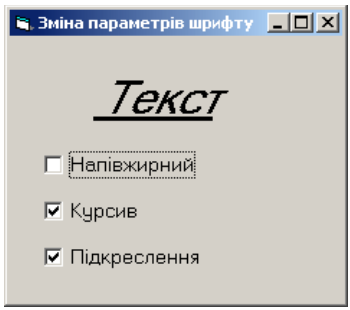
Властивість	Призначення	Можливі значення
FontBold	Напівжирний	True, False
FontItalic	Курсив	
FontUnderline	Підкреслення	

Перевірка стану прапорця *Check1* (Напівжирний) виглядає так:

```
If Check1.Value = 1 Then
    Label1.FontBold = True
Else: Label1.FontBold = False
End If
```

Перевірка стану прапорця *Check2* (курсив):

```
Private Sub Check2_Click()
    If Check1.Value = 1 Then
       Label1.FontItalic = True
    Else: Label1.FontItalic = False
    End If
End Sub
```



Питання для самоконтролю

1. *Дайте визначення алгоритмічної конструкції розгалуження.*
2. *Як записується і виконується умовний оператор у неповній формі? В якому випадку в неповному умовному операторі використовується ключове слово End If?*
3. *Як записується і виконується умовний оператор у повній формі?*
4. *Що слугує умовою в логічному операторі?*
5. *Запишіть у вигляді простої умови:*
   а) х – від'ємне число;
   б) число х не більше числа у;
   в) х – ділиться на 5.

6. *Запишіть у вигляді складеної умови:*
   а) 2 < x < 10;
   б) x не належить проміжку (2,10);
   в) x, y, z – додатні числа;
   г) хоча б одне з чисел x, y, z – додатне;
   д) жодне з чисел x, y, z – не додатне;
   є) x – парне число, більше від 10.
7. *Запишіть умовні оператори, які позначають такі дії:*
   а) перевірити, чи є число A парним;
   б) дано числа a і b. Від більшого числа відняти менше;
   в) перевірити, чи є серед чисел a,b,c рівні.
   г) упорядкувати числа a і b за неспаданням (якщо a>b, потрібно поміняти місцями значення a і b).
8. *Запишіть у вигляді умовного оператора обчислення функції Y = |X|.*
9. *Яких значень набудуть змінні A і B після виконання умовних операторів, наведених у таблиці, якщо початкові значення A=–3, B=5?*

A = –3, B = 5		A	B

1	*If A>B Then*
	A=0
	Else : B=0
	End If		

2	*If A<0 Then A=–A*		
3	*If A<>B Then A=B*		
4	*If A Mod 3 =0 Then* *A=A\3*		

A = –3, B = 5		A	B

5	*If A>B Then*
	A=A-B
	Else : B=B-A
	End If		

6	*If A<B Then*
	*A=2*A*
	*Else : B=B*A*
	End If		

10. *З якою метою в програмі може бути використаний елемент управління Прапорець? Яка подія є основною для нього?*
11. *Яка властивість дозволяє визначити, встановлений прапорець чи скинутий? Як організувати перевірку значення цієї властивості при зміні стану прапорця?*

1.10. Алгоритмічна структура Повторення. Цикл з параметром

Повторення (цикл)

Одним з найважливіших засобів програмування є можливість багаторазового повторення деякого набору команд. Команди, що повторюються, разом зі службовими словами, що забезпечують керування цим процесом, називаються *циклом*.

Цикл *– це алгоритмічна структура, за допомогою якої реалізується багаторазове повторення операторів.*

При цьому одна й та ж послідовність дій виконується доти, поки залишається істинною деяка умова. Серія команд, що

повторюється при кожному проході (ітерації) циклу, називається ***тілом циклу***.

Є два типи повторення: з передумовою та з післяумовою.

У випадку *повторення з передумовою* спочатку перевіряється умова, і якщо вона істинна, то тіло циклу виконується черговий раз, якщо ж ні – то виконання цих дій припиняється.

Якщо умова у вказівці повторення виявиться хибною при першій перевірці, то тіло циклу не виконається жодного разу.

Якщо ж при повторенні циклу умова незмінно залишається істинною, то виконання тіла циклу може повторюватися нескінченно (кажуть, що програма «зациклена).

Приклад 1.69. Алгоритм підрахунку суми N перших парних чисел (див. блок-схему). Позначимо через S шукану суму, через i – номер числа в послідовності парних чисел, a – поточне парне число. До початку циклу $S=0$ (ще нічого не підсумовували), a=0, i=0.

Щоб одержати наступне парне число, слід a збільшити на 2:

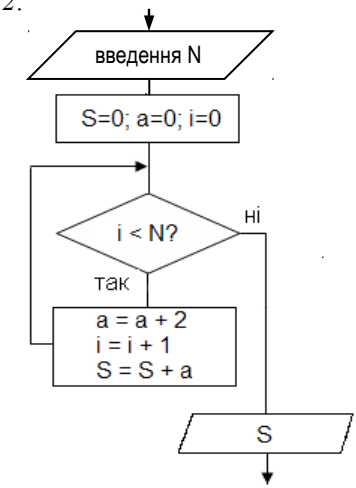
$$a = a + 2.$$

При переході до наступного доданка його номер збільшується на 1: $i=i+1$.

Наступний доданок додається до загальної суми: $S=S+a$.

Виконання циклу продовжується доти, поки $i < N$.

Зауважимо, що тіло циклу не виконається жодного разу, якщо на початку буде введене значення $N=0$. При першій перевірці умова $i<N$ виявиться хибною, тому відразу буде виведено нульове значення суми (змінної S).



Цикл із параметром

Цикл із параметром (або з лічильником) використовують тоді, коли треба виконати деякі дії певну кількість разів. Цикл із параметром реалізує оператор *For…Next*.

Синтаксис циклу *For…Next* такий:

For Змінна = ПочЗнач To КінЗнач [***Step Крок***]

<*оператори тіла циклу*>

Next **[*Змінна*]**, де:

- *For* – ключове слово, що позначає початок циклу;
- *Змінна* – ім'я змінної − параметра циклу;
- *ПочЗнач* – вираз − початкове значення параметра;
- *КінЗнач* – вираз − кінцеве значення параметра;
- *Step* – ключове слово, яке використовується, щоб вказати крок циклу (необов'язковий параметр);
- *Крок* – число, що задає крок циклу, тобто значення, на яке збільшується (чи зменшується) значення параметра після кожної ітерації;
- <*оператори тіла циклу*> – оператори, які повторюються в циклі. Як ви вже знаєте, кожне виконання тіла циклу називають ***ітерацією***;
- *Next [Змінна]* – ключове слово, що позначає кінець циклу *For...Next*. Вказувати після нього змінну-параметр не обов'язково, однак у деяких випадках це покращує читабельність коду, особливо за наявності вкладених циклів.

Виконується цикл за такою схемою:

1) параметр *Лічильник* одержує значення виразу *ПочЗнач*;
2) робиться перевірка, чи не перевищує значення змінної-параметра значення *КінЗнач*;
3) якщо *Лічильник* не перевищує *КінЗнач*, то виконуються оператори тіла циклу; якщо *Лічильник* перевищує *КінЗнач*, то цикл припиняється і керування переходить до рядка коду, який йде безпосередньо за ключовим словом *Next*;
4) якщо використовується ключове слово *Step*, параметр збільшується на величину *Крок*; якщо ж слово *Step* відсутнє, параметр *Лічильник* збільшується на одиницю і відбувається повернення до кроку 2.

Значення змінної-параметра змінюється на вказану величину або збільшується на одиницю щоразу після виконання тіла циклу.

Приклад 1.70. Програмний код, який реалізує алгоритм, представлений блок-схемою з прикладу 1.69:

```
N = InputBox("N?") ' Задання початкових значень змінних
S = 0 :  a = 0
For i = 1 To N    ' заголовок циклу: і приймає значення від 1 до N
      a = a + 2   ' оператори тіла циклу
      S = S + a
 Next i           ' кінець циклу
Print "S="; S     ' після завершення циклу виводиться значення S
```

Приклад 1.71. Параметр *A* збільшується на 2 на кожному кроці:

```
For A = 0 To 10 Step 2
      Print A
Next A
```

Приклад 1.72. Якщо необхідно змінювати значення параметра від більшого значення до меншого, то крок циклу роблять від'ємним:

```
For A = 10 To 0 Step –1
      S = S & Str(A)
Next A
```

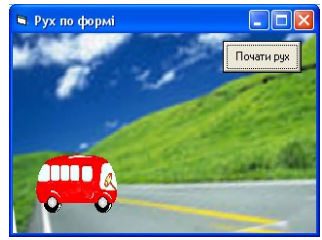
Приклад 1.73. Якщо кінцеве значення параметра циклу менше ніж початкове і крок циклу не зазначений, то тіло циклу взагалі не буде виконуватися:

```
For A = 10 To 0 ' Цей цикл не виконається жодного разу
      S = S & Str(A)
Next A
```

Приклад 1.74. Якщо значення змінної-параметра циклу змінюється в тілі циклу, є ризик ніколи не досягти кінцевого значення і програма «зациклиться».

```
For A = 1 To 10 ' Цей цикл буде працювати нескінченно
      A = A – 1
Next A
```

Приклад 1.75. В якості початкового та кінцевого значень параметра циклу можуть використовуватися вирази відповідного типу. Рух об'єкта по формі можна реалізувати за допомогою оператора циклу *For*:



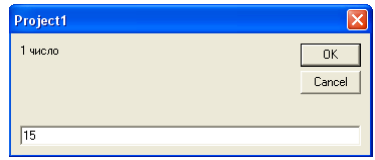
```
Dim i As Integer
Image1.Left = 0
For i = 1 To Form1.Width - 300
    Image1.Left = Image1.Left + 1
Next i
```

Приклад 1.76. Обчислення середнього арифметичного 5 дійсних чисел, що вводяться з клавіатури:

```
Dim i As Integer, A As Single, Sr As Single
Sr = 0
For i = 1 To 5
    A = InputBox (Str(i) & "число")
    Sr = Sr + A
Next i
Sr = Sr / 5
MsgBox "Середнє арифметичне " & Str(Sr)
```

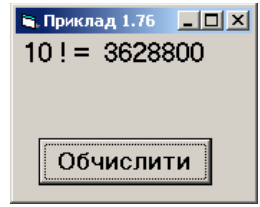
Приклад 1.77. З клавіатури вводяться 5 дійсних чисел, серед яких можуть бути і додатні, і від'ємні числа. Знайти середнє арифметичне додатних чисел.

```
Dim i As Integer, A As Single, Sr As Single, K As Integer
Sr = 0
K = 0 'лічильник додатних чисел
For i = 1 To 5
    A = InputBox (Str(i) & "число")
    If A >0 Then
        Sr = Sr + A
        K = K + 1
    End If
Next i
Sr = Sr / K
MsgBox "Середнє арифметичне додатних чисел " & Str(Sr)
```



Приклад 1.78. ***n!*** (читається «n-факторіал») обчислюється за формулою: n! = 1·2·3·…·n.

В програмному коді реалізоване обчислення $F = 10!$ із застосуванням прийому накопичення добутку:

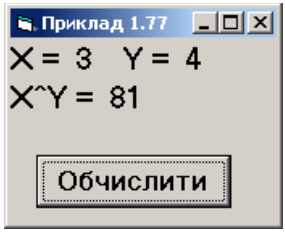



```
Dim i As Integer, N As Integer, F As Long
F = 1  :  N = 10
```

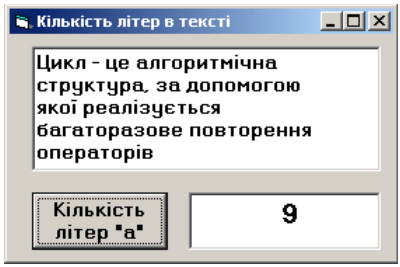
```
For i = 1 To N
    F = F * i
Next i
Print N; "! = "; F
```

Приклад 1.79. Не у всіх мовах програмування передбачено операцію піднесення до степеня $\mathbf{x}^{\mathbf{y}}$. Виконання цієї операції можна реалізувати за допомогою циклу *For*.

```
Dim X As Single, Y As Integer,
Dim P As Single, i As Integer
    P = 1
    X = InputBox ("X=?")
    Y = InputBox ("Y=?")
    For i = 1 To Y
       P = P * X
    Next i
    Print "X^Y = "; P
```



Приклад 1.80. Оператор циклу *For* зручно використовувати для аналізу значень рядкових величин (функції для роботи з даними типу *String* розглядалися в п.1.7). Підрахуємо кількість літер «а» в тексті *S*, який вводиться у текстове поле *Text1*. Для зручності, значення властивості *Multiline* об‘єкта *Text1* дорівнює *True*:




```
Dim S As String
Dim i As Integer
Dim K As Integer
Private Sub Command1_Click()
    S = Text1.Text
    For i = 1 To Len(S)
       If Mid(S, i, 1) = "a" Then K = K + 1
    Next i
    Text2.Text = Str(K)
End Sub
```

Приклад 1.81. Існує клас задач, розв’язування яких зводиться до перебору різних варіантів, в пошуках того, що задовольняє умові задачі. До цього класу належить задача пошуку дільників цілого числа.

Ціле число *K* є дільником числа *N*, якщо остача від ділення *N* на *K* дорівнює 0. Щоб знайти всі дільники, достатньо перебрати всі числа від 2 до *N* \ 2 і перевірити, чи є вони дільниками.

```
Dim N As Integer, K As Integer
N = InputBox ("N=?")
For K = 2 to N \ 2
    If N Mod K = 0 Then Print K;
Next K
```

Питання для самоконтролю

1. *Поясніть порядок виконання циклу з параметром. Скільки разів виконується тіло циклу?*
2. *З яким кроком змінюється значення параметра? Як змінюється значення параметра, якщо крок циклу не вказаний?*
3. *У якому випадку застосовують від'ємне значення кроку циклу?*
4. *Чому дорівнює S після виконання циклу:*

1	2	3	4

`S = 0`
`For A=5 To 7`
`  S = S + 1`
`Next A`
	`S = 0`
	`For A=5 To 7`
	`  S = S + A`
	`Next A`
		`S = 0`
		`For A=10 To 5 Step –1`
		`  S = S + 1`
		`Next A`
			`S = 0`
			`For A=10 To 5 Step–1`
			`  S = S + A`
			`Next A`

5. *Скільки слів «Hello» виведеться на форму?*

```
For i = -2 To 7
   Print "Hello"
Next i
```

6. *Що надрукують на формі наведені програми?*

1	2	3	4

`S = 1`
`For i=1 To 5`
`  If i>2 Then S = S * i`
`Next  i`
`Print S`
	`S = 0`
	`A = 3`
	`For  i =1 To 10`
	`  S = A + i`
	`Next A`
	`Print S`
		`S = 0`
		`A = 5`
		`For  i = 1 To 3`
		`  S = 2 * A`
		`Next A`
		`Print S`
			`S = 0`
			`A = 5`
			`For  i =1 To 3`
			`  S = S + 2 * A`
			`Next A`
			`Print S`

7. *Складіть таблиці зміни значень змінних для циклів:*

1	2	3

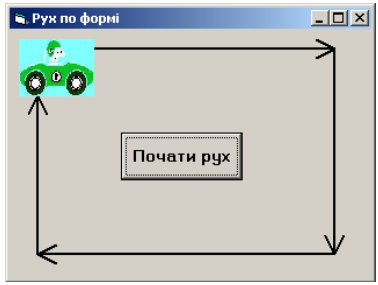
`S=0 :  P=1`
`For I=1 To 5`
`  P=P*I`
`  S=S+P`
`Next  I`
	`C =2`
	`For I=1 To 4`
	`  C=1/(1-C)`
	`Next  I`
		`S=0 :  P = 1`
		`For I=1 To 5`
		`  P =–P*2`
		`  S = S+P`
		`Next  I`

8. *Використовуючи прийом накопичення суми, запишіть програмний код для знаходження:*
   а) суми квадратів чисел від 1 до N (число N вводиться з клавіатури);
   б) суми парних чисел в діапазоні від 100 до 200;
   в) суми парних чисел від 1 до N (число N вводиться з клавіатури);

г) виразу R = 2N+4N+…+14N (число N вводиться з клавіатури);

д) виразу B=(A+3)(A+2.8)(A+2.6)…(A+1) (число A вводиться з клавіатури);

9. *Запишіть програмний код, який реалізує обчислення добутку цілих чисел N*M без використання операції множення.*
10. *Запишіть програмний код для підрахунку кількості літер B в тексті S, якщо значення змінної B типу String вводиться за допомогою функції InputBox.*
11. *Створіть проект, в якому реалізується рух об'єкта Image1 по формі за вказаною на малюнку траєкторією.*



12. *Розв'яжіть задачі за допомогою прийому перебору варіантів:*

а) Перевірити, чи є задане ціле число A кубом іншого цілого числа.

б) Задача Аль-Хорезмі (прибл. 780-850). Розкласти число 10 на 2 доданки, сума квадратів яких дорівнює 58.

1.11. Тематична робота «Базові поняття програмування»

Див. робочий зошит «Інформатика. Третій рік – єдиний курс. 11 клас.» /Бондаренко О.О., Ковшун М.І., Пилипчук О.П – Шепетівка: «Аспект», 2011.