І. О. ЗАВАДСЬКИЙ, Р. І. ЗАБОЛОТНИЙ

Основи візуального програмування

Київ
Видавнича група BHV
2007

УДК 375.5:004.432+004.432](075.3)
ББК 32.973-018.1я72
З-13

Рецензенти: Т. В. Ковалюк, вчений секретар науково-методичної комісії Міністерства освіти і науки України за напрямом професійного спрямування «Комп'ютерні науки», кандидат технічних наук, доцент НТУ «КПІ»,

Д. І. Кожем'яка, вчитель інформатики Фінансово-правового ліцею Фінансово-правового коледжу КНУ ім. Т. Шевченка

Гриф наданий Міністерством освіти і науки України, лист №

Завадський І. О., Заболотний Р. І.

З-13 Основи візуального програмування / І. О. Завадський, Р. І. Заболотний: [Навч. посіб.]. — К.: Вид. група BHV. — 2007. — 272 с.: іл.

ISBN 978-966-552-195-2

Посібник призначено для учнів загальноосвітніх навчальних закладів і вчителів інформатики. Запропоновано методику навчання програмування на прикладах розробки у візуальному середовищі програм із графічним інтерфейсом. Навчальною мовою програмування обрано Visual Basic .NET. Видання містить теоретичний матеріал, вправи, завдання для самостійного виконання і тести.

ББК 32.973-018.1я72

ISBN 978-966-552-195-2

Зміст

Передмова

Не секрет, що для більшості учнів програмування не є улюбленою темою курсу інформатики. По-перше, діти вважають, що програмувати важко. По-друге, вони не розуміють, як можна застосувати здобуті з цієї теми знання у реальному житті. Чим пояснити настільки несправедливе ставлення учнів до такого цікавого і захоплюючого предмета? На жаль, доводиться констатувати, що нині в українській шкільній освіті панує застаріла методика викладання основ програмування. Традиційний, «алгоритмічний», підхід до створення програм абсолютно не узгоджується з архітектурою та принципами функціонування всього сучасного програмного забезпечення. Наразі очевидною стає необхідність у навчанні програмування у візуальному середовищі, результатом роботи учня з яким має бути розробка програм з інтерфейсом, що відповідає вимогам сучасної графічної операційної системи.

Часто втіленню в життя такої методики навчання програмуванню стають на заваді міркування про те, що Windows-програми розробляти складно, що для цього потрібно спочатку на високому рівні оволодіти як структурним, так і об'єктно-орієнтованим програмуванням, вивчити різноманітні аспекти взаємодії прикладного програмного забезпечення з компонентами операційної системи тощо. Насправді такі погляди є не більш ніж стереотипом, який прагнули спростувати автори цього видання. У книжці пропонується вчитися програмувати в середовищі Visual Studio .NET, яке поєднує потужність функціональних можливостей із простотою візуальних засобів програмування. Завдяки цьому розробка простих Windows-програм стає не складнішим завданням, ніж опанування засобів графічного чи текстового процесора.

Слід підкреслити, що в посібнику належна увага приділяється всім основним концепціям класичного програмування, які мають вивчатися в середній школі, — змінним, операторам присвоєння, введення й виведення даних, алгоритмічним конструкціям розгалуження й повторення, масивам. Автори цієї книжки пропонують змінити лише форму й методику викладання, не звужуючи набору знань і умінь, які мають набути учні, та навіть розширюючи його деякими базовими засобами й методами Windows-програмування.

Однією з найпомітніших особливостей посібника є художній стиль тексту. Предмет викладається на тлі фантастичної історії про подорож підлітків у майбутнє, яка здатна зацікавити будь-кого, навіть тих, хто ставився до програмування упереджено. Проте напівігровий спосіб подання матеріалу не призводить до «урізання» його змістового наповнення, а, навпаки, дозволяє емоційно забарвити найважливіші факти, знайти життєві приклади для певних концепцій програмування тощо.

Зміст посібника повністю відповідає програмі курсу за вибором «Основи візуального програмування», що була рекомендована Міністерством освіти та науки України для використання в загальноосвітніх навчальних закладах. Програма розрахована на 33 навчальні години, а посібник містить 14 розділів, кожен з яких доцільно вивчати протягом двох годин. Таким чином, на опрацювання матеріалу цієї книжки потрібно 28 навчальних годин, а ще 5 залишаються для проведення контрольних робіт і захисту учнівських проектів.

Усі розділи посібника містять по дві вправи та по одному завданню. Перша вправа, як правило, підсумовує матеріал приблизно двох третин кожного розділу книжки. Якщо уроки проводяться в різні дні тижня, цю вправу можна давати учням як домашнє завдання після першого з двох уроків, відведених на опрацювання розділу. Друга вправа розміщується в кінці розділу і її слід виконувати протягом другої половини другого уроку — більшості учнів для цього буде достатньо 20-25 хвилин. Оскільки подані в посібнику завдання складніші за вправи, їх варто задавати учням додому наприкінці навчання за матеріалом розділу.

Для більшості вправ і завдань є заготовки — архіви проектів, що містять форми, але не містять коду, або ж містять код, який учні мають відредагувати тощо (відповідні файли можна знайти за адресою **www.osvita.info).** Використовуючи такі заготовки, учні, з одного боку, не виконуватимуть дій, навчання яким не передбачалося у пройденій частині курсу, та, з іншого боку, не повторюватимуть багаторазово одноманітні дії, які вони вивчили дуже добре. Саме завдяки заготовкам термін у 20-25 хвилин, рекомендований для виконання вправи на уроці, стає реальним.

Слід зазначити, що основний матеріал посібника, який розміщено між вправами, не є суто теоретичним. Щоб його засвоїти, необхідно виконати описані у книжці дії в середовищі програмування. Кожен із лекційних фрагментів уроку має тривати не більше п'яти хвилин, решта часу відводиться на практику: аналіз й покрокове виконання наведеного в посібнику коду, а також розв'язання найпростіших задач. Теорія викладена у книжці у формі розповіді, тому текст розділів може легко стати основою для лекційних фрагментів уроків.

У кінці кожного розділу наведено тест, що складається з восьми запитань щодо матеріалу розділу. Тест не дозволяє оцінити, чи набули учні певних навиків з програмування, а лише дає можливість перевірити, чи були вони уважними на уроці, чи зрозуміли основні поняття, чи запам'ятали синтаксичні конструкції мови програмування тощо. Тому автори не рекомендують за результатами тестування виставляти учневі оцінку за 12-бальною шкалою. Тест може використовуватися учнями для самоконтролю, або враховуватися вчителем під час формування інтегрованої оцінки за роботу учня над темою розділу. Наприклад, вдалим рішенням могло б стати формування інтегрованої оцінки на основі оцінювання результатів тестування за 4-бальною шкалою (по одному балу за кожні дві правильні відповіді), а також виставлення 4-бальних оцінок за виконання вправ і домашнього завдання.

Наведені вище рекомендації щодо методики використання посібника в навчальному процесі, безперечно, не є обов'язковими. Видання може стати основним підручником і під час

викладання тем «Алгоритмічне програмування» та «Основи візуального програмування» базового курсу інформатики. Модульна структура посібника дає змогу вчителю гнучко варіювати обсяг та зміст навчального матеріалу.

Автори щиро дякують корпорації Microsoft за матеріали, що лягли в основу цієї книжки, та за відчуття зручності й комфорту, що не полишало їх під час програмування у Visual Studio .NET.

Від видавництва

За повною інформацією, що стосується посібників серії «Інформатика в школі» та інших освітніх видань Видавничої групи BHV звертайтеся за адресою **http://www.osvita.info.**

Вступ

— Ну все, здається, ми готові, — прошепотів Михась, закривши блискавку на рюкзаку, яка не хотіла сходитися через гострий край ноутбука, і востаннє оглянув кімнату. Даринка перевірила свої речі (улюблена лялька, ліхтарик, мамина косметичка — все начебто на місці) та підійшла до вікна.

— А батьки не будуть нас шукати? — схвильовано запитала дівчинка. Михась на мить припинив натягувати рюкзак і задумався.

— Напишемо їм записку! — вирішив він. — Пиши ти, в тебе це краще вийде.

— Ну звичайно, дитя комп'ютерної ери не звикло писати, — не стрималася сестра, але все ж таки підійшла до столу, вирвала із зошита аркуш паперу, і за якусь мить на ньому з'явилися рівненькі літери.

— І що це таке? Цей текст не несе ніякого повідомлення! — прочитавши записку, пхикнув Михась.

— Напиши краще, — образившись, відповіла Даринка і попрямувала до дверей. Проте вона була впевнена, що Михась не дійде до того, аби писати будь-що звичайною ручкою. І справді, на хвилинку замисливши, брат рушив за нею.

Минулого тижня, коли з радісним криком «Еврика!» до квартири влетів їхній тато, Михась відразу зрозумів, що відбулася насправді визначна подія. Вже довгий час, разом з іншими провідними науковцями планети, їхній батько працював над величезним і дуже важливим для всього людства проектом — Трансконтинуальним Апаратом Томсера, або по-простому («поселянськи», як любив казати тато) над машиною часу.
Теорія, згідно якої час зв'язаний із простором, виникла давно — про це говорив ще великий винахідник Ейнштейн. Можливість впливати на час відразу вразила Михася, а тому він довго дошкуляв татові запитаннями, на які той змушений був відповідати довгими вечорами, вимикаючи телевізор, витягуючи старі книжки і малюючи схеми на папері. Звичайно, хлопчик усього не розумів, але найважливіше він усвідомлював: саме його батько працює над машиною часу! І ось нарешті експерименти завершилися. Батько того дня був удома раніше звичайного і чи не вперше за довгі роки пообідав разом із родиною. Саме за обідом він розповів про свої досягнення та проблеми. Так, проблеми були, і над їхнім вирішенням уже довго працював не один оптимізатор — реальні зразки ТАТів не могли переносити у часі на значну відстань об'єкти, що мали велику масу. Не можна було перевірити, що відчуває людина, подорожуючи у часі, бо вага дорослої людини була для цього завеликою.

— Тату, давай ми з Даринкою спробуємо! — відразу запропонував Михась.

— Ну, ні! Ніколи не дозволю своїм дітям ризикувати власним життям! — різко відповіла мама. Тато змушений був із нею погодитися.

Саме після тієї розмови Михась вирішив, що він стане першим мандрівником у часі, навіть якщо мама йому після цього заборонить гратися в мережі цілий тиждень. Саме з того часу всі свої знання (а він знав чимало) хлопчик спрямовував на втілення цієї мрії.

І ось день «Х» настав. Михась не міг повірити в те, що вони разом із Даринкою ідуть до татка в лабораторію, щоб здійснити першу подорож у часі, йдуть без усяких урочистостей, парадів

і промов (чесно кажучи, Михась ніколи не любив усієї тієї балаканини) започатковувати історію подорожей людства у часі. Даринку дивувало те, що вона так легко погодилася на авантюру брата, і ось вони разом прокрадаються до лабораторії, де зараз має бути тільки сторож, який завжди спить перед телевізором. Батько розповідав, що на сторожа постійно скаржилися вчені: «Стадо слонів, протупотівши повз нього, може дістатися найдорожчого обладнання людства та повернутися назад, а той навіть ока не відкриє!». І справді, коли діти нишком підійшли до дверей лабораторії, їх не зупинив ніхто, хоча, як Михась пам'ятав, уся територія дослідницького центру простежувалася за допомогою різних детекторів, і їх було чудово видно, як би тихенько вони не прокрадалися.

Нарешті здійснилась їхня мрія. Невеликий кулеподібний апарат, обвішаний різними дротами та датчиками, ззовні, здавалося, майже не змінився з того часу, коли вони відвідували востаннє тата на роботі. Але Михась знав, що головні зміни відбулися не у фізичній конструкції апарата, а у програмному забезпеченні. Як любив казати їхній учитель інформатики: «Комп'ютер — це тільки купа металу, яка не здатна ні до чого, поки людина її не запрограмує».

— Михасю, ти знаєш що тепер робити? — сполохано запитала Даринка. Хоча хлопчик дещо й розбирався в техніці, та не так добре, щоб бути абсолютно впевненим у своїх діях. Але для того щоб Даринка раптом не кинула все і не втекла додому (хто там тих дівчат знає), він упевнено відповів:

— Так. Я залізу всередину, а ти спочатку вистав час ось на цьому зеленому індикаторі, а тоді натисни цю червону кнопку. Так-так, не бійся, — заохотив Михась сестру. — На синьому дисплеї задаються параметри, за допомогою яких можна повернутися назад, — час і місце, куди необхідно перенестися, а також маса об'єкта, який

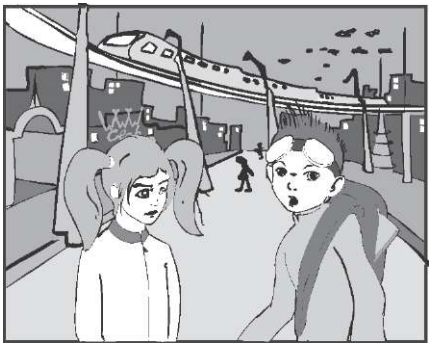
повертається в минуле. Та дивися, зроби все правильно, а то я не повернуся! — налякав він сестру і впевнено, по-геройськи, зайняв місце пасажира. Це була невеличка камера, що мала вигляд кулі та в яку він ледь-ледь помістився.

Через кілька хвилин усе закрутилося, і Михась подумав: «Ось це і є та сама мить, заради якої варто жити».

Коли все втихомирилося, хлопчик опинився посеред вулиці, яка мала звичайний вигляд. Проте щось тут було не так. На вулиці не було нічого дивного: та сама дорога, ті самі тротуари, ліхтарі, птахи, перехожі — все як і вдома. Але водночас із цим усе було трохи інакшим — незвичайне покриття дороги, чимось подібне до тканини, одежа людей, наче зроблена з металу. Коли Михась спробував підійти до якогось дядька зі своїми запитаннями, той його просто проігнорував і обійшов, як неживого. У Михася відразу промайнула думка — «Може, я невидимий?». Але він швидко її відкинув, бо той чоловік замислено озирнувся на нього. Тоді хлопчик спробував пройти кілька кварталів, здивовано розглядаючи все навколо.

Через певний час, переходячи чергову вулицю, він усвідомив, що не всі перехожі однакові — деякі з них були цілком нормальні, дуже схожі на його сучасників, а дехто мав вигляд робота. Згодом Михась призвичаївся до оточення і вирішив купити собі сувенір — хто не хотів би мати якусь річ із майбутнього?

Коли хлопчик усе навколо оглянув, то почав думати про повернення назад, тим паче що час, який йому було відведено на подорож у майбутнє, вже потроху спливав. Хоча він і ходив незнайомими вулицями, та був упевнений у тому, що не заблукає, бо добре пам'ятав дорогу назад. Але раптом серед похмурого і незвичного натовпу промайнула знайома постать. Михась обімлів — це була Даринка.



— Що ти наробила! Як ти потрапила сюди? — крикнув він, схопивши її за рукав.

— Не кричи на мене! Я опинилася тут так само, як і ти! — налякано відповіла Даринка.

— А ти змінила масу об'єкта, щоб я зміг повернутися назад?

— Ні, так і лишила, маса повернення дорівнює масі перенесення, плюс-мінус кілограм, — відповіла Даринка, від чого Михасю стало моторошно.

— Ти скасувала програму мого повернення, завантаживши нову, за допомогою якої ти потрапила сюди! То ж виходить... так тепер... я ж не повернуся? — ледве не заплакав Михась. Лише тепер Даринка зрозуміла, що вона накоїла. Вони відійшли на узбіччя та сіли на бордюр, щоб обміркувати, як їм бути.

— Добре, повертайся, — вирішив Михась. — Розкажи батькам про те, що трапилося, і нехай вони допоможуть мене врятувати.

— Але я не можу тебе тут залишити! Що ти будеш робити?

— Я почекаю, а ти повертайся та виправляй свою помилку! — знову крикнув Михась.

— Може, існує інший спосіб, щоб не залучати батьків? — запитала сама в себе Даринка, але питання повисло в повітрі.

Раптом вони почули — вперше у цьому часі — людську мову:

— А ви чого без роботи? Що, програма злетіла? Ну чому в мене знову нічого не вийшло?

Спочатку Михась із Даринкою злякалися і не змогли вимовити жодного слова, але потім Даринка несміливо промовила:

— Ми... не звідси... а ти хто?

Незнайомець (він був трохи старшого віку) здивовано на них поглянув і запитав:

— Чиї ви будете? Яка модель? Що робите тут? — а після невеликої паузи радісно вигукнув: — Штучний інтелект! Я знайшов штучний інтелект! — на що Михась, який про це дещо читав, відповів:

— Ні, ми не штучний інтелект, ми люди... То як тебе звати? Та й хто ти взагалі?

— Я — ВВ/550, а ви хто? Тут немає людей, крім мене і моєї родини. То як ви поясните своє перебування тут?

— Ну а хто тоді оці всі люди? — здивувалася Даринка. — Це все твоя родина?

— Ні, це роботи. Я спочатку думав, що ви теж... А ви точно не роботи? — трохи здивовано запитав ВВ.

— Здається, ні, але у нас виникла проблема — ми з іншого часу, з минулого, і не можемо повернутися назад. — Михась спробував пояснити ситуацію, в якій вони з Даринкою опинились, а потім додав: — Ти можеш нам допомогти?

— То ви хочете дізнатися про майбутнє? Аж ніяк! Нам заборонено взагалі спілкуватися з іншочасовиками! Я зараз викличу робота-поліцейського! — протарахкотів ВВ, зробивши крок назад.

— Ні, нам потрібно лише повернутися назад, додому! — швидко вимовила Даринка.

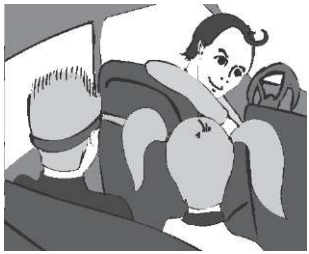
— А... Ну тоді пішли за мною, я вас проведу, — промовив ВВ і розвернувся, щоб іти, але відразу зупинився. — Хоча, може, перед тим як ви повернетеся додому, ми трохи розважимось? А то спілкуватися із роботами мені вже набридло.

— Добре, — відповів Михась. — Але розкажи хоча б щось про себе, поки ми дістанемось потрібного місця.

— Гаразд, сідайте в мою машину! — відповів ВВ. Коли він доторкнувся до свого годинника, який також мав дивну форму, біля них зупинилося чудернацьке авто.

День 1
Знайомство

— То ким ти мрієш стати? — запитав Михась у ВВ/550.



— Звичайно ж, програмістом! — відповів той. — У наш час, на відміну від вашого, немає такого різноманіття професій — всю просту роботу виконують роботи і різні механізми, а люди в основному займаються творчістю. Але серед людей є такі, які не поступаються роботам. Вони працюють будівельниками чи садівниками — для більшості з них це хобі. А я хочу бути саме програмістом і виконувати найважливішу роботу в цьому світі! — останнє речення він вимовив не без захоплення.

— А хіба програміст — це настільки важлива професія? Я от думаю, що вчителькою бути набагато важливіше, і коли виросту, то обов'язково стану нею, — зауважила Даринка.

— Бути вчителькою! Думаєш, ти зрівняєшся з найновішим роботом «Вчитель-1124», який ніколи нічого не забуває, завжди об'єктивний і знайде підхід до всіх дітей? А його, між іншим, також запрограмували! — відповів ВВ.

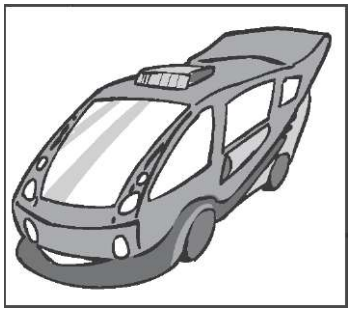
Де використовуються програми

— А де ще у ваш час використовуються програми? — запитав Михась.

— Ви, напевно, здивуєтесь, якщо я скажу, що більшість речей, які нас оточують містять у собі програми. А ще дивніше те, що багато речей розробляється та перевіряється за допомогою

комп'ютерних програм, які тепер застосовуються не лише в роботі калькуляторів, автоматів або комп'ютерів, а й у роботі багатьох приладів, іграшок і навіть машин.

Візьмемо, наприклад, цю машину, яка складається з речей, що, напевно, були розроблені за допомогою комп'ютера. Аеродинамічна форма була, ймовірно, змодельована та перевірена на комп'ютері в імітованій аеродинамічній трубі. Сидіння, кухня та ліжка теж були розроблені на комп'ютері, а потім розміщені у трьохвимірній моделі машини.



— А тут і кухня є? — здивувалася Даринка.

— Ну, звісно, а якщо пасажирам захочеться їсти? — здивувався ВВ. — Робот, який теж, до речі, запрограмований, приготує їжу за їхнім замовленням не відриваючись від дороги. Хіба це не зручно?

У цій машині є багато інших речей, зокрема, глобальна система визначення місцезнаходження, мікрохвильова піч, стереосистема. Усі вони програмовані, і для них передбачене віддалене керування. Програми визначають, що робить прилад, коли ви натискаєте певну кнопку. Вони «запаяні» в мікросхемах на платах, але спочатку всі ці програми потрібно було комусь написати. Програми використовуються для вирішення найрізноманітніших проблем та для створення різних речей. Тепер майже кожен користується персональним комп'ютером на роботі та вдома для написання листів, складання таблиць і баз даних, а також для ведення чекової книжки. Практично всі установи користуються програмами, що дають змогу відстежувати рух клієнтів, коштів і товарів. Величезна кількість програм використовується для збереження, отримання та фільтрування інформації в різний спосіб. Чимало програм написано для дизайну, малювання, гри тощо. Крім того, існує багато різних типів спеціалізованих програм, що застосовуються для досліджень у галузях штучного інтелекту, генетики, кліматичних змін. Яку б сферу людської діяльності ви не

назвали, напевно, що знайдуться програми, які використовуються в ній.

Деякі люди пишуть програми заради розваги! Я маю робота, який складає для мене домашні завдання — його запрограмував тато. А я цього робота перепрограмував — і тепер він не тільки складає, а й розв'язує задачі! Мій старший брат створив робота Lego та запрограмував його, щоб той складав кубик Рубіка. Я хочу програмувати тривимірні інтерактивні відеоігри або писати для страхової компанії програми, що аналізуватимуть статистичні дані.

— Здається, у моєму кишеньковому комп'ютері є вправа, що стосується програмування, — ВВ витягнув комп'ютер, який був хоч і меншим від того, що містився в сумці Михася, але значно кращим — у цьому Михась не сумнівався — та протягнув його Даринці.

— Ой, який гарний, — все, що змогла вимовити Даринка. Їй нічого не залишалось, як на кілька хвилин зосередитися на поставленому завданні.

Вправа 1.1. Сфери застосування комп'ютерів

Запишіть, у яких сферах використовуються комп'ютери в повсякденному житті. Увага! Не користуйтеся прикладами, які наведено в цій книжці.

Робота програміста

Закінчивши вправу, Даринка запитала:

— А чим займаються програмісти? Багато людей уявляють програмістів такими собі диваками, що сидять за комп'ютерами вдень і вночі, вистукуючи код на клавіатурі, поїдаючи холодну піцу та п'ючи збагачену кофеїном суміш. Невже ти теж хочеш так жити? — запитала вона у ВВ.

— Хорошу програму за ніч не створиш. Для того щоб створити комп'ютерну програму, яка буде робити саме те, чого ви від неї очікуєте, її необхідно старанно планувати та розробляти.

Це потрібно робити так, щоб програму було легко змінювати, щоб у ній не було помилок і вона була зручною для використання. Над великими проектами працюють команди із сотень програмістів. Усі вони мають пройти навчання, а потім потрібно організувати роботу, яку вони виконуватимуть, та керувати нею. Хоча написання коду входить в обов'язки програміста, та він виконує й інші дії:

- вирішує, що програма буде робити;
- розробляє інтерфейс користувача;
- обирає мову програмування;
- розробляє архітектуру програми та визначає, як її частини будуть взаємодіяти;
- визначає стилі написання коду та дизайну;
- вирішує, хто буде писати код окремих частин програми;
- упорядковує розклад розробки та стежить за його дотриманням;
- навчає інших програмістів;
- пише код!!!
- документує код;
- розробляє базу даних для збереження відомостей, які потрібні для програми або створюються нею;
- керує розробкою графічних зображень;
- керує введенням інформації в базу даних;
- налагоджує код, виправляючи в ньому помилки;
- перетворює код на програму, яку можна встановити на комп'ютері користувача або на сервері;
- навчає користувачів працювати з програмою;
- ліквідує проблеми, які виникають у користувачів під час роботи з програмою;
- пише документацію та навчальні матеріали для програми;
- модернізує програму.

Отже, написання коду — це лише одна з багатьох функцій програмістів. У моєму кишеньковому комп'ютері є ще одна вправа для програміста.

Вправа 1.2. Визначення заробітної платні IT-фахівців

Користуючись Інтернетом та іншими ресурсами, визначте початкову зарплатню спеціалістів, що працюють у сфері інформаційних технологій у вашій місцевості. А потім порівняйте заробітну платню IT-спеціалістів різного фаху, наприклад, програмістів, системних аналітиків, розробників веб-застосунків тощо.

Цього разу комп'ютер нарешті потрапив до рук Михася, який одразу ж почав його порівнювати зі своїм, абсолютно забувши про завдання. Та добре розглянути комп'ютер йому не вдалося — завдання вимагало розв'язання, і він швидко почав пригадувати все, що знав із цього питання.

Поняття програми

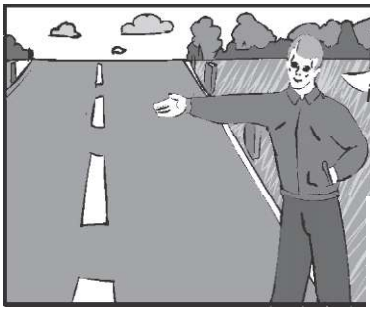
— То що ж таке програма? — запитав ВВ, перебиваючи черговий потік запитань від Даринки і Михася. — Комп'ютерна програма складається з рядків *коду,* що написані англізованою *мовою програмування.* Обираючи мову програмування, враховуйте її призначення та вимоги вашої програми. Після написання програма *компілюється* (перекладається) мовою, яку «розуміє» комп'ютер (тобто може виконувати вказівки, записані цією мовою).

Ви можете створювати різні типи програм, користуючись однією мовою програмування, наприклад, для гри у хрестики-нулики, для показу слайдів та для відображення орбіти польоту ракети навколо Землі. Відмінності між цими програмами полягатимуть у коді, який ви напишете. Він визначатиме функціональне призначення програм, тобто те, що вони будуть виконувати. Наприклад, у коді визначається, що відбуватиметься, коли ви клацнете кнопку або виберете елемент зі списку. Ваш код також описує «інтелект» програми: як програма буде «приймати рішення», скільки разів виконуватиме певні дії та які розрахунки здійснюватиме. Ви можете написати код для розв'язування математичних задач, введення тексту, реагування на дії користувачів, збирання даних або відображення повідомлень.

Написання коду може бути складним, але це цікавий процес, що компенсує витрачені зусилля. Коли ви вивчаєте мову програмування, то опановуєте новий матеріал, за допомогою якого можна виражати творчі ідеї, досліджувати дані та повідомлення, вирішувати проблеми й розважатися. У цьому програмування подібне до будь-якої іншої творчої роботи, наприклад, до ліплення з глини або до написання музики.

Історія мов програмування

— А які бувають мови програмування? — запитала Даринка. — Мій дідусь знав чотири мови: японську, іспанську, суахілі та українську. П'яту він згадував тоді, коли страшенно розлючувався! — всі разом засміялися.



Будь-якою мовою цей жест означає: «Зупиніться, будь ласка»

— Комп'ютерні мови багато в чому подібні до розмовних. Вони можуть по-різному звучати або мати різний вигляд, але призначення в них одне й те саме: описувати інструкції для комп'ютера щодо того, яким чином вирішувати проблему, — поважно відповів ВВ.

— Ви знаєте, що постійно створюються нові мови програмування? З часу виникнення першої мови було розроблено тисячі їх різновидів. Як і деякі розмовні мови, певні мови програмування більше не використовуються. Форми інших мов програмування змінюються, але їх використовують протягом десятків років.

— А в нашому часі також були різні мови програмування! — згадав Михась. — Хіба ще й досі не дійшли висновку, яка мова програмування найкраща? — запитав він у ВВ.

— Ні, кожна мова орієнтована на певну область застосування, тому неможливе створення єдиної оптимальної мови — всі спроби такої оптимізації завершувалися породженням черго-

вого громіздкого і незручного проекту, який був орієнтований на все і, водночас, на ніщо. Для створення нової мови програмування беруться найкращі риси однієї або кількох існуючих мов, а потім об'єднуються та доповнюються. У сучасних мовах програмування знаходить своє відображення розвиток комп'ютерного устаткування і програмного забезпечення.
Програмувати мовами, які були першими, доводилося таким чином, щоб комп'ютер міг «розуміти» код — використовуючи лише одиниці та нулі. До чого ж це було нудно! З часом були розроблені програми-компілятори. Ці програми перетворювали код, що був написаний мовою, схожою на англійську, на одиниці та нулі, які «розуміє» комп'ютер. Майже всі сучасні мови програмування передбачають компіляцію. Тобто програми ними пишуться за допомогою англізованої мови, а вже потім вони компілюються у код, що придатний для прочитання машиною.

З роками з'явилося багато різних мов програмування, що призначені для вирішення конкретних проблем. Наприклад, мова Фортран — мова перекладу формул (англ. FORTRAN походить від FORmula TRANslating — перекладання формул) — була розроблена для вирішення числових задач та виконання обчислювань. Мова Кобол (COBOL — Common Business-Oriented Language) — мова програмування для ділової сфери, яка була розроблена переважно для використання в бізнесі, де потрібно обробляти велику кількість нечислових даних. Мова Ліспі (LISP — List Processing) — мова обробки списків, що була розроблена для вивчення штучного інтелекту.
Відтоді розвивалися нові мови, кожна з яких мала одну або кілька переваг порівняно з попередніми. Назвемо приклади таких мов:

- C — швидко розроблюваний, ефективний код;
- Java — незалежність від системної платформи;
- BASIC — простота у використанні;
- Visual Basic — побудова інтерфейсу користувача за допомогою форм;
- Perl — маніпулювання текстом.

Коли ви розпочинаєте новий програмний проект, потрібно визначитися щодо мови програмування. Вибираючи мову, керуйтеся такими критеріями.

- Чи забезпечує мова необхідні вам функціональні можливості?
- Чи легко нею програмувати?
- Чи достатньо знаєте ви цю мову?
- Скільки коштують засоби розробки програм цією мовою?

— А якою мовою програмуєш ти? — запитав Михась.

— Моя улюблена мова — Visual Basic .NET, — відповів ВВ. — Вона має потужні можливості, та нею легко користуватися. Я вас можу з нею ознайомити — у нашому світі неможливо жити активно, не знаючи жодної мови програмування. Та й у вашому часі знати основи мов програмування корисно кожному, тому протягом вашого перебування тут я спробую розповісти вам про цю мову, — вирішив він, на що Михась та Даринка ствердно захитали головами.

Елементи мови програмування

Якщо ви мандруєте світом, вам може знадобитися послуга когось, хто розмовляє англійською. У кожній мові є свій спосіб запитати: «Ви говорите англійською?», наприклад, англійське «Do you speak English?», німецьке «Sprechen Sie Englisch?», іспанське «Ї Habla ingles?», португальське «Vo^ fala ingles?». У цих прикладах різними є не лише слова, а й їхнє розташування в реченні: підмети, присудки та доповнення розміщені в різному порядку. Порядок слів визначається *синтаксисом* мови. Тобто, синтаксис — це встановлені правила, згідно з якими у мові будується речення.

Як і розмовні мови, кожна мова програмування має свій синтаксис. Синтаксис мови програмування — це словник, граматика, правила використання слів та утворення складніших структур. Синтаксис визначає правила написання рядків коду та їх поєднання у функціонуючу програму.

Наприклад, в усіх сучасних мовах програмування можуть виконуватись оператори `If...Then...Else`. Завдяки ним забезпе-

чується вибір одного з двох варіантів дій, залежно від певних умов. Зараз я покажу вам на своєму портативному комп'ютері приклад коду, записаного мовами Visual Basic .NET та C#, де використовується синтаксична конструкція `If...Then...Else`. Поки що вам не потрібно перейматися змістом цієї конструкції або розуміти код, просто зверніть увагу на подібне та відмінне в синтаксисі мов програмування.
Ось фрагмент коду мовою Visual Basic .NET:

```
If x>5 Then
   MessageBox.Show("МеНі вже виповнилося 5!")
Else
   MessageBox.Show("Мені ще не виповнилося 5!")
End If
```

А ось код мовою C#, який виконує ті самі дії:

```
If (x>5)
   MessageBox.Show("Мені вже виповнилося 5!");
Else
   MessageBox.Show("Мені ще не виповнилося 5!");
```

Вивчення мови програмування полягає у вивченні її словника, синтаксису та способу використання базових конструкцій мови. Ви маєте запам'ятати основні зарезервовані слова. Певні слова вважаються «зарезервованими», або «ключовими», оскільки вони використовуються лише як команди мови. Кожне з цих слів має спеціальне призначення. Наприклад, слова `If` та `Else` є зарезервованими в більшості мов програмування. Вони використовуються для створення в коді інструкцій, що забезпечують прийняття рішень.

Коли ви вивчаєте мову програмування, окрім синтаксису, слід осягнути її функціональні можливості. Пам'ятайте: не всі мови програмування є багатофункціональними. Але всі сучасні мови програмування мають спільні риси та, як правило, схоже функціональне призначення.

Visual Basic .NET

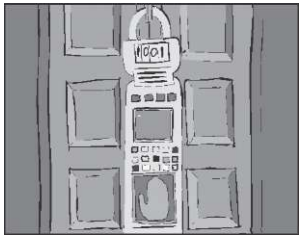
Перейдемо до конкретної мови. Visual Basic .NET є достатньо потужною мовою, але вивчити її легко. Навчаючись програмувати цією мовою, ви дізнаєтеся про зарезервовані слова, структуру

програм та базові об'єкти, з яких вона складається. Спочатку ви маєте дізнатися про основні синтаксичні особливості мови. Я зробив короткий список головних особливостей синтаксису Visual Basic .NET:

- у рядках коду немає індикатора кінця рядка, наприклад, крапки з комою (;);
- рядки коментарів починаються з апострофа (');
- для розміщення блоків коду не використовують фігурні дужки, ключові слова `BEGIN` та `END` та інші конструкції;
- мова Visual Basic .NET не чутлива до регістру символів, для неї MyCase — те саме, що й myCase або MYCASE.

— ВВ, а які основні синтаксичні відмінності є в C#? — запитав Михась.

— У C# кожний рядок коду закінчується крапкою з комою. Блоки коду містяться у фігурних дужках — { та }. Рядки коментарів починаються з двох скісних рисок (//). Мова C# чутлива до регістру. В ній MyCase відрізняється від myCase або MYCASE.



— От ми і приїхали додому! Ну як вам мій будинок? Але є одна проблема — це тато! Я, як і всі звичайні діти, не маю жодного бажання вчитися, тому він поставив скрізь двері з кодовими замками, які не відчиняються, доки я не відповім на кілька запитань! Щось на кшталт Сфінкса, як він казав. Отже, якщо хочете потрапити всередину, маєте скласти тест. Я його відрегулюю, щоб запитання відповідали тій інформації, яку я вам надав. Тож працюйте, а я поки виконаю те, що тато запрограмував для мене. Чекаю на вас всередині!

Кодовий замок

1. Чим різняться мови програмування?
   а) синтаксисом;
   б) перекладом;
   в) написанням.

2. У чому полягає компіляція програми?

а) у збиранні даних у внутрішньому стеку;

б) у зв'язуванні об'єктів форми зі сторінкою коду;

в) у перекладі тексту програми на машинний код.

3. Чому існують тисячі різних мов програмування?

а) тому що спеціалістам із комп'ютерних технологій потрібно чимось займатися;

б) кожна з них створювалася для вирішення певних проблем;

в) для кожного типу комп'ютера потрібна унікальна мова.

4. З чого зазвичай складається мова програмування?

а) з таблиць перекладу;

б) з рядків коду, що написані англізованою мовою;

в) з математичних символів, які створюють комп'ютери.

5. Коментарі у мові Visual Basic .NET:

а) не існують;

б) перекладаються компілятором на машинний код;

в) записуються в рядках, що починаються з символу апострофа.

6. Сучасні мови програмування:

а) мають спільні риси;

б) не мають спільних рис;

в) непорівнювані.

7. Регістр літер у мові програмування Visual Basic .Net:

а) є суттєвим;

б) є несуттєвим;

в) може бути суттєвим залежно від того, в якому місці розміщено код.

8. Яку з поставлених задач комп'ютер не зможе розв'язати?

а) знайти розв'язки рівняння;

б) знайти площу трикутника;

в) придумати нову задачу.

День 2

Інтегроване середовище розробки програм

Наступного ранку Михась і Даринка вже призвичаїлися до нового середовища і познайомилися з ріднею ВВ/550 (наскільки вони зрозуміли, ВВ/550 — це повне ім'я, бо його батько та мати офіційно представились як ВВ/548 і ВВ/549). Після дивного сніданку (погодьтеся, не кожен день тебе обслуговує робот, який готує за кілька хвилин будь-яку страву) ВВ привів Михася та Даринку до себе в кімнату, і вони продовжили розмову про навколишній світ, звичайно, зосередившись на програмуванні, яке їх найбільше цікавило.

Перш за все, Михась показав товаришу свій новенький ноутбук, який на ВВ не справив ніякого враження, оскільки його власний був у сотні разів потужніший. Але і «старенький» комп'ютер Михася підтримував програмне забезпечення, яке було необхідне для того, щоб продовжувати навчання. Поки інсталювалися потрібні програми, Даринка встигла розпитати у ВВ про моду та про те, як одягаються жителі майбутнього. Виявляється, що в епоху роботів одежа також буде роботоподібною: найбільші модниці та модники намагатимуться наслідувати «металевих людей». Потім ВВ розповів дітям про те, що батько подарував йому робота, який був підключений до світової мережі.

— Цей робот, RT1124.34, інколи дає настільки корисні поради, немов він має розум! — із захопленням продовжував свою розповідь ВВ. — Я спочатку подумав, що тато мені подарував ще не випробуваний варіант штучного інтелекту, і був страшенно радий! Але наступного дня, на моє прохання, батько

пояснив принцип дії цього робота, і я відразу засмутився! Усе виявилося дуже простим!

Після настроювання певних програм ВВ продовжив свою розповідь про мови програмування, ілюструючи її на своєму ноутбуку.

— Отже, сьогодні я розповім вам усе, що необхідно знати для складання тесту на керування IDE. IDE — це *інтегроване середовище розробки програм* (англ. — Integrated Development Environment). Ми познайомимося з IDE Visual Studio корпорації Microsoft. Ви будете користуватися ним, коли розроблятимете власні програми на мові Visual Basic .NET. Крім того, IDE можна застосовувати для написання програм іншими мовами програмування.

Знайомство з Visual Studio

— Ви знаєте, що можна писати комп'ютерний код, використовуючи звичайний текстовий редактор, наприклад Блокнот або Microsoft Word? — запитав ВВ у Михася та Даринки.

— Так, — відповів Михась. — Мені тато розповідав, що раніше саме так і працювали програмісти — у будь-якому текстовому редакторі набирали код, а потім компілювали його.

— Але це був складний і дуже довгий процес, — зауважив ВВ. — Багато часу витрачалося тільки на перевірку помилок, що автоматично ставали видимими у зручному для них середовищі програмування! Отже, ми будемо користуватися спеціальним засобом, за допомогою якого можна значно швидше розробляти програми, — Visual Studio корпорації Microsoft. У середовищі цього інструмента розробки можна створювати програми різними мовами програмування з родини .NET. Наприклад, коли ви додаєте до форми кнопку, Visual Studio автоматично вставляє код для обробки клацання цієї кнопки. Звичайно, Visual Studio не може написати весь код за вас, тому що не знає, які дії має виконувати ваша програма. Але використання інтегрованого середовища розробки може скоротити обсяг коду, що вам доведеться писати, і час, який ви на це витратите.

Visual Studio робить життя простішим ще й тому, що в інтегрованому середовищі автоматично змінюється колір вашого коду, залежно від його призначення. Наприклад, коментарі в ньому завжди позначаються зеленим кольором, ключові слова — синім, а синтаксичні помилки підкреслюються синьою хвилястою лінією, щоб вам було легше знайти та виправити їх.
За допомогою Visual Studio можна впорядковувати код шляхом збереження його частин в окремих файлах. У такий спосіб код поділяється на функціональні одиниці. Так, окремі файли слід використовувати для зберігання коду кожної з ваших форм. У Visual Studio автоматизовано процес компіляції та запуску програми: для цього достатньо лише кілька разів клацнути мишею. Завдяки утилітам налагодження, що входять до складу інтегрованого середовища, можна знаходити помилки у програмі та виводити допоміжні повідомлення під час її виконання. Перевага середовища Visual Studio полягає ще й у тому, що за його допомогою розробляють програми різними мовами, які називають мовами .NET.

— А які мови входять до складу родини мов .NET? — запитала Даринка.

— До основних мов .NET належать:

- Visual Basic .NET,
- C# («сі шарп»),
- J# («джей шарп»),
- C++ («сі плюс плюс»),
- ASP.NET (мова створення веб-сторінок).

Ми зосередимо увагу на мові Visual Basic .NET. Якщо вам цікаво, можете самостійно або за допомогою підручників ознайомитися з іншими мовами програмування.

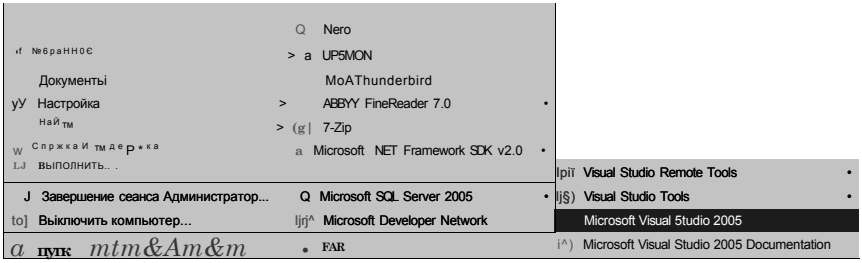
Запуск Visual Studio

Отже, перейдемо від теорії до практики. Щоб відкрити Visual Studio, виконайте такі дії.

1. Клацніть кнопку **Пуск.**
2. Виберіть меню **Програми.**

3. Виберіть папку **Microsoft Visual Studio 2005.**

4. Запустіть програму Microsoft Visual Studio 2005.



Розмістіть на робочому столі свого комп'ютера ярлик програми Visual Studio, яким можна скористатися в подальшому для її відкриття. Це дасть вам змогу прискорити пошук і запуск інтегрованого середовища: достатньо лише двічі клацнути мишею ярлик.

Пам'ятайте, що на різних комп'ютерах програма Visual Studio може розташовуватись у різних папках, тому на шкільному або чужому комп'ютері вона може перебувати в незвичному для вас місці. Якщо ви не знаєте, як відкрити Visual Studio, зверніться за допомогою до вчителя або товариша.

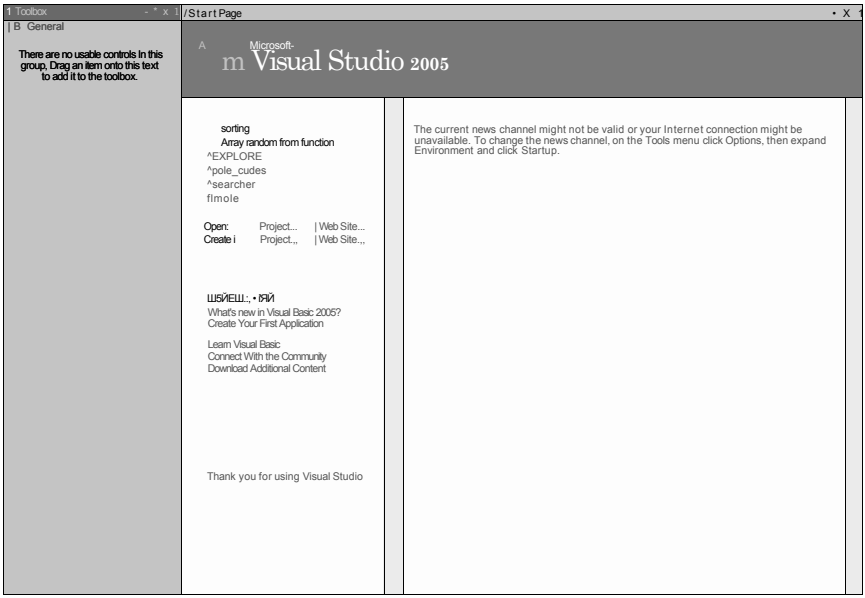
— Коли ми нарешті створимо щось дієве? — не витримала Даринка.

— Яка ж ти нетерпляча! — відповів ВВ. — Гаразд, давайте розпочнемо розробку нашої першої програми мовою Visual Basic .NET у середовищі Visual Studio, яке вже має бути відкрито на вашому комп'ютері.

Стартова сторінка інтегрованого середовища розробки

Перше зображення, яке ви побачили на екрані комп'ютера після того, як відкрили Visual Studio, — це вкладка **Start Page** (Стартова сторінка). Стандартно в ній активізовано область **Recent Projects** (Поточні проекти), що зазвичай містить список проектів. Але якщо ви працюєте з Visual Studio вперше, вона може бути порожньою.

^Ї'УЙ * ^ Л ї ' *1-е-* - • л • • ,г а

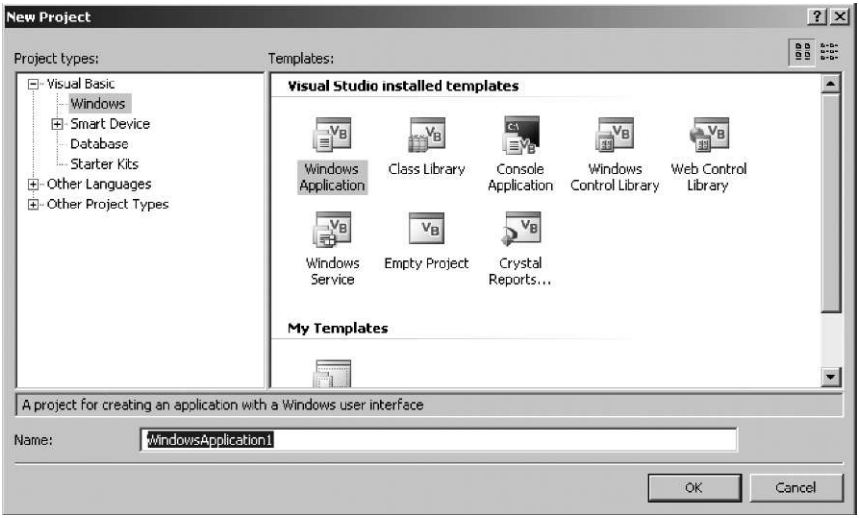


Розробка першої програми

Розглянемо область **Recent Projects** (Поточні проекти) більш детально. Використовуючи її, ви можете створити новий проект, який відобразиться у списку наявних, або відкрити той, що вже є. Для відкриття вже існуючого проекту необхідно вибрати його зі списку, або клацнути посилання **Project** (Проект) у рядку **Open** (Відкрити) та вказати шлях до проекту.

Тепер я покажу вам, як створити новий програмний проект та розробити власне програму, що називатиметься HelloWorld. Програма з такою назвою була першою для більшості програмістів. Її завдання — генерувати вікно повідомлення, в якому виводиться текст «Hello, World».

Щоб створити новий проект, клацніть посилання **Project** (Проект) у рядку **Create** (Створити). На екрані з'явиться діалогове вікно **New Project** (Новий проект).



Вікно New Project

У верхній лівій області вікна **New Project** (Новий проект), що має заголовок **Project types** (Типи проектів), містяться різні типи проектів, які можна створювати у Visual Studio. На мою думку, програма HelloWorld є Windows-програмою. В області праворуч відображуються шаблони, що відповідають певним типам проектів. Надалі ми використовуватимемо лише шаблон **Windows Application** (Застосунок Windows).

У вікні **New Project** (Новий проект) потрібно не лише обрати тип проекту та шаблон, але й задати назву проекту.

1. В області **Project types** (Типи проектів) клацніть значок **+** біля посилання **Visual Basic.** Зі списку, що розкриється, виберіть **Windows.**
2. В області **Templates** (Шаблони) виберіть **Windows Application** (Застосунок Windows).
3. Введіть назву проекту `HelloWorld` у полі **Name** (Ім'я). Зауважте, що назва не може містити пробілів.
4. Клацніть кнопку **OK.** Буде створено новий проект і відкрито вкладку **Form1.vb [Design],** де відображено форму `Form1`.

Вкладка розробки форми

Більшість Windows-програм, які ви розроблятимете, будуть починати свою роботу з виведення форми. *Форма* — це видима частина простої програми. На ній ви можете розташовувати елементи керування, наприклад кнопки або текстові поля. Коли під час виконання програми користувач клацатиме кнопку, буде виконуватися код, що їй відповідає. Тому програміст має розробити код для кожної створюваної ним кнопки та, можливо, інших елементів керування.

Отже, на екрані ви бачите порожню форму. Вкладку, на якій вона розташована, називають вкладкою розробки форми, і вона має заголовок **Form1.vb [Design].** Поки що у формі немає жодної кнопки чи текстового поля, тому я розкажу вам, як їх створити.

Вкладка Toolbox

Доступ до багатьох засобів Visual Studio здійснюють відкриванням та закриванням певних вкладок. Наприклад, щоб помістити кнопки та текстові поля у форму, потрібно відкрити вкладку **Toolbox** (Панель інструментів), на якій показано всі елементи керування, які можна розмістити у формі: кнопки, перемикачі, текстові поля, списки тощо. Саме за допомогою цієї вкладки ви згодом створюватимете цікаві форми.

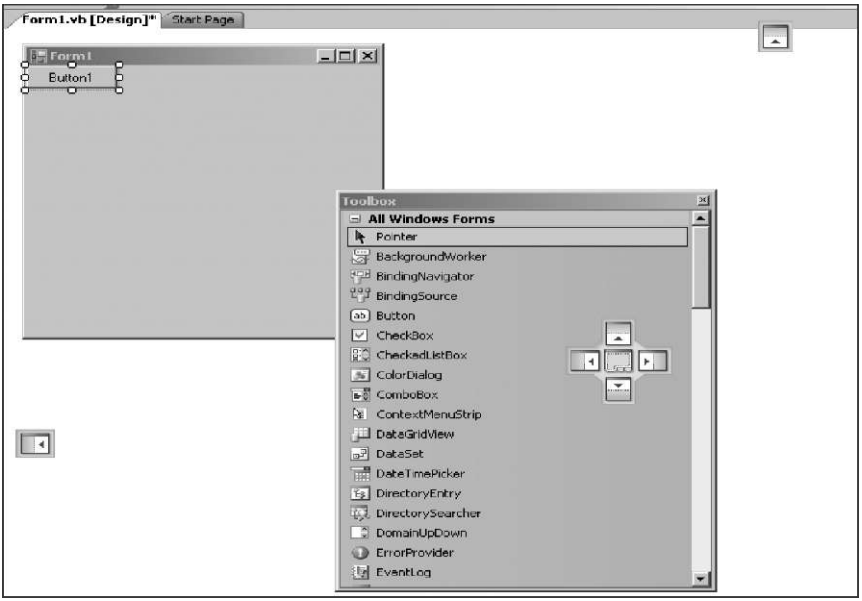
1. У головному меню виберіть команду **View** (Перегляд).
2. Виконайте команду **Toolbox** (Панель інструментів). Відобразиться однойменна вкладка.
3. Двічі клацніть елемент **Button** (Кнопка), і до форми буде додано кнопку `Button1`.

У такий самий спосіб до форми можна додати текстове поле, спробуйте зробити це самостійно.

Вкладку **Toolbox** (Панель інструментів) можна також відкрити кнопкою , що розміщується на панелі інструментів **Standard** (Стандартна).

Якщо вкладка **Toolbox** (Панель інструментів) затуляє вкладку форми, перетягніть ярличок вкладки **Toolbox** (Панель інстументів), що розташований над вкладкою біля її лівого верхнього

кута, до центру екрана й відпустіть ліву кнопку миші. Відобразиться меню, з якого виберіть команду **New Horizontal Tab Group** (Нова горизонтальна група вкладок) або команду **New Vertical Tab Group** (Нова вертикальна група вкладок). Це дасть змогу розташувати вкладку **Toolbox** (Панель інструментів) праворуч від вкладки форми або під нею.



Переміщення й змінення розміру кнопки

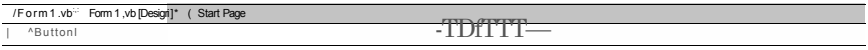
Ви вже додали кнопку до форми, а тепер я покажу, наскільки легко перемістити цю кнопку та змінити її розмір.

1. Клацніть мишею кнопку `Button1` та перетягніть її в центр форми, утримуючи натиснутою ліву кнопку миші.
2. Наведіть покажчик миші на один із білих квадратів (це маркери змінення розміру).
3. Для того щоб збільшити або зменшити розмір кнопки, перемістіть маркер змінення розміру, утримуючи ліву кнопку миші.

Тепер додамо до кнопки код, щоб виконувалася певна дія, коли користувач клацатиме кнопку.

Додавання до кнопки коду

Коли ви створили кнопку, програма Visual Studio автоматично додала кілька рядків коду, що відображені у вікні коду. Подивіться на приклад.



— А це точно правильно? — запитала Даринка. — Здається, що це написано грецькою. Чи не так?

— Не хвилюйся, — відповів ВВ, — я допоможу вам перекласти написане, коли настане час. А поки що знайдіть фразу `Button1_Click`. Розташована під нею частина коду буде виконуватися, коли ви клацнете кнопку. Якщо зараз запустити програму і клацнути кнопку, нічого не відбудеться. Спочатку необхідно вказати програмі, що вона має робити, коли ви клацнете кнопку під час її виконання. Ось до чого зводиться програмування! Готові?

1. Двічі клацніть кнопку `Button1`, відкриється вікно коду.
2. Клацніть порожній рядок над словами `End Sub`.
3. Уведіть такий рядок коду:

```
MessageBox.Show("Hello, World")
```

Цей рядок коду виконуватиметься, коли користувач клацатиме кнопку у формі.

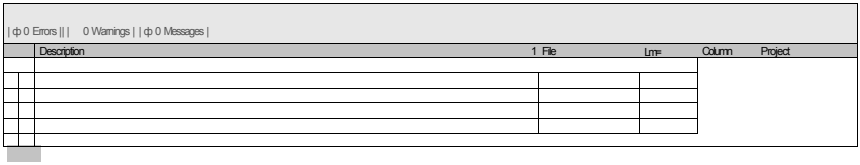
Побудова застосунку

Тепер час побудувати застосунок HelloWorld. Саме зараз код, написаний вами на Visual Studio, компілюватиметься в інструкції, які «розуміє» комп'ютер.

1. У головному меню виберіть команду **Build** (Побудова).
2. Виконайте команду **Build HelloWorld** (Побудувати HelloWorld).

Успішна побудова

Після побудови застосунку у вікні **Error List** (Список помилок), не міститиметься жодного повідомлення.

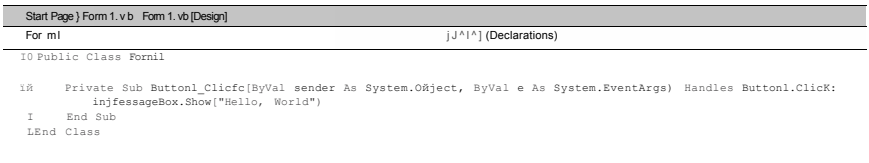


Невдала побудова

— А в мене чомусь не спрацьовує, — сумно сказав Михась.

— Ось бачиш, замість того, щоб ввести `MessageBox`, ти помилково набрав `mMessageBox`. У вікні **Error List** (Список помилок) відображується повідомлення про помилку.

Visual Basic підкреслить помилку, якщо це можливо. Коли двічі клацнути повідомлення про помилку, курсор буде встановлено на тексті, що її містить.



Якщо побудова застосунку була невдалою, поверніться назад до коду та перевірте, чи ви не зробили помилок під час його введення. Потім побудуйте застосунок знову.

Запуск програми

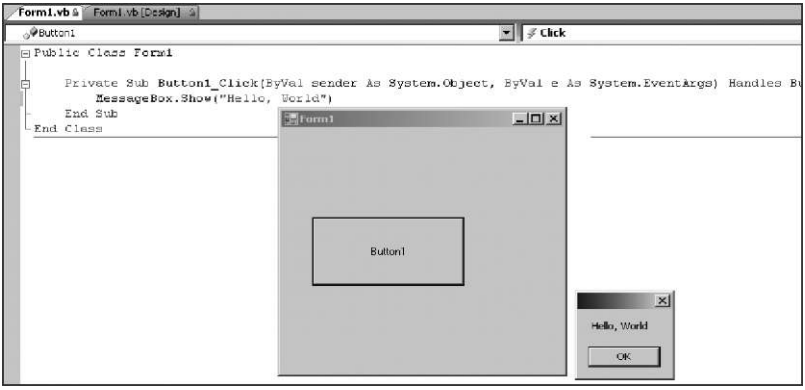
Тепер ваша програма не містить помилок і успішно скомпільована. Отже, ви вже на шляху до того, щоб стати справжніми програмістами! Час запустити вашу першу програму.

1. На панелі головного меню виберіть команду **Debug** (Налагодження).
2. Виконайте команду **Start Debugging** (Розпочати налагодження). Відобразиться форма з кнопкою, яку ви додали.
3. Клацніть кнопку.

Замість виконання команди **Debug •Start Debugging** можна клацнути кнопку • на панелі інструментів **Standard** (Стандартна).

Відображення вікна повідомлення

Після того як ви клацнете кнопку, відкриється вікно повідомлення з текстом «Hello, World». Зрештою, саме це ви й наказали робити програмі! Вітаю! Ви щойно розробили свою першу програму на Visual Basic .NET.



1. Клацніть кнопку **OK,** щоб закрити вікно повідомлення. Воно зникне, а ваша форма залишиться на екрані. Ви можете знову клацнути кнопку, щоб удруге вивести на екран вікно повідомлення.
2. Для завершення програми клацніть кнопку X у правому верхньому куті форми.

Збереження проекту

Тепер час зберегти проект. Цей процес подібний до збереження документа в будь-якій іншій Windows-програмі.

1. У головному меню виберіть пункт **File** (Файл).
2. Виконайте команду **Save All** (Зберегти все).
3. Для того щоб вийти з Visual Studio, виберіть меню **File** (Файл), а в ньому — команду **Exit** (Вийти).

Спробуйте зробити це самі, виконавши вправу, що міститься на моєму кишеньковому комп'ютері.

Вправа 2.1. Привітання

Напишіть програму HelloPeople, що виводитиме повідомлення «Hello, People». Для цього виконайте дії, аналогічні тим, що описані вище.

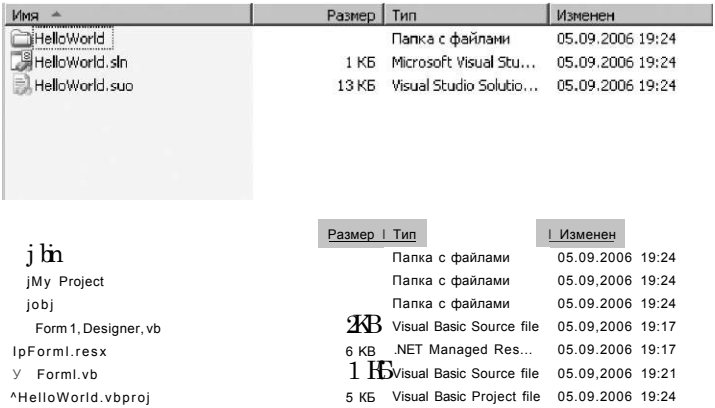
Файли та папки проекту

Коли ви створюєте програму, генерується багато файлів. Ви маєте знати, які файли створюються і для чого вони призначені. Використовуючи Провідник Windows, знайдіть папку, де зберігаються проекти Visual Studio. Розкрийте її, і побачите, що для проекту було створено папку **HelloWorld.** Щоб побачити файли, створені Visual Studio для вашого проекту, розкрийте папку **HelloWorld** та її підпапку **HelloWorld.** Найважливішими з цих файлів є такі:

- **HelloWorld.sln** — файл, в якому розміщуються відомості про всі інші файли розробки. Розробка — це набір кількох проектів, але ваша розробка HelloWorld містить лише один файл проекту — **HelloWorld.vbproj.** Щоб відкрити вже створений проект HelloWorld, достатньо буде двічі клацнути файл **HelloWorld.sln;**
- **Form1.vb** — файл, який містить форму та пов'язаний із нею код.

Тепер розкрийте папку **bin\Release.** У ній розміщується виконуваний файл програми **HelloWorld.exe,** який можна запустити на

будь-якому комп'ютері, навіть якщо на ньому не встановлено Visual Studio.



Отже, спробуйте виконати ще кілька вправ, і підемо дивитися телевізор.

Запустити виконуваний файл на іншому комп'ютері можна лише тоді, коли на ньому встановлене спеціальне програмне забезпечення, яке називають .NET Framework (Каркас .NET) і яке дає можливість виконувати програми, розроблені в середовищі .NET.

Вправа 2.2. Як тебе звати

В цій вправі ми напишемо програму MyNameIs, що подібна до програми HelloWorld, яку ви вже розробили.

Створіть проект MyNameIs. На стартовій сторінці клацніть посилання **Project** у рядку **Create** або виконайте команду **File • New Project.**

До форми `Forml` додайте кнопку та текстове поле. Залиште для них імена, які використовують стандартно — `Buttonl` і `TextBoxl`.

Щоб відкрити сторінку коду, двічі клацніть кнопку `Buttonl`. У підпрограмі `Button1_Click` уведіть такий рядок коду:

```
TextBoxl.Text = "ваше ім'я"
```

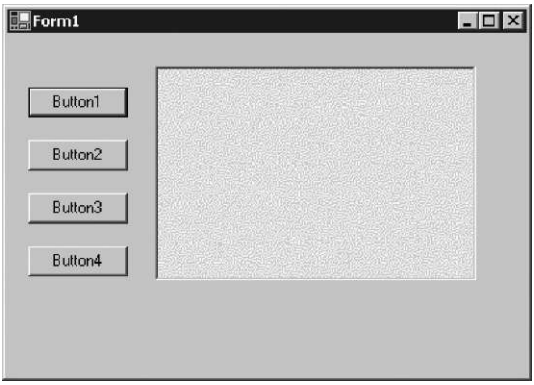
Слова `ваше ім' я` замініть своїм справжнім іменем (обов'язково збережіть лапки). Побудуйте та запустіть програму. Що відбувається, коли ви клацаєте кнопку?

Завдання 2. Кав'ярня

Михась за вечерею довго пояснював ВВ, що таке кав'ярня. ВВ попросив його намалювати форму, що керуватиме роботом, який має обслуговувати цей заклад.

— Невже без роботів не можна обійтися? — здивувалася Даринка, але ВВ не звернув на запитання уваги і продовжив розповідь:

— Тепер ви вже знаєте, як додати до форми кнопку, побудувати програму та запустити її. У папці **Завдання_2** міститься готова програма, до якої потрібно лише додати кнопки `Button1`–`Button4`. Після додавання кнопок форма має набути такого вигляду, як на рисунку.



Побудуйте програму, а потім виконайте команду **Debug** • **Start.** Що відбувається, коли ви клацаєте кнопки? Перегляньте вікно коду і спробуйте пояснити, чому це відбувається.

Додаткове завдання

Дізнайтеся самостійно, як змінювати текст на кнопці, користуючись вікном **Properties** (Атрибути).

— Отже, тепер ми можемо подивитися телевізор, — сказав ВВ після того, як Даринка та Михась виконали всі вправи, що він їм задав. — Але мій тато і тут поставив кодовий замок!

— І щоб його відкрити, нам потрібно відповісти на тестові запитання? — запитав Михась.

— Так, адже якщо цього не зробити, то ми взагалі не вийдемо з цієї кімнати, — відповів весело ВВ.

Кодовий замок

1. Яке програмне забезпечення необхідне для того, щоб виконувалися програми, розроблені в середовищі .NET?
   а) середовище розробки Visual Studio;
   б) каркас .NET Framework;
   в) операційна система Windows XP.
2. Яке вікно використовують для додавання елементів керування до форми?
   а) **Properties;**
   б) **Toolbox;**
   в) **Output.**
3. Що потрібно зробити, перш ніж виконати (запустити) програму?
   а) побудувати розробку;
   б) налагодити програму;
   в) закрити форму.
4. Яка мова не належить до родини .NET?
   а) Visual Basic;
   б) Pascal;
   в) C#.
5. Синтаксичні помилки у середовищі .NET:
   а) підкреслюються зеленою хвилястою лінією;
   б) підкреслюються синьою хвилястою лінією;
   в) не розпізнаються.

6. На вкладці **Toolbox** (Панель інструментів):

а) розташовані зарезервовані слова;

б) виводяться помилки й поради;

в) розташовані елементи керування.

7. Яке розширення має файл, що містить відомості про всі інші файли розробки Visual Studio?

а) **sln;**

б) **vbproj;**

в) **exe.**

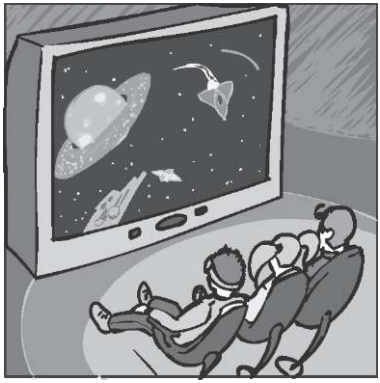
8. Проект Visual Studio на жорсткому диску зберігається у вигляді:

а) одного текстового файлу;

б) одного файлу програми, готової до запуску;

в) папки чи групи файлів.

День 3
Кодування



Вранці Даринка та Михась прийшли до кімнати ВВ, щоб разом продовжити навчання. Але через те, що вони допізна дивилися телевізор, вчитися писати код не було ніякого бажання. Тому ВВ пообіцяв, що цього разу не примушуватиме їх писати жодного рядка коду.

— Сьогодні ми замість коду будемо вчитися писати псевдокод! — знайшов вихід із ситуації ВВ.

— А що це таке? — поцікавилася Даринка.

— Це щось на зразок коментарів? — запитав Михась.

— Так, і коментарі також. Давайте розберемося з усім поступово, — відповів ВВ і розпочав чергову розповідь.

Які є оператори

Ви знаєте, що можна написати комп'ютерну програму, навіть не знаючи жодної мови програмування? Багато програмістів пише свої програми спочатку в «псевдокоді», а потім перекладає їх мовою програмування. Так їм легше визначити логічну структуру програми та її функції. Псевдокод складається із простих інструкцій, записаних англійською або іншими мовами, які пояснюють, що має робити програма. Коли псевдокод готовий, легше написати справжній код, адже псевдокод визначає структуру програми. Перш ніж я покажу,

як користуватися псевдокодом, хочу познайомити вас із основними типами операторів, синтаксис яких у різних мовах різний. Комбінуючи ці оператори, ви зможете написати дуже складні програми.

— Ми ж домовлялися, що не писатимемо код! — простогнала Даринка.

— Я не порушую наших домовленостей. Ми не будемо нічого писати, а лише розглянемо основні типи операторів! — заспокоїв її ВВ.

Пригадуєте рядок коду, який ви вводили, коли створювали програму HelloWorld?

```
MessageBox.Show("Hello, World")
```

Це оператор, який називають оператором виведення даних. Він є одним із базових різновидів операторів. В усіх мовах програмування міститься однаковий набір базових операторів, який використовують для виведення даних. Хоча ці оператори мають однакове призначення, їх синтаксис залежить від мови. Різні мови програмування підтримують й інші базові мовні конструкції. Розглянемо їх.

- *Змінні* використовують для збереження інформації, зокрема чисел та тексту. Вони є «вмістилищами» даних.
- За допомогою *операторів присвоєння* змінним надають певні значення, які можуть мати вигляд чисел, тексту, інших змінних або результатів обчислень.
- *Оператори прийняття рішення* використовують для того, щоб вибирати сценарій подальших дій. Вибір роблять на основі порівняння. Якщо результат порівняння є істинним, обирають один сценарій, якщо хибним — інший. Такі оператори часто називають розгалуженнями, або умовними операторами, оскільки після їх виконання обчислення можуть бути спрямовані в одному чи іншому напрямку залежно від істинності певної умови.
- *Циклічні оператори* використовують для повторного виконання операцій. Завдяки цим операторам зменшується обсяг коду, який необхідно написати для того, щоб програма багаторазово виконувала одну й ту саму дію.

— Схожі конструкції, здається, є в усіх мовах програмування! — пригадав Михась розповідь свого тата.

— Так, у більшості мов програмування є всі ці конструкції, адже саме на них ґрунтується мислення, а тому й кожна програма! — відповів ВВ.

Якщо ви навчитеся програмувати, то вмітимете користуватися базовими операторами, для того щоб «змусити» програму робити те, що вам потрібно. Під час навчання мови Visual Basic .NET (або будь-якій іншій мові програмування) ваше завдання насамперед полягатиме в тому, щоб вивчити певний синтаксис, необхідний для написання та використання базових операторів.

Псевдокод

Я вже вам розповідав, що програмісти часто пишуть перші варіанти своїх програм на псевдокоді. Слово «псевдо» означає «замість». Наприклад, деякі автори користуються псевдонімами замість своїх справжніх імен. Псевдокод пишуть «несправжньою» мовою програмування. Він складається із простих речень, які описують, що робитиме програма. Ви можете написати псевдокод англійською, українською, латиною або будь-якою іншою мовою: комп'ютер усе одно ніколи не буде його читати. Ваш псевдокод має бути зрозумілим, оскільки він містить інструкції для програміста, який писатиме справжній код програми. Псевдокод пишуть, використовуючи різні типи базових операторів: присвоєння, порівняння, прийняття рішення тощо.

Перевага псевдокоду полягає у тому, що він є незалежним від мови програмування. Спочатку визначте логіку та структуру програми, використовуючи псевдокод, а потім «перекладіть» його певною мовою програмування.

Дозвольте навести приклад використання псевдокоду для розробки програми. Припустімо, я хочу поїхати до бабусі. Для цього мені варто розрахувати ціну пального, яке потрібно мати для подорожі. Якщо пальне коштуватиме надто дорого, я можу скористатися громадським транспортом.

— А яким пальним ви користуєтеся? Невже бензином? — перебив Михась свого друга.

— Звісно ні, — відповів той. — Бензин — це надто дороге для нас пальне. Зараз їздять на швидких автомобілях, які заправляють водневим пальним, і на дещо повільніших, але дешевших, що мають електродвигун. Тому я замість пального використовую електричний струм. Отже, давайте повернемося до дій, які вам необхідно виконати, щоб обчислити середню вартість струму в кредитах на трубель.

— Трубель — це щось типу кіловат-години? — запитав Михась, почувши незнайоме слово.

— Так, щось подібне до того, але значно більше. Витрати струму в нас значно збільшилися, а тому стало незручно користуватися застарілими величинами, — відповів ВВ. — Щоб обчислити середню вартість струму, нам необхідно виконати такі дії:

1. Підсумувати всі кошти, які було витрачено на струм, та надати значення знайденої суми змінній (усього кредитів).
2. Додати кількості трубелів струму, які ми використаємо, та присвоїти результат іншій змінній (усього трубелів).
3. Першу змінну (всього кредитів) поділити на другу змінну (всього трубелів) та присвоїти результат третій змінній (кредитів на трубель).
4. Вивести повідомлення, де б зазначалася середня вартість струму в кредитах на трубель.

Ми щойно записали псевдокод, який чітко визначає, що робитиме ваша програма. Тепер можна написати код, що розв'язуватиме визначені псевдокодом задачі.

Візьмемо інший приклад, який передбачає більше дій. Запрограмуємо робота, який би замінював спущені шини.

1. З'ясуйте, яка саме шина спущена (оператор порівняння).
2. Визначте розмір спущеної шини (оператор присвоєння).
3. Якщо для заміни немає шини відповідного розміру, зверніться до автомагазину, інакше — здійсніть кроки 4-11 (оператори порівняння та прийняття рішення).

4. Підійміть машину, використовуючи для цього гідравлічний домкрат.
5. За допомогою викрутки зніміть кришку, яка закриває центральну частину колеса.
6. Використовуючи гайковий ключ, зніміть усі гайки (циклічний оператор).
7. Зніміть пошкоджену шину.
8. Поставте нову шину.
9. За допомогою гайкового ключа встановіть та закрутіть усі гайки (циклічний оператор).

10. Вставте кришку на місце та забийте її.
11. І насамкінець опустіть гідравлічний домкрат.

У наведеному вище псевдокоді використано комбінацію операторів присвоєння, порівняння, прийняття рішення та циклічних операторів. Наприклад, щоб запам'ятати розмір спущеної шини, застосовують оператор присвоєння, а щоб порівняти розмір спущеної шини з розміром неспущеної — оператор порівняння. Оператор прийняття рішення застосовують разом з оператором порівняння. Якщо у вас є шина потрібного розміру, робота триватиме, інакше доведеться відвідати автомагазин. Зняття та закручування гайок можна розглядати як циклічні процеси. Кожен із них потрібно повторити кілька разів, доки всі гайки не буде знято, а потім — закручено.

Чому б вам не спробувати написати псевдокод? Я думаю, що це варто зробити саме зараз.

Вправа 3.1. Як зібратися в дорогу

Напишіть псевдокод, що пояснюватиме дії, які вам необхідно виконати, щоб зібратися в дорогу. У вашому розв'язанні має бути принаймні сім дій. Якщо кожен пункт починається зі слова «якщо», ви не зрозуміли сутність псевдокоду.

Коментарі у програмах

За допомогою сучасних мов програмування, до яких належить зокрема й Visual Basic .NET, до коду можна додавати коментарі. Вони, на відміну від операторів, не компілюються і не виконуються після того, як запускається програма. *Коментарі* — це спосіб задокументовування коду. Додавати коментарі до коду варто завжди, адже їх наявність є ознакою досвідченості програміста, що старанно обмірковує та документує код, який він пише. Записуйте коментарі під час змін коду або додавання певних функціональних можливостей. Крім того, слід супроводжувати коментарями ті частини коду, які важко зрозуміти або які відповідають за надзвичайно складні обчислення.

Додавайте до коду коментарі, але не надто багато. Адже ви документуєте код, а не пишете роман.

У мові Visual Basic .NET коментарі починаються з одинарних лапок ('). Будь-який рядок коду, що починається з такого символу, інтерпретується як коментар, і не буде компілюватися та виконуватися під час запуску програми. Наведемо приклад коментарю в мові Visual Basic .NET:

```
'У вікні повідомлення буде виведено "Всім привіт!"
MessageBox.Show("ВсіМ привіт!")
```

У Visual Basic .NET для коментарів використовують одинарні лапки, а от у C# та J# на початку рядка коментарю записують дві скісні риски (//). Наприклад:

```
//Тепер виконується функція перевірки вхідних даних
```

Окрім документування коду, у коментарів є й інша функція: вони можуть забезпечити невиконання рядків справжнього коду. Такий процес називають «закоментовуванням» коду. За його допомогою ви можете шукати в коді помилки. Якщо у програмі є помилка, то можна «закоментовувати» рядки або цілі блоки коду, доки помилка не перестане виникати. Код, винесений у коментарі останнім, імовірно, і спричиняє помилку.

Зараз я наведу приклад винесення коду в коментарі. Зауважте, що із трьох рядків коду «закоментовано» другий, бо саме на

його початку записано одинарні лапки. Отже, перший і третій рядки виконуватимуться, а другий — ні!

```
MessageBox.Show("Всім привіт!")
'MessageBox.ShowC'А це ви ніколи не побачите!")
MessageBox.Show("Вітаю вас із першим запуском моєї програми")
```

Коментарі та псевдокод

А тепер — про одне з найцікавіших і найцінніших застосувань коментарів. Не здогадуєтесь, про що йтиметься? Коментарі можна використовувати для розробки та планування програми. Це дуже легко. Ви вже написали псевдокод, що визначає дії вашої програми. Тож скористайтеся ним. Скопіюйте свій псевдокод у програму та поставте одинарні лапки перед кожним його рядком, щоб перетворити псевдокод на коментар, а потім напишіть під кожним рядком коментарів справжній код мовою Visual Basic .NET. Тепер у вас є не просто готова програма — вона ще й має коментарі.

Візьмемо, наприклад, псевдокод, що я написав для обчислення середньої вартості струму в кредитах на трубель. Записавши під кожним рядком псевдокоду відповідний оператор мови Visual Basic .NET, отримаємо готову програму.

```
'Підсумувати всі кошти, які ми витратили на струм,
'та присвоїти суму змінній (усього кредитів)
Dim totalCredits As Double
totalCredits = 88.76 + 100.8 + 120.3

'Додати кількості трубелів струму, які ми використали,
'та присвоїти результат іншій змінній (усього трубелів)
Dim totalTrubels As Double
totalTrubels = 49.3 + 55.5 + 46.3

'Першу змінну (усього кредитів) поділити на
'другу змінну (усього трубелів) та присвоїти результат
'третій змінній (кредитів на трубель)
Dim CreditsPerTrubel As Double
CreditsPerTrubel = totalCredits / totalTrubels

'Вивести повідомлення, де зазначено середню
'вартість струму в кредитах на трубель
MessageBox.Show(CreditsPerTrubel)
```

Як зробити код читабельним

Хоча ви, можливо, й не розумієте весь код Visual Basic .NET, що наведений вище, але зверніть увагу на кілька речей. По-перше, ви бачите, що всі коментарі починаються з одинарних лапок. Окрім того, у Visual Studio коментарі виділені зеленим кольором — так їх легше відрізняти від коду. По-друге, зауважте, що перед кожним коментарем з'явився порожній рядок. Такі рядки поділяють код та коментарі на групи за функціональним призначенням. Завдяки порожнім рядкам програма стає більш зрозумілою.

У більшості випадків середовище Visual Studio автоматично додає пробіли та відступи, щоб зробити код читабельним. Ви можете зробити це також і власноруч, адже мови родини .NET «не зважають» на порожні рядки або відступи, які ігноруються під час компіляції програми. На відміну від компілятора людині це не байдуже, бо їй зручніше читати код, в якому є і пробіли, і відступи.

На моєму кишеньковому комп'ютері є ще дві вправи, які вам теж мають сподобатись.

Вправа 3.2. Якщо немає посудомийної машини

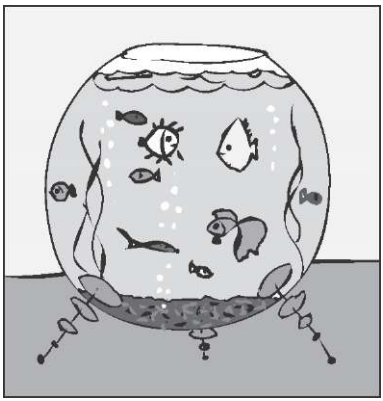
У фургоні, в якому ми подорожуємо, є кухня — ми нею користуємося, коли поруч немає затишного ресторану. Після приготування їжі доведеться мити посуд, а посудомийної машини у фургоні немає.

Опишіть псевдокодом процес миття посуду. Якщо ви не знаєте, як це робити, проведіть відповідне польове дослідження. У вашому розв'язанні має бути принаймні сім дій.

Завдання 3. Кольорові рибки

Перед вами акваріум? Полічіть, по скільки рибок того чи іншого кольору плаває в ньому, а потім визначте відсоткову частку риб кожного кольору.

У мене для цього є програма. Знайдіть у папці **Завдання 3** файл, який має назву **FishTank.sln**. Двічі клацніть його ім'я, щоб відкрити Visual Studio .NET. Перейдіть на сторінку коду.



Ваше завдання — уважно переглянути код та скопіювати всі коментарі в документ Word або інший текстовий документ.
Скажіть, чи не схожий результат на псевдокод?

— Отже, розгляд нашої теми завершено. На мою думку, ви добре попрацювали, тому заслуговуєте на відпочинок, — сказав ВВ, схвально подивившись на своїх друзів.

— Може пограємо в якусь комп'ютерну гру? — запитала Даринка.

— Слушна пропозиція. Я вже навіть гру підібрав, — відповів ВВ, а потім зауважив: — Але щоб отримати до неї доступ, вам необхідно відкрити кодовий замок.

Кодовий замок

1. У якій послідовності зазвичай розробляють програму?
   а) спочатку пишуть псевдокод, потім — програму;
   б) спочатку пишуть програму, потім — псевдокод у вигляді коментарів;
   в) псевдокод пишуть у вигляді коментарів як до створення програми, так і під час її написання.
2. Яку конструкцію використовують для повторного виконання операцій?
   а) оператор порівняння;
   б) циклічний оператор;
   в) оператор присвоєння.

3. Що таке псевдокод?

а) справжня мова програмування;

б) прості речення, в яких описано, що робитиме програма;

в) вигадане ім'я автора.

4. Додавання яких елементів до програми дає змогу уникнути помилок?

а) коментарів;

б) відступів;

в) і коментарів, і відступів.

5. Оператори прийняття рішення передбачають:

а) порівняння;

б) виконання однієї дії, коли результат порівняння істинний, та іншої, коли результат порівняння хибний;

в) і те й інше.

6. Чи залежить синтаксис коментарів від мови програмування?

а) так, залежить;

б) ні, не залежить;

в) в усіх мовах родини .NET однаковий.

7. За допомогою коментарів можна:

а) написати програму;

б) виправити помилки у програмі;

в) знайти помилки у програмі.

8. Що відбувається з розміром відкомпільованої програми, якщо до коду дописати коментарі?

а) збільшується;

б) не змінюється;

в) зменшується.

День 4
Форми

Прокинувшись наступного ранку (точніше, це було ближче до обіду), Михась та Даринка не могли думати ні про що інше, крім комп'ютерних ігор, що стали незрівнянно реалістичнішими, ніж ті, в які вони грали вдома, перебуваючи у своєму часі. Отже, ВВ приклав чимало зусиль, щоб знову змусити їх працювати.

— Давайте спочатку вивчимо щось нове, розберемося з цікавим матеріалом, а тоді розважатимемося! — вкотре вигукнув ВВ. — Та й кодовий замок не відкриється, якщо ви не зможете відповісти на тестові запитання! Вони, між іншим, щоразу стають складнішими!

— А у нас сьогодні справді буде цікава тема? — запитала Даринка.

— Звісно. Одна з найцікавіших, — запевнив її ВВ.

— Ну, добре, — відірвався нарешті Михась від спогадів про віртуальну реальність. — Давай, розповідай щось нове, а потім ми зможемо продовжити знайомитися з чудесами вашої ігрової індустрії.

— Отже, починаємо, — сказав ВВ. — Не в усіх програмах є інтерфейс користувача: деякі запускаються лише один раз або виконуються у фоновому режимі й не потребують введення даних. Раніше інтерфейс не був настільки важливим і поширеним явищем.

— Справді! — вигукнув Михась. — Наш тато розповідав, що раніше інтерфейс був текстовим, тобто робота відбувалася в режимі запитання/відповідь, однак із появою графічних операційних систем почали з'являтися і програми з графічним інтерфейсом.

— Так, звичайно. Але що таке інтерфейс? Пам'ятаєте, ви застосовували середовище Visual Studio, щоб написати свою першу програму? Перше ваше рішення стосувалося типу проекту. Ви обрали Windows-програму, що має графічний інтерфейс, за допомогою якого користувач може взаємодіяти з нею, — нагадав ВВ.

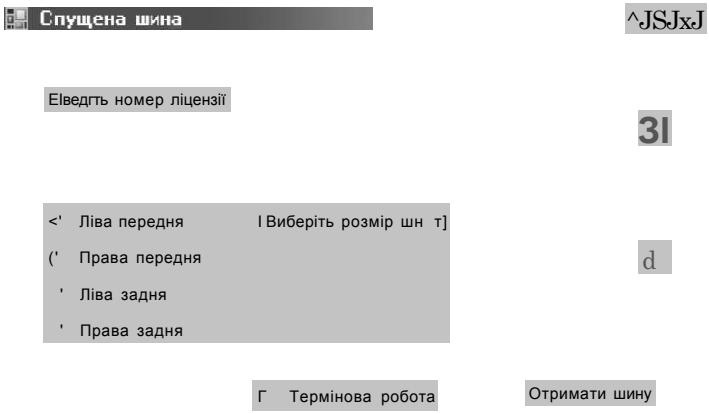
— На мою думку, програма, яка «не запитує» нічого в користувача, може виконувати тільки наперед запрограмовані дії та є суто спеціалізованою, — зауважила Даринка.

— Я не можу цілком із тобою погодитися, — висловив свою думку ВВ, — хоча в основному так воно і є. Зокрема, до такого типу програм належать більшість вірусів. Але повернімося до простіших програм — тих, які мають графічний інтерфейс.

Конструювання форм

З Windows-програмою користувач може взаємодіяти за допомогою форм. Коли ви вперше створюватимете Windows-програму мовою Visual Basic, у середовищі Visual Studio до проекту буде автоматично додано форму **Form1.vb.** Щоб користувач міг взаємодіяти із програмою, потрібно додати до форми *елементи керування:* кнопки, написи, текстові поля, списки, перемикачі, прапорці тощо. Припустімо, що у програмі є форма `Спущена шина`. Ось перелік елементів керування, що необхідні для такої форми:

- текстове поле для введення номерного знака автомобіля;
- група перемикачів для визначення, яка саме шина спущена (ліва передня, права передня, ліва задня, права задня);
- прапорець, установлення якого вказує на терміновість роботи;
- поле зі списком для вибору розміру шини;
- кнопка для пошуку шини в інвентарному переліку;
- текстове поле, в якому виводитиметься список наявних шин (у ньому буде зазначено: хто є виробником шини, номер моделі, розмір, вартість та наявна кількість).



— А що, у вас шини теж проколюються? — запитав Михась.

— Так, — відповів ВВ, — хоча сучасна гума набагато міцніша та еластичніша, аеромобілі все ще пошкоджуються під час посадки, тому проблеми залишаються. Отже, — продовжив ВВ, — у формі `Спущена шина` є два текстових поля, група з чотирьох перемикачів, список, прапорець і кнопка. Ви можете по-різному розташувати ці елементи на формі. Саме тут починається цікава та творча робота.

— Для мене розробка форми — це один із найцікавіших аспектів програмування, — вирішила Даринка. — Мені подобається малювати, а форма подібна до полотна художника.

— А елементи керування — до пензлів, — додав Михась.

— Однак тобі слід бути обережною. Передусім ти мусиш переконатися, що форма відповідає своєму призначенню. Користування формою має бути легким, у ній не може бути нічого зайвого. Коли ти розробляєш форму, не переставай себе запитувати: «Для чого вона потрібна? Які функції вона буде виконувати?» Крім того, слід переконатися, що форма має гарний вигляд! У ній не може бути багато контрастних кольорів чи дражливих візуальних ефектів, які відволікатимуть користувача (наприклад, тексту, що миготить). Елементи керування у формі потрібно розташовувати акуратно та логічно. Ти маєш

продумати порядок, в якому користувач вводитиме інформацію та взаємодіятиме із програмою. Іноді слід попереджати користувача про те, щоб він не ввів усієї потрібної інформації в одному полі, або про те, що деяка інформація введена неправильно. Коли ти розробляєш форми, то мусиш міркувати як користувач, та уявляти всі ймовірні способи взаємодії користувача із програмою, — зауважив ВВ Даринці, а потім продовжив далі свою розповідь, звертаючись вже до обох.
Нарешті, коли ви перейдете до розробки складніших програм, візьміть до уваги, що всі Windows-програми мають стандартний стиль та вигляд. Користувач очікує, що ваша програма буде відповідати певним нормам і працюватиме в певному стилі. Наприклад, першим меню у Windows-програмі має бути меню **File** (Файл), яке міститиме команду **Exit** (Вихід), призначену для виходу із програми.

Користувач буде невдоволений, якщо він клацне кнопку з написом «Обчислити результати», а програма натомість змінить колір головної форми, або всі написи на кнопках перекладе латиною.

Форми у програмах

Ви вже маєте знати, що коли розробляється Windows-програма, Visual Studio автоматично створює форму **Form1.vb,** яка є головною, або стартовою у вашій програмі. Саме її побачить користувач, коли відкриє програму вперше. Головна форма подібна до центральної панелі керування. З неї користувач може отримати доступ до будь-якої іншої частини програми. Зазвичай вздовж верхнього краю головної форми розміщується рядок меню або панель інструментів, команди якої відповідають різним функціям програми.

У головній формі користувач обирає пункти меню або клацає кнопки, що відкривають інші форми, які можна застосовувати для введення чи перегляду інформації або для взаємодії з програмою в інший спосіб. Такі форми мають певне призначення, наприклад, за їх допомогою можна демонструвати рахунок гри у боулінг, ввести список побічних ефектів або

модифікувати значення параметрів програми. Зазвичай після введення або перегляду даних в одній із таких форм користувач її закриває та повертається до головної форми. Хоча Visual Studio автоматично створює лише одну форму **Form1.vb,** ви самостійно можете додати до свого проекту стільки форм, скільки потрібно.

Іншим важливим типом форм є вікно повідомлення. Ви вже користувалися ним у програмі HelloWorld. Воно призначене для того, щоб виводити на екран будь-які повідомлення у діалоговій формі. Прочитавши повідомлення, користувач клацає кнопку **OK** і закриває вікно.

Отже, вам слід виконати завдання: створити форму під назвою `Спущена шина`, а потім додати до неї кілька елементів керування та визначити їхні атрибути. Я покажу вам, як елементи керування у формі реагують на клацання мишею кнопки. Крім того, ви знову працюватимете зі спеціальною вбудованою формою — вікном повідомлення.

Атрибути форми

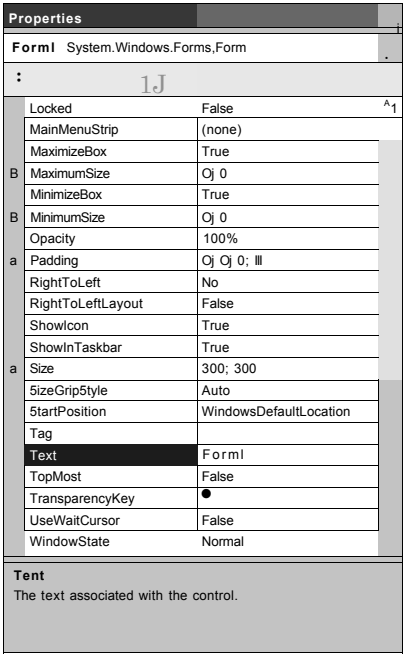
Після того як ви створите нову Windows-програму, до неї буде автоматично додано форму `Form1`, що має стандартний розмір 300x300 пікселів.

— А звідки ти знаєш цей розмір? — здивувався Михась, дивлячись на сіренький прямокутничок форми.

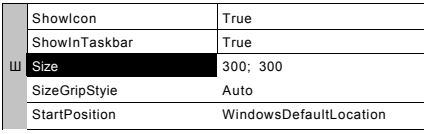
— Його можна легко віднайти у вікні **Properties** (Атрибути). У цьому вікні ви можете не лише переглянути, але й змінити розмір форми, — відповів ВВ.

Отже, виконаймо разом кілька простих дій.

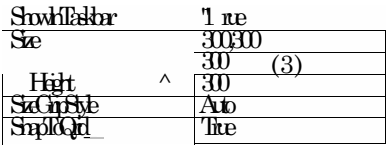
1. Створіть нову Windows-програму з назвою FlatTire. Після створення проекту клацніть один раз форму `Form1`, щоб виділити її, а потім натисніть клавішу **F4,** щоб відкрити вікно **Properties** (Атрибути).
2. Оскільки обрано форму `Form1`, вікно **Properties** (Атрибути) міститиме атрибути саме цієї форми. Зауважте, що стандартно виділено атрибут **Text** (Текст). Змініть його значення на `Спущена шина`.



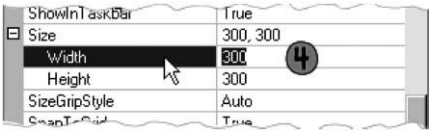
3. Знайдіть атрибут **Size** (Розмір). Зауважте, що зліва від нього розміщений значок +. Це означає, що атрибут є насправді групою атрибутів, яку можна розкрити.



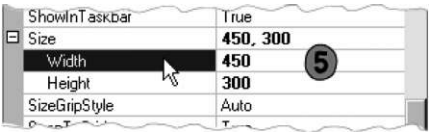
4. Клацніть значок + поруч із категорією **Size** (Розмір). Відобразяться атрибути **Height** (Висота) та **Width** (Ширина). Висота та ширина форми становлять 300 пікселів. Змінимо ці розміри.



5 Виділіть значення атрибута **Width** (Ширина), двічі клацнувши його, а потім натисніть клавішу **Backspace,** щоб видалити це значення.



6. Введіть число 450 і натисніть клавішу **Enter.** Зроблено! Ви щойно змінили ширину форми. Зверніть увагу на те, як розшириться форма, коли ви натиснете клавішу **Enter.** Зауважте також, що поруч із атрибутом **Size** (Розмір) було встановлено нове значення атрибута **Width** (Ширина).



Користуючись смугою прокрутки, у вікні **Properties** (Атрибути) знайдіть атрибут **BackColor** (Колір фону), за допомогою якого визначається колір форми, і виділіть його. Праворуч від слова **Control** (Керування) у рядку атрибута **BackColor** (Колір фону) буде відображено кнопку зі стрілкою, спрямованою вниз. Якщо ви клацнете її, то відкриється палітра кольорів, в якій оберіть вкладку **Custom** (Вибір). Двічі клацніть зразок кольору, який вам більше до вподоби. Мені здається, що краще пасуватиме світло-жовтий, адже він такий яскравий.

Створіть проект та запустіть його на виконання. Зверніть увагу на новий колір форми. Так вона виглядає веселіше! Крім того, тепер ваша форма перетворилась із квадратної на прямокутну, а також має заголовок Спущена шина.

Отже, ви вже знаєте, як змінювати атрибути форми за допомогою вікна **Properties** (Атрибути). Прокрутіть список атрибутів

форми, щоб ознайомитися з ними всіма. За допомогою вікна **Properties** (Атрибути) можна змінювати атрибути форми під час розробки, тобто, коли ви створюєте форму. Крім того, воно призначене для настроювання атрибутів кнопок, текстових полів та інших елементів керування.

Елементи керування та їхні атрибути

Ви вже помітили, що на вкладці **Toolbox** (Панель інструментів) є багато різних елементів керування. Я покажу вам, як застосовувати найпоширеніші з них та задавати значення їхніх найважливіших атрибутів. Почнемо з написів.

Напис

Написи використовують для позначення елементів керування чи їхніх груп та для виведення допоміжних інструкцій або повідомлень. Відкрийте вкладку **Toolbox** (Панель інструментів) та двічі клацніть елемент керування **Label** (Напис). До форми буде додано напис. Виділіть напис `Label1` та відкрийте вікно **Properties** (Атрибути). Тепер у ньому відображено список атрибутів напису, в якому знайдіть атрибут **Text** (Текст). Виділіть текстовий рядок, двічі клацнувши мишею, та видаліть його, натиснувши клавішу **Backspace.** Натомість введіть слова `Введіть номер ліцензії`. Перетягніть напис на відповідне місце, утримуючи ліву кнопку миші.

Текстове поле

Текстове поле можна використовувати для відображення або отримання інформації. На вкладці **Toolbox** (Панель інструментів) двічі клацніть елемент керування **TextBox** (Текстове поле). До форми буде додано текстове поле `Textbox1`. Перетягніть його,

помістивши праворуч від напису. Щоб можна було вводити текст у поле, атрибуту **ReadOnly** (Тільки для читання) потрібно надати значення `False`. Зауважимо, що це стандартне значення.
Текстове поле часто використовують для відображення великого обсягу інформації. Якщо текст дуже довгий, ви можете додати до текстового поля смуги прокрутки. Попрактикуємось у цьому. Додайте до форми ще одне поле. Атрибуту **Multiline** (Багаторядкове) надайте значення `True`. Висоту (атрибут **Height)** та ширину (атрибут **Width)** зробіть рівними 100 пікселям. Майте на увазі, якщо атрибуту **Multiline** (Багаторядкове) присвоїти значення `False`, ви не зможете змінити атрибут **Height** (Висота). Атрибуту **Scrollbars** (Смуги прокрутки) надайте значення `Vertical` (Вертикальна), а атрибуту **ReadOnly** (Тільки для читання) — значення `True`, щоб користувач не міг вводити до поля текст.

Група перемикачів

Перемикачі дають змогу користувачеві зробити вибір. Він може встановити лише один перемикач у групі. Це схоже на тест, де можна вибрати один варіант із кількох запропонованих.
На вкладці **Toolbox** (Панель інструментів) двічі клацніть елемент керування **RadioButton** (Перемикач) — до форми буде додано перемикач `RadioButton1`. Виділіть його там, клацніть правою кнопкою миші та виберіть із контекстного меню команду **Copy** (Копіювати). Виділіть всю форму, клацніть у ній правою кнопкою миші та виберіть із контекстного меню команду **Paste** (Вставити). В результаті з'явиться перемикач `RadioButton2`. Повторіть операцію вставки, щоб додати перемикачі `RadioButton3` та `RadioButton4`. Виділіть перемикач `RadioButton1`. Знайдіть його атрибут **Text** (Текст)

і змініть його значення на Ліва передня. Для інших перемикачів задайте такі значення атрибута **Text** (Текст): RadioButton2 — Ліва задня, RadioButton3 — Права передня, RadioButton4 — Права задня.

Прапорець

Як і перемикачі, прапорці застосовують для того, щоб користувач міг зробити вибір. Він може установити як один, так і кілька прапорців у групі. На вкладці **Toolbox** (Панель інструментів) двічі клацніть елемент керування **CheckBox** (Прапорець). До форми буде додано прапорець CheckBox1. Відкрийте для нього вікно **Properties** (Атрибути) та змініть значення атрибута **Text** (Текст) на Термінова робота.

Поле зі списком

Цей елемент керування дає користувачу можливість вибрати один текстовий рядок із кількох запропонованих. Обраний рядок відображується в полі. На вкладці **Toolbox** (Панель інструментів) двічі клацніть елемент керування **ComboBox** (Поле зі списком). До форми буде додано елемент керування ComboBox1. Відкрийте вікно **Properties** (Атрибути) та змініть значення атрибута **Text** (Текст) поля зі списком на Виберіть розмір шини. Знайдіть атрибут **Items** (Елементи). Клацніть слово Collection (Набір). Праворуч буде виведено кнопку з трьома крапками. Якщо ви клацнете її, то відкриється вікно, в яке можна ввести елементи, що міститимуться у списку. Щоб перейти на новий рядок, після введення кожного елемента натискайте клавішу **Enter.** Введіть такі елементи: 12, 13, 14 та 15 — це розміри шини в дюймах. Щоб закрити вікно, клацніть кнопку **OK.**

Кнопка

Додайте до форми кнопку та присвойте її атрибуту **Text** (Текст) значення Отримати шину. Зробіть фоновий колір кнопки червоним. Знайдіть атрибут **Font** (Шрифт) і клацніть його, щоб з'явилася кнопка з трьома крапками. Клацнувши цю кнопку, виведіть на екран палітру **Font** (Шрифт), на якій змініть стиль шрифту на жирний. Щоб закрити її, клацніть кнопку **OK.**

Побудуйте проект та запустіть його. Поклацайте різні елементи керування, які ви додали. Що сталося, коли ви клацнули у полі зі списком, обрали один із перемикачів, інший перемикач, установили прапорець? Тепер уведіть текст до текстового поля поруч із написом `Введіть номер ліцензії`. Зверніть увагу на те, що інше текстове поле зафарбоване сірим кольором, і до нього не можна вводити текст. Щоб закрити програму, клацніть кнопку X у правому верхньому куті її вікна.

Ви помітили, що вже створили «форму спущеної шини» і водночас вивчили кілька різновидів елементів керування та їхніх атрибутів? Щоб створити справжню «працюючу» форму, потрібно ще додати код Visual Basic .NET до форми та її елементів керування.

Ознайомтеся з атрибутами кожного з елементів керування, які ви додали до форми. Можна змінити їхнє розташування, щоб форма стала охайнішою і нею було легше користуватися. Ви можете також додати написи, які повідомлятимуть, для чого використовуються ті чи інші елементи керування, або міститимуть інструкції щодо змісту інформації, яка вводиться. Як ви вважаєте, де на формі потрібно розташовувати елементи керування? Які елементи слід розмістити зверху, а які — знизу?

Деякі атрибути є спільними для форм, більшості елементів керування та інших об'єктів. Серед них найчастіше використовують такі:

- **BackColor** (Колір фону);
- **Enabled** (Активовано);
- **ForeColor** (Колір написів на елементах керування);
- **Location** (Розташування);
- **Name** (Ім'я);
- **Size** (Розмір): **Height** (Висота), **Width** (Ширина);
- **Text** (Текст);
- **Visible** (Видимий).

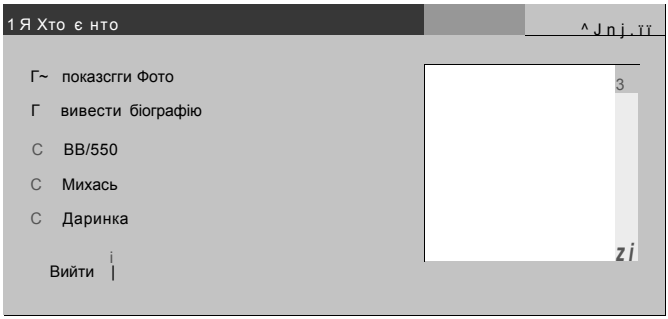
Вправа 4.1. Наші герої

Хочете дізнатися більше про Михася, Даринку та ВВ? Розробіть програму, яка виводитиме на екран їхні фотографії та біогра-

фії. На цьому етапі ви маєте довести, що здатні створити форму та настроїти атрибути об'єктів, які в ній містяться, за допомогою вікна **Properties** (Атрибути). Щоб дізнатися більше про своїх супутників, вам потрібно правильно встановити відповідність елементів керування наданому коду.

Відкрийте Visual Studio .NET, виберіть команду **Open Project** (Відкрити проект) і перейдіть до файлу **4.who_is_who.sln** із папки **Вправа_4.1** (його також можна знайти за допомогою Провідника Windows). Після цього двічі клацніть ім'я файлу — і проект відкриється.

Додаючи до форми об'єкти із вкладки **Toolbox** (Панель інструментів), надайте формі приблизно такого вигляду, як на рисунку. Для цього ви змінюватимете атрибути форми та елементів керування у вікні **Properties** (Атрибути).



1. Виділіть форму, клацнувши її один раз, а потім у вікні **Properties** (Атрибути) змініть її назву на `Хто є хто`.
2. Додайте до форми два прапорці. Залишіть для них назви, задані стандартно: `CheckBox1` і `CheckBox2`. Змініть значення їхніх атрибутів **Text** (Текст) на `показати фото` та `вивести біографію` відповідно.
3. Додайте до форми групу із трьох перемикачів. Змініть значення їхніх атрибутів **Text** (Текст) на `ВВ/550`, `Михась`, `Даринка`. Залишіть стандартні назви перемикачів: `RadioButton1`—`RadioButton3`.
4. Додайте до форми поле зображення `PictureBox1`. Скориставшись вікном **Properties** (Атрибути), задайте для нього

висоту 160 та ширину 140 пікселів. Щоб знайти атрибути висоти й ширини, клацніть значок +, що розміщений поруч з атрибутом **Size** (Розмір).

5. Додайте до форми текстове поле `TextBox1`. Ви маєте змінити деякі з його атрибутів. Порядок внесення змін є важливим! Вам потрібно, щоб текстове поле було досить великим для того, аби в ньому відображалося кілька рядків біографії. У Visual Basic не можна настроювати висоту текстового поля, доки ви не присвоїте атрибуту **Multiline** (Багаторядкове) значення `True`: якщо поле буде однорядковим, то для чого робити його високим? Надавши значення `True` атрибуту **Multiline** (Багаторядкове), встановіть висоту поля 160 пікселів. Задайте для атрибута **Scrollbars** (Смуги прокрутки) значення `Vertical`. Це потрібно робити у тому випадку, коли рядків у біографії буде більше, ніж вміщуватиме поле.
6. Додайте до форми кнопку `Button1`. Задайте для її атрибута **Text** (Текст) значення `Вийти`.

Створивши форму, побудуйте проект. Якщо не буде виявлено помилок, виберіть команду **Debug • Start Debugging,** щоб запустити програму на виконання.

Події у Windows-програмах

Події активізуються діями користувачів. Наприклад, коли користувач клацає кнопку, вибирає елемент у полі зі списком або змінює текст у полі, відбувається *подія*. Вона також відбувається, коли користувач клацає один раз або двічі елемент керування чи наводить на нього або знімає з нього покажчик миші. Виникнення кожної з цих подій приводить до виконання зв'язаного з нею VB-коду, який називають обробником події. Він «змушує» програму виконувати певні дії. Пригадуєте, які дії ви виконували, коли створювали програму HelloWorld? Я вам у цьому допоможу. У вікні, яке ви відкрили, двічі клацнувши кнопку, було введено код для обробки події `Button1_Click`. Цю подію використовують стандартно для кнопки `Button1`. Певна стандартна подія пов'язана з кожним елементом керування. Якщо у режимі розробки двічі

клацнути елемент керування, буде відкрито вікно коду для такої події. Перелічимо події, що задані стандартно для основних елементів керування:

- для кнопки — `Click`;
- для текстового поля — `TextChanged`;
- для перемикача — `CheckedChanged`;
- для прапорця — `CheckedChanged`;
- для поля зі списком — `SelectedIndexChanged`.

Щоб зрозуміти процес створення обробників подій, додайте код до подій, що задані стандартно для елементів керування форми спущеної шини. Цей код забезпечуватиме відображення вікна повідомлення, в якому зазначатиметься тип елемента керування, що спричинив подію. У режимі розробки двічі клацніть кожний з елементів керування, що були додані до форми. При цьому відкриватимуться вікна коду для подій, що задані стандартно. Вставте такий рядок коду до кожного обробника події:

```
MessageBox.Show(sender.GetType.Name)
```

Побудуйте проект та запустіть його. Змініть текст у полі та виберіть елементи зі списку. Для кожного елемента керування активуватиметься подія, задана стандартно. При цьому відкриватиметься вікно повідомлення, у якому зазначено тип елемента керування, що спричинив подію. Якби форма спущеної шини була частиною справжньої програми, ви б додавали код до обробників подій, заданих стандартно, щоб забезпечити виконання ними певних корисних функцій. Наприклад, якби користувач клацав кнопку **Отримати шини,** у текстовому полі виводився би список наявних шин.

Виведення повідомлень

Ви часто стикатиметесь із необхідністю забезпечити діалог між вашою програмою та користувачем. Можливо, вам знадобиться лише надіслати користувачеві повідомлення, а потім переконатися, що він його отримав. Такий різновид діалогу настільки поширений, що в середовищі Visual Studio .NET для нього

передбачено спеціальний об'єкт — **MessageBox** (Вікно повідомлення).

У першій програмі ви написали рядок коду, який забезпечував відображення вікна повідомлення:

```
MessageBox.Show("Hello, World")
```

Вікно повідомлення — це вбудований у VB .NET різновид форми. Щоб використати вікно повідомлення, вам не потрібно додавати його до свого проекту. Для цього потрібно лише викликати метод `Show ()` об'єкта `MessageBox` та вказати текст, який має міститися у вікні повідомлення. Синтаксис оператора виведення вікна повідомлення є таким:

```
MessageBox.Show("Текст, що виводитиметься")
```

Зауважте, що повідомлення, яке ви хочете вивести на екран, потрібно взяти в лапки.

Елементи керування форм мають не лише події, що задані стандартно. Для більшості з них визначено десятки подій. Наприклад, для текстового поля є такі події:

- `TextChanged` (задана стандартно);
- `Click`;
- `DoubleClick`;
- `MouseEnter`;
- `MouseLeave`.

Щоб побачити всі події для елемента керування, виділіть та клацніть його правою кнопкою миші. У контекстному меню виберіть команду **View Code** (Переглянути код). Відкриється вікно коду, у верхній частині якого ви побачите два списки. У списку праворуч містяться всі події для елемента керування. Події, що виділені напівжирним шрифтом, супроводжуються кодом.

— Справді, створювати форми цікаво! — сказав Михась, а Даринка підтвердила його слова, кивнувши головою.

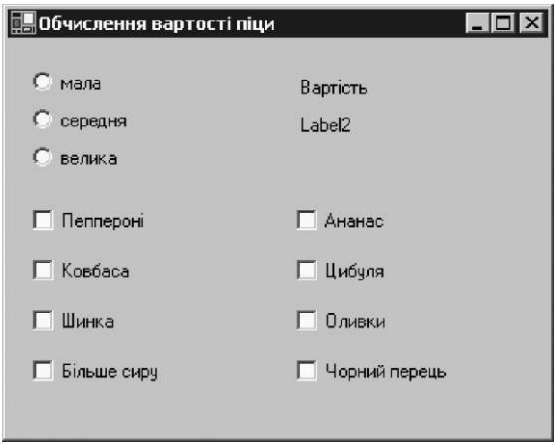
— Мені це також до вподоби, але давайте завершувати сьогоднішню роботу, — вирішив ВВ. — Ще одна-дві вправи, і переходимо до ігор. До речі, у мене є кілька нових ігор, які ніхто не знає у вашому часі. Але є і прикра новина: домашній робот не зумів завантажити з мережі програму приготування піци!

Тепер необхідно замовити піцу в найближчій піцерії, а для цього нам потрібно буде створити форму, що допоможе визначати вартість піци. Переконаний, що ви зможете її написати, оскільки вмієте розробляти форму, а код уже є.

Вправа 4.2. Готуємося до вечері

Створіть форму, яка відображатиме вартість піци. Файли, які потрібні для виконання цієї вправи, зокрема файли з кодом VB .NET, містяться в папці **Вправа_4.2.**

Отже, знайдіть у папці **Вправа_4.2** файл **PizzaPrice.sln** та, двічі клацнувши, відкрийте його у Visual Studio. Додайте об'єкти із вкладки **Toolbox** (Панель інструментів) до форми, щоб вона набула приблизно такого вигляду, як зображено на рисунку.



Для цього вам потрібні три перемикачі, два написи та вісім прапорців. Не забудьте змінити назву форми. Програма змінюватиме текст напису `Label2`.

У піцерії звичайна піца коштує 5 кредитів, за кожен додатковий м'ясний або овочевий інгредієнт покупець має доплачувати 1 кредит або 0,50 кредиту відповідно. Піца середніх розмірів коштує на 25 % дорожче, ніж маленька, а велика — на 50 %.

Створивши форму, побудуйте проект. Якщо під час побудови не виникло помилок, виберіть команду **Debug • Start Debugging.**

Додаткове завдання

Зробіть так, щоб значення вартості піци подавалося у форматі, який нагадує гроші. У Visual Basic .NET для цього використовується спеціальна функція. Передайте значення змінної, що зберігає вартість піци, до функції `FormatCurrency`. Це можна зробити приблизно так:

```
FormatCurrency(моЯЗмінна)
```

Результат виводитиметься у грошовому форматі.

Додайте також до форми зображення піци.

Завдання 4. Дитяча забавка

Створіть класичну гру в хрестики-нулики. Код для програми міститься в папці **Завдання_4,** але форми у цій програмі немає. Тому вам потрібно «намалювати» її, але так, щоб вона відповідала наявному коду.

Скопіюйте папку проекту **Завдання_4,** в якій мають міститися дві підпапки **(bin, obj)** та сім файлів, на жорсткий диск свого комп'ютера.

Відкрийте Visual Studio .NET, виберіть команду **Open Project** (Відкрити проект) і перейдіть до файлу **TicTacToe.sln.** Цей файл ви також можете знайти за допомогою Провідника Windows, двічі клацнути його, — і проект відкриється у середовищі Visual Studio .NET. Додаючи об'єкти із вкладки **Toolbox** (Панель інструментів), надайте формі приблизно такого вигляду, як показано на рисунку, наведеному нижче. При цьому у вікні **Properties** (Атрибути) змініть атрибути форми та елементів керування.

1. Виділіть форму, клацнувши її один раз, і у вікні **Properties** (Атрибути) змініть її назву на `Хрестики-нулики`.
2. Створіть дев'ять написів `Label1-Label9`. Розташуйте їх так, щоб напис `Label1` розміщувався в лівому верхньому куті, а напис `Label9` — у правому нижньому. Ви

зекономите час, якщо створите один напис, настроїте його атрибути, а потім скопіюєте цей напис ще у вісім різних місць.

Для всіх написів потрібно встановити такі значення атрибутів:

`Height`	48
`Width`	48
`Font`	Microsoft Sans Serif Bold 36
`BackColor`	Можна залишити заданий стандартно
`Text`	Можна залишити заданий стандартно

3. Під дев'ятим написом створіть ще один, десятий, із іменем `Label10` та задайте його атрибути:

`Height`	25
`Width`	175
`Font`	Microsoft Sans Serif Bold 16
`BackColor`	Можна залишити заданий стандартно
`Text`	Можна залишити заданий стандартно

4. Створіть ще три написи `Label11-Label13`. Задайте для них тексти `Виграли хрестики`, `Виграли нулики` та `Нічия`, як показано на рисунку.
5. Створіть три текстових поля `TextBox1-TextBox3` і розташуйте їх поруч із написами, створеними на попередньому кроці.

6. Створіть кнопку `Button1`. Для її атрибута **Text** (Текст) задайте значення `Грати знову`.
7. Створіть кнопку `Button2`, як показано на рисунку. Для її атрибута **Text** (Текст) задайте значення `Завершити`.

Створивши форму, побудуйте проект. Якщо під час побудови не виникло помилок, виконайте команду **Debug •Start Debugging.**

Додаткове завдання

Змініть фоновий колір форми, шрифт усіх кнопок і текстових полів. Використайте шрифт, який легше читається (наприклад, Microsoft Sans Serif Bold 10).

Чи можна водночас змінити шрифти зразу всіх елементів, якщо спершу виділити їх (для виділення кількох елементів слід клацнути кожний із них, утримуючи клавішу **Ctrl)?**
Створіть текстові поля для імен гравців.

— Отже, ви непогано справилися зі своїм завданням. Я думаю, що тепер ми можемо спокійно перейти до ігор, але... — ще не встиг ВВ договорити, як Михась із Даринкою разом завершили його думку:

— Але спочатку потрібно дати відповіді на тестові запитання, щоб отримати доступ до цих ігор.

Кодовий замок

1. Що використовують у формі для керування програмою?
   а) елементи керування;
   б) об'єкти;
   в) і те й інше.
2. Що є спільного у форми та її елементів керування?
   а) глибина;
   б) атрибути;
   в) дочірні вікна.

3. Який елемент керування використовують для отримання даних?

а) текстове поле;

б) напис;

в) кнопку.

4. Що ініціює подія у Visual Basic?

а) змінення значень атрибутів текстових полів;

б) введення даних;

в) виконання коду.

5. Кількість подій для різних елементів керування:

а) однакова;

б) різна;

в) залежить від програми.

6. Чи можна розтягувати текстові поля у висоту?

а) ні;

б) так;

в) так, але лише в тому випадку, коли встановлено прапорець **Multiline** (Багаторядкове).

7. Для чого призначена функція `FormatCurrency (x)` ?

а) перевіряє, чи можна перевести значення змінної `x` у грошовий формат;

б) переводить значення змінної `x` у грошовий формат;

в) перевіряє, чи має значення змінної `x` грошовий формат.

8. Якого атрибута не має кнопка?

а) **Name;**

б) **Items;**

в) **Text.**

День 5

Варіативність — запорука функціональності

Ще з вечора Михась із Даринкою почали просити ВВ пояснити їм код, бо їхнє перебування тут не може тривати довго. Крім того, батько ВВ казав, що вже мав розмову з часовими операторами щодо зворотної подорожі.

— Але ж він сказав, що через надмірне навантаження модулів вам доведеться ще кілька днів почекати! — зауважив ВВ.

— Адже «кілька днів» — це недовго, — вимовила Даринка. — Давай швидше вчитись, а то ми не встигнемо навіть познайомитися з вашим містом!

— Добре-добре, завтра після обіду підемо на прогулянку містом, — пообіцяв ВВ.

І ось «завтра» настало. Відразу після сніданку всі знову зібралися в кімнаті ВВ.

Код, що керує виглядом форми

— Створювати форми і настроювати елементи керування за допомогою вікна **Properties** (Атрибути) цікаво, та це лише початок, — продовжив учорашню розповідь ВВ. — Це вікно легко використовувати, але його можливості не надто широкі.

— А якщо користувач хоче змінити фоновий колір форми? Чи розмір або стиль шрифту? Звідки програма «дізнається», що користувач установив прапорець або клацнув перемикач? — запитала Даринка.

— Оскільки користувачі не мають доступу до вікна атрибутів під час виконання програми, як вони можуть робити такі речі? — поцікавився Михась.

— Відповіді на ці запитання дає код Visual Basic .NET. Ви можете робити такі речі й багато інших за допомогою кодування. Вам необхідно написати код, який зчитуватиме значення атрибутів елементів керування. Щоб задати значення атрибутів, також пишеться код. Хіба не просто? Для мене саме з цього й починалося програмування.

Вікно **Properties** (Атрибути) використовують для того, щоб задавати вихідні значення атрибутів форми та елементів керування під час розробки форми. Код застосовують для змінення атрибутів під час виконання програми.

— А де розміщується весь код? — поцікавився Михась, який вже чув про кілька середовищ програмування, де код займав чільне місце.

— На сторінці коду форми! — відповів ВВ. — У більшості випадків ви помістите його до обробників подій елементів керування. Пам'ятаєте, що відбувається у програмі після клацання мишею командної кнопки? Коли користувач виконує певну дію, наприклад, клацає кнопку, керування передається коду, який ви написали для обробника відповідної події. Цей код може зчитувати або задавати значення атрибутів форми чи елементів керування. Додаючи код до обробників подій, ви робите перші кроки на шляху перетворення форми на справжню робочу програму. Пам'ятаєте форму спущеної шини, яку ви розробили? За її допомогою не можна буде виконати жодної дії, доки ви не додасте код.

Щоб змінити код обробника події, яка пов'язана з елементом керування стандартно, виберіть у формі елемент керування та двічі клацніть його. Відкриється вікно, в якому вже міститься готовий до редагування код. Ви можете вибрати обробник іншої події зі списку, що відображається у верхній правій частині вікна коду.

Зчитування значень атрибутів

Отже, сьогодні наше перше завдання — навчитися зчитувати значення атрибутів за допомогою коду Visual Basic .NET — проголосив ВВ. — Спочатку розглянемо кілька простих атрибутів, значення яких зазвичай задають у вікні **Properties** (Атрибути). Розпочинаємо!

Створіть нову Windows-програму, яка називатиметься ReadProperties. Відкрийте вікно **Toolbox** (Панель інструментів) і додайте до форми `Form1` кнопку. Потім відкрийте вікно **Properties** (Атрибути) та змініть значення атрибута `Text` кнопки `Button1` на `Читати текст`.

Двічі клацніть кнопку, щоб внести зміни до коду обробника стандартної події. Введіть такий рядок коду:

```
MessageBox.Show(Button1.Text)
```

Побудуйте та запустіть проект. Клацніть кнопку **Читати текст.** Відкриється вікно повідомлення з текстом `Читати текст`, який є значенням атрибута `Text` кнопки `Button1`. Отже, код зчитує значення атрибута `Text` кнопки `Button1`, а після цього виводить повідомлення, яке містить це значення. Чудово, чи не так?

Візьмемо інший приклад. Додайте до форми ще одну кнопку. У вікні **Properties** (Атрибути) змініть значення атрибута `Text` кнопки `Button2` на `Розмір форми`. Двічі клацніть кнопку `Button2`, щоб внести зміни до коду обробника стандартної події. Введіть такий рядок коду:

```
MessageBox.Show(Form1.ActiveForm.Height & ","
& Form1.ActiveForm.Width)
```

Побудуйте та запустіть проект. Клацніть кнопку **Розмір форми.** У вікні повідомлень відобразяться два числа, наприклад 300, 300. Ваш код тільки-но показав значення атрибутів `Height` (Ширина) та `Width` (Висота) форми `Form1`, розділивши їх комами. Знову чудово! Та ще й просто!

— Які ще атрибути може зчитувати код Visual Basic .NET? — запитав Михась.

— Фактично, всі атрибути форми та елементів керування — все те, що зібрано у вікні **Properties** (Атрибути). Зокрема, можна зчитувати значення таких атрибутів, як `Height` (Ширина) та `Width` (Висота), `BackColor` (Колір фону), `ForeColor` (Колір написів на елементах керування), `X and Y Locations` (Координати X та Y). Проте значення більшості атрибутів, що відображаються у вікні **Properties** (Атрибути), задаються під час першої розробки форми та не змінюються, коли виконується

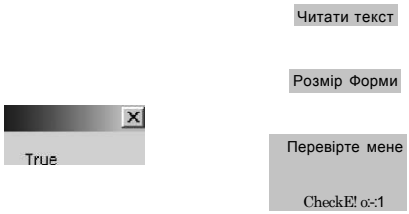
програма. Тож зчитування значень атрибутів не завжди є корисним або цікавим, — відповів ВВ.

Значення атрибутів елементів керування доцільно зчитувати тоді, коли користувачі мають змогу їх змінювати під час роботи із програмою. Наприклад, вони мають змогу змінити значення атрибута `Text` поля або `Checked` прапорця. Зчитуючи значення атрибутів, програма отримує інформацію від користувачів.

Розглянемо приклад. Напишемо код, що визначатиме, чи встановлено прапорець. Відкрийте вікно **Toolbox** (Панель інструментів) та додайте до форми `Form1` прапорець і третю кнопку, а потім, відкривши вікно **Properties** (Атрибути), змініть значення атрибута `Text` кнопки `Button3` на `Перевірте мене`. Двічі клацніть кнопку `Button3`, щоб внести зміни до коду обробника події. Введіть такий рядок коду:

```
MessageBox.Show(CheckBox1.Checked)
```

Побудуйте та запустіть проект. Клацніть кнопку **Перевірте мене.** Буде виведено повідомлення `False`. Установіть прапорець і знову клацніть ту саму кнопку. Яким є повідомлення цього разу? Тепер воно має містити слово `True`. Кожного разу, коли ви клацаєте кнопку, вікно повідомлення відображає поточне значення атрибута `Checked` прапорця.



А тепер подивимось, як зчитувати та виводити на екран слова, які користувач вводить у текстове поле.

Додайте до форми `Form1` текстове поле. Відкрийте вікно **Properties** (Атрибути) цього поля та переконайтеся, що атрибуту

ReadOnly надане значення False. Атрибуту Text надайте значення Введіть текст. Додайте четверту кнопку до форми. Змініть значення атрибута Text кнопки Button4 на Читати текст. Клацніть кнопку Button4, щоб внести зміни до коду обробника стандартної події. Введіть такий рядок коду:

```
MessageBox.Show(TextBox1.Text)
```

Побудуйте проект та запустіть його. Двічі клацніть мишею в полі **TextBox1** і натисніть клавішу **Backspace,** щоб видалити текст. Введіть будь-який новий текст. Клацніть кнопку **Читати текст.** Після цього відкриється вікно повідомлення. Яким є зміст повідомлення? Змініть текст у полі та знову клацніть кнопку **Читати текст.** Що сталося?

Наведені вище приклади показують, як використовувати код Visual Basic .NET для зчитування значень атрибутів елементів керування. Крім цього, код Visual Basic .NET застосовують для того, щоб задавати значення атрибутів.

Надання атрибутам значень

Добре, що можна зчитувати значення атрибутів елементів керування в коді Visual Basic .NET. У такий спосіб у коді можна визначити, які значення атрибутів установлюють користувачі, коли працюють із програмою. Залежно від зчитаних значень, програма може виконувати різний код, а отже, і різні дії.

— А чи може бути навпаки? — запитав Михась. — Тобто чи може програма сама змінювати значення атрибутів елементів керування? Наприклад, мені потрібно, щоб прапорець **Термінова робота** був установлений завжди, коли користувач уперше запускає програму; щоб у текстовому полі з'являлися всі наявні розміри шин, коли користувач клацає кнопку **Отримати шини;** щоб список розмірів шин формувався автоматично після того, як користувач уведе номер машини.

— Звичайно, за допомогою коду Visual Basic .NET ти можеш зробити все це! — відповів ВВ. — Задавати значення атрибутів елементів керування так само легко, як і зчитувати їх. Щоб задати значення атрибута в коді Visual Basic .NET, застосовують оператор присвоєння. Пам'ятаєте, як ми обговорюва-

ли цей оператор? Він призначений для надання певного значення певній змінній. Код оператора присвоєння схожий на звичайне рівняння:

```
TextBox1.Text = "Правила кодування"
```

Спочатку обчислюється значення виразу, який записано праворуч від символу =, а потім це значення надається змінній, що вказана ліворуч від цього символу.

Ось кілька прикладів того, як задають значення атрибутів у коді Visual Basic .NET. У першому прикладі задається значення атрибута `Text` текстового поля. У такий спосіб легко надавати користувачеві інформацію, адже текстове поле можна використовувати замість вікна повідомлення.

Створіть нову Windows-програму та назвіть її SetProperties. Додайте два текстових поля та кнопку до форми `Form1`. Щоб надати атрибуту `Text` кнопки `Button1` значення `Повторити текст`, використайте вікно **Properties** (Атрибути). Атрибути `Text` полів `TextBox1` і `TextBox2` не повинні мати жодних значень. Видаліть значення цих атрибутів, клацаючи двічі кожне з них і натискаючи клавішу **Backspace.** Двічі клацніть кнопку `Button1`, щоб змінити код обробника події клацання кнопки. Додайте такий рядок коду:

```
TextBox2.Text = TextBox1.Text
```

Побудуйте та запустіть проект. Уведіть текст у поле `TextBox1` і клацніть кнопку **Повторити текст.** Що сталося? Як бачите, текст, що міститься у полі `TextBox1`, «скопійовано» у поле `TextBox2`. Атрибути `Text` полів `TextBox1` і `TextBox2` набули однакових значень. Оператором присвоєння зчитано значення атрибута `Text` поля `TextBox1`, а потім це значення надане атрибуту `Text` поля `TextBox2`. Дуже зручно!

Створимо код, завдяки якому один прапорець автоматично встановлюватиметься, а інший — автоматично зніматиметься, коли користувач запускатиме програму. Додамо до форми `Form1` прапорець і за допомогою вікна **Properties** (Атрибути) надамо його атрибуту `Text` значення `Термінова робота`. Додамо до форми `Form1` другий прапорець та надамо його атрибуту `Text` значення `Будь-коли`. Якщо двічі клацнути форму

Form1 (не прапорці CheckBox1 або CheckBox2), відкриється для редагування код обробника події Form1_Load. Це подія завантаження форми. Код її обробника виконується тоді, коли форма створюється та вперше відображається на екрані. Додайте до обробника події Form1_Load такі два рядки коду:

```
CheckBox1.Checked = True
CheckBox2.Checked = False
```

Побудуйте та запустіть проект. Який прапорець установлено? Під час завантаження форми атрибутам Checked обох прапорців було автоматично надано відповідні значення.

Подію завантаження форми часто використовують для ініціалізації програми, тобто для задання певних значень до того, як користувач побачить форму. Наприклад, щоб задати розмір форми та її фоновий колір, а також установити певні перемикачі, ви могли б додати код до обробника події завантаження форми.

Наскільки мені відомо, найпоширеніший розмір шин — 14 дюймів. Отже, давайте зробимо так, щоб у полі зі списком програма автоматично обирала значення 14. По-перше, додамо поле зі списком до форми Form1. За допомогою вікна **Properties** (Атрибути) задамо для атрибута Items поля зі списком значення 12, 13, 14, 15 та 16. Не забувайте натискати клавішу **Enter,** щоб значення кожного елемента списку вводити з нового рядка. Двічі клацніть форму Form1 (не в полі зі списком), щоб змінити код обробника події завантаження форми. Додайте такий рядок коду:

```
ComboBox1.SelectedItem = "14"
```

Побудуйте проект та запустіть його на виконання. Який елемент обрано в полі зі списком?

Раптом Михась побачив за вікном веселку, що нагадала йому та Даринці їхній час, і вони на мить засумували за своєю домівкою та за батьками. Щоб хоч якось розвеселити друзів, ВВ запропонував їм створити художню програму, яка б імітувала веселку.

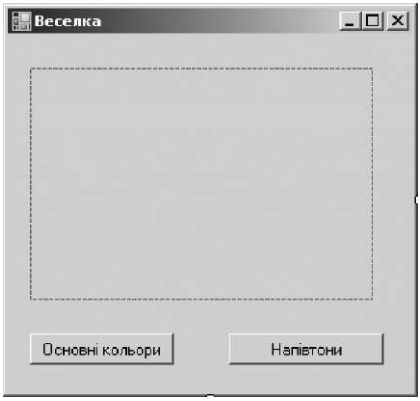
— Доведіть своє уміння оперувати атрибутами, використовуючи справжній код Visual Basic, а не вікно **Properties** (Атрибути), — спонукав їх до праці ВВ.

Вправа 5.1. Сім кольорів веселки

Розробіть форму, в якій відображатиметься барвиста веселка. Змініть існуючий код, щоб веселка стала кольоровою, та додайте його до обробників подій, які пов'язані з елементами керування. Файли, необхідні для виконання цієї вправи, містяться в папці **Вправа_5.1.**

Запустіть Visual Studio .NET, виберіть команду **Open Project** (Відкрити проект) і перейдіть до файлу **Rainbow.sln.** Двічі клацнувши його, відкрийте проект.

Форма матиме такий вигляд, як показано на рисунку.



Двічі клацніть кнопку, щоб переглянути код, який уже створено для вас. Ви побачите оператори, призначені для малювання семи смуг веселки. Усі смуги матимуть чорний (Black) колір. Ваше завдання полягає в тому, щоб змінити їхні кольори на сім традиційних: червоний (Red), оранжевий (Orange), жовтий (Yellow), зелений (Green), блакитний (Blue), синій (Indigo) та фіолетовий (Violet). Ви можете, наприклад, замінити один з операторів

```
MyPen.Color = System.Drawing.Color.Black
```

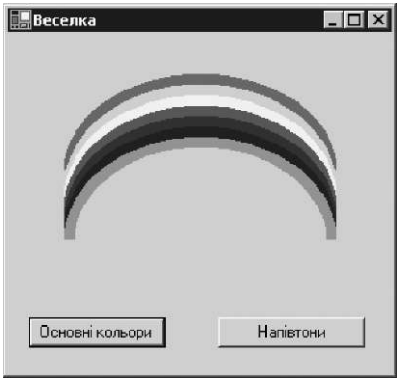
таким:

```
MyPen.Color = System.Drawing.Color.Red
```

Тільки-но після слова `Color` ви поставите крапку, як буде виведено список усіх доступних кольорів. Для того щоб обрати елемент списку, достатньо ввести одну чи дві перших літери кольору.

Двічі клацніть кнопку `Напівтони`, щоб побачити код малювання веселки в напівтонах. Так само оберіть сім кольорів на свій смак замість чорного.

Створивши форму, побудуйте проект. Якщо при цьому не виникло помилок, виберіть команду **Debug • Start Debugging,** щоб запустити програму.



Додаткове завдання

Змініть аргументи методів DrawArc для додаткових кольорів, щоб отримати інші цікаві ефекти (наприклад, можна зменшити чи збільшити дуги).

Засіб автовведення імен атрибутів і методів

— Тепер ви вже знаєте, як писати код Visual Basic .NET, що зчитує та встановлює значення атрибутів форм і елементів керування, — зробив висновки ВВ. — Я навіть здогадуюсь, яким буде ваше наступне запитання: як я керую всіма атрибутами

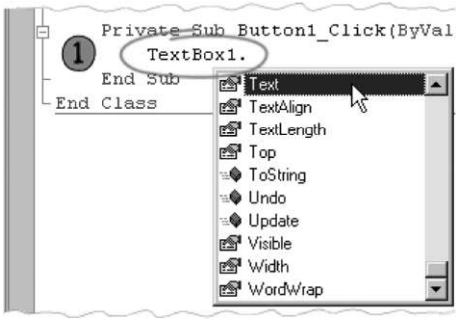
за такої їх кількості в багатьох формах? Не дуже зручно весь час переключатися то на вікно коду, то на вікно **Properties** (Атрибути), коли потрібно дізнатися, які атрибути належать певному елементу керування.

— А існує простіший спосіб? — запитала Даринка.

— Такий спосіб є і зветься він IntelliSense. Засіб IntelliSense — частина Visual Studio. Використовуючи його під час роботи з вікном коду, можна заощадити багато часу. IntelliSense знає, з яким елементом керування ви працюєте та які він має атрибути. Завдяки цьому засобу можна вивести список атрибутів та обрати будь-який з них. Користуючись клавішами керування курсором, ви зможете переміщуватися списком, щоб знайти потрібний атрибут. Коли ви натискаєте клавішу **Tab,** атрибут додається до коду. Застосовуючи засіб IntelliSense, вам не доведеться запам'ятовувати всі атрибути всіх елементів керування. До речі, я завжди користуюся цим засобом.

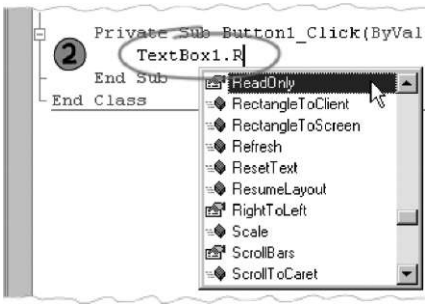
Перевіримо його! Створіть нову Windows-програму і назвіть її IntelliSense. Додайте текстове поле (стандартно йому буде надано ім'я `Textboxl`) та кнопку (`Buttonl`) до форми `Forml`. Двічі клацніть кнопку, щоб відкрити код обробника події її клацання.

1. Введіть `TextBoxl` та натисніть клавішу із символом крапки (.). Тільки-но ви введете крапку, IntelliSense виведе список атрибутів елемента керування `TextBoxl`.

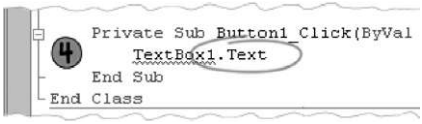


Цей список буде таким самим і для будь-якого іншого текстового поля, яке ви додасте до форми. Стандартно у списку IntelliSense обрано атрибут, що використовується найчастіше. У цьому випадку обрано атрибут `Text`.

2. Тепер уведіть літеру R. IntelliSense перейде до першого атрибута у списку, що починається з R. Коли ви вводите літери, цей засіб шукає атрибут, який найбільше відповідає введеному тексту.



3. За допомогою клавіш керування курсором перемістіться списком та знову знайдіть атрибут `Text`. Виберіть його.



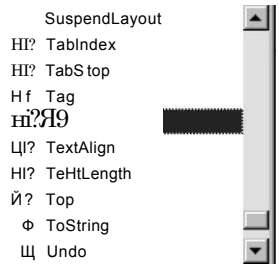
4. Натисніть клавішу **Tab**. Погляньте-но, що сталося! Слово `Text` вставлено в код після слова `TextBox1`. Тепер залишилося ввести оператор присвоєння значення (наприклад, `= "Михась")`, і код набуде такого вигляду:

```
TextBox1.Text = "Михась"
```

Легко та швидко, чи не так? Перевага засобу IntelliSense полягає також у тому, що він дає змогу скоротити кількість помилок.

Коли атрибут виділяється у списку IntelliSense, біля нього з'являється підказка з інформацією про атрибут, зокрема про те, чи призначений він лише для читання (тобто, чи не заборонено йому надавати значення). У підказці також повідомляється, значення якого типу може бути надане атрибуту (наприклад, текстовий рядок або ціле число).

Ви, мабуть, помітили, що поруч із кожним елементом у списку IntelliSense є піктограма. Піктограма атрибута містить зображення руки, що вказує на елемент у списку. Піктограма методу має вигляд пурпурового ромбу. (Згодом я поясню вам, що таке методи). Залежно від типу елемента керування, ви також можете побачити інші піктограми.



Ви звернули увагу, що для виведення списку IntelliSense вам довелося поставити крапку? У мові Visual Basic .NET використовується синтаксис, який називають крапковим позначенням. Після того як ви введете ім'я елемента керування, наприклад `TextBoxl`, вам потрібно ввести крапку, а потім — назву атрибута. Крапка зв'язує елемент керування або форму з їхніми атрибутами:

```
Ім'яФорми.АтрибутФорми
Ім'яЕлементаКерування.АтрибутЕлементаКерування
```

Ви вже бачили чимало прикладів імен елементів керування й атрибутів, але наведемо ще кілька:

```
TextBoxl.Text
CheckBoxl.Checked
Forml.ActiveForm.Height
```

Зауважте, що в останньому прикладі `ActiveForm` — це атрибут форми `Forml`, а `Height` — атрибут атрибута `ActiveForm`. Тому, щоб з'єднати атрибути разом, потрібно дві крапки. Засобу IntelliSense відомо, коли атрибут має свої атрибути. Якщо у вас виникли сумніви, введіть ще одну крапку, і IntelliSense відобразить список усіх наявних атрибутів. Відсутність списку вказує на те, що додаткових атрибутів немає. У цьому разі вам

доведеться видалити останню крапку. Список не з'явиться й тоді, коли ви неправильно введете ім'я елемента керування.

Поняття методу

Ви, мабуть, помітили, що деякі атрибути зі списку, який виводить IntelliSense, позначені пурпуровим ромбом. Насправді це не атрибути, а методи. Атрибути — це характеристики об'єктів. Наприклад, колір, кількість місць, вага та потужність мотору — характерні ознаки машини. Методи більше схожі на дії об'єкта, наприклад, повернути ліворуч та зупинитися — це дії машини. Через вікно **Properties** (Атрибути) не можна отримати доступ до методів, їх потрібно викликати в коді.
Більшість елементів керування мають методи `Hide` та `Show`. Якщо викликати метод `Hide`, елемент керування стає невидимим, хоча, як і раніше, міститься на формі. Якщо викликати метод `Show`, елемент керування відображається знову. Деякі методи «беруть на себе» виконання дій, які зазвичай виконують користувачі. Наприклад, у кнопки є метод `PerformClick`, за допомогою якого клацання кнопки здійснюється так, ніби це зробив користувач.

Деякі методи потребують, аби під час їхнього виклику для них ввели певну інформацію, яка може впливати і на те, як виконується метод, і на результати його виконання. Дані, що передаються методу, називають його аргументами. Метод може мати один або кілька аргументів, деякі з них можуть бути необов'язковими.

Так, аргументом методу `Show` вікна повідомлення є текстовий рядок. Цей аргумент містить текст, який відображатиметься у вікні повідомлення. Ви вже використовували метод `Show` вікна повідомлення, але тоді, мабуть, не усвідомлювали, що викликали метод із аргументом:

```
MessageBox.Show("Hello, World")
```

У наведеному коді текст `"Hello, World"` є аргументом методу `Show`. Аргумент визначає вміст вікна повідомлення. Методи можуть також повертати значення, що потім використовуватимуться в коді.

Виклик методів

Синтаксис виклику методу схожий на синтаксис встановлення або зчитування значення атрибута. Щоб з'єднати елемент керування форми з його методом, використовують крапкове позначення. Деякі методи, наприклад методи `Button.Hide` та `Button. Show`, не мають аргументів, а інші — мають. Методу `MessageBox.Show` аргумент потрібен задля того, щоб знати, яке повідомлення виводити. Отже, синтаксис виклику методу є таким:

```
Ім'яЕлементаКерування.Ім'яМетоду(аргумент1, аргумент2, ...)
```

Як бачите, основна синтаксична відмінність між викликом методу та зчитуванням (або встановленням) значення атрибута полягає у тому, що під час виклику методу замість знака рівності використовують дужки.

Тепер поглянемо на деякі з моїх улюблених методів і дізнаємося, як їх можна викликати з коду. Я покажу вам, як приховувати та відображати елементи керування, видаляти текст із текстових полів та пересувати курсор. Спочатку створіть нову Windows-програму, яка називатиметься Methods. Додайте дві кнопки та текстове поле до форми `Form1`. Надайте атрибуту `Text` кнопки `Button1` значення `Показати`, а атрибуту `Text` кнопки `Button2` — значення `Приховати`. До обробника події клацання кнопки `Button1` додайте такий рядок коду:

```
TextBox1.Show()
```

До обробника події клацання кнопки `Button2` додайте код:

```
TextBox1.Hide()
```

Побудуйте проект і запустіть його. Клацніть почергово кнопки **Показати** та **Приховати.** Кожного разу, коли ви клацаєте кнопки, то викликаєте метод `Hide` або `Show` текстового поля.

Додайте до форми `Form1` ще одну, третю, кнопку. Надайте атрибуту `Text` кнопки `Button3` значення `Скинути`. До обробника події клацання кнопки `Button3` додайте такі рядки коду:

```
TextBox1.ResetText()
TextBox1.Focus()
```

Метод `ResetText` видаляє текст із поля та надає його атрибуту `Text` порожній рядок. Метод `Focus` встановлює курсор у текстове поле.

Створіть проект і запустіть його. Введіть текст до текстового поля та клацніть кнопку **Скинути.** Текст буде видалено з поля, а курсор встановлено в ньому, тобто поле буде підготовлене до введення нового тексту.

— Час завершувати заняття, — сказав ВВ, закриваючи середовище Visual Studio в себе на комп'ютері. — Ви вже дізналися чимало нового. Тепер ви вмієте зчитувати та задавати значення атрибутів, а також викликати методи, використовуючи код Visual Basic .NET. Як ви бачили, більша частина коду припадає на обробники подій елементів керування. Щоб відкрити код обробника події, потрібно просто двічі клацнути елемент керування. Щоб зчитати або задати значення атрибута, чи викликати метод, введіть ім'я елемента керування, а потім крапку. Засіб IntelliSense надасть вам список атрибутів та методів, які доступні для цього елемента керування. Виберіть із них той, що вам потрібен, і натисніть клавішу **Tab,** щоб додати його до коду.

В нас є ще трохи часу для виконання кількох вправ. А коли надворі спаде спека і сонце спуститься нижче, ми погуляємо містом.

— Ура! — вигукнула Даринка.

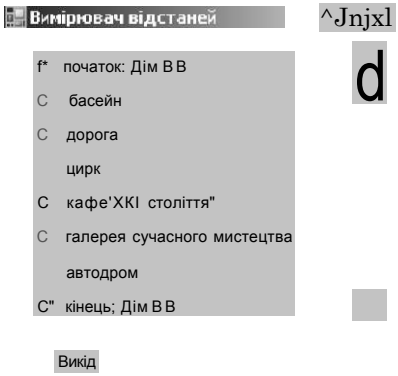
— Нарешті, — підтримав її Михась.

Оскільки ВВ, Михась і Даринка збираються йти до міста, вони вирішили оптимізувати свій маршрут і подивитися, яку відстань їм необхідно буде подолати, щоб відвідати всі визначні місця. Маєте бажання їм допомогти? Тоді відкрийте Visual Studio .NET і створіть новий проект TripMeter.

Вправа 5.2. Прилад для мандрівника

Розробіть форму для приладу, що визначатиме довжину пройденого під час мандрівки шляху. Додайте код до елементів форми, щоб зробити її інтерактивною. Файли, необхідні для виконання цієї вправи, містяться в папці **Вправа_5.2.**

Додаючи до форми об'єкти з вікна **Toolbox** (Панель інструментів), надайте їй такого вигляду, як на рисунку. Згідно з наведеними інструкціями змініть елементи у вікні коду.



1. Клацніть форму та спочатку змініть її назву на `Вимірювач відстаней`.
2. Створіть вісім перемикачів. Задайте ширину кожного перемикача 200 пікселів. (Ви можете використати копіювання та вставляння).
3. Вирівняйте перемикачі по вертикалі та розташуйте їх на однаковій відстані один від одного. Вам відомо, що на панелі інструментів є кнопки, які здійснюють це? Наводьте покажчик миші на кнопки панелі інструментів, доки не знайдете потрібну, потім клацніть її.
4. Установіть для перемикачів текст так, як це зображено на рисунку.
5. Створіть кнопку **Завершити.** Двічі клацніть її, щоб відкрити зв'язану з нею сторінку коду. Помістіть слово `End` у код обробника події клацання кнопки.
6. Розташуйте на формі вертикальну смугу прокручування. Для цього вам, можливо, доведеться прокрутити вікно **Toolbox** (Панель інструментів). Стандартно смуга прокручування починається з позиції 0 та завершується в позиції 100, тому ви отримаєте завеликий індикатор позиції.

Щоб зменшити його, змініть значення атрибута `Maximum` для смуги прокручування на 700.

7. Тепер можна писати код. Двічі клацніть перший перемикач. Помістіть до обробника події клацання кнопки такий рядок коду:

```
VScrollBar1.Value = 0
```

У такий спосіб ви переміщуєте індикатор у положення 0. Для кожного перемикача використовуйте такий самий рядок коду, тільки робіть значення на 100 більшим за попереднє.

Після того як ви створите форму, побудуйте проект. Якщо при цьому не будуть виявлені помилки, виберіть команду **Debug • Start Debugging,** щоб запустити програму.

ВВ вирішив зробити Даринці приємний подарунок, а тому він скачав із мережі кілька малюнків квітів і попросив у Михася, щоб той допоміг йому розробити форму, використовуючи яку, можна було б легко переглядати ці картинки. Спробуйте допомогти їм!

Завдання 5. Зробіть дівчинці приємне

Створіть форму, в якій демонструватимуться квіти (таку, як показано на рисунку). Напишіть код, що змінюватиме значення атрибутів форми. Файли із зображеннями квітів містяться в папці **Завдання_5.**

1. Надайте атрибуту `Visible` чотирьох полів картинок значення `False`.
2. Надайте атрибуту `Border` чотирьох полів картинок значення `Fixed3D`.
3. Надайте атрибуту `SizeMode` чотирьох полів картинок значення `AutoSize`.
4. Надайте атрибуту `Image` кожного поля ім'я файлу відповідного зображення.
5. Після цього перейдіть до сторінки коду. Над самим кодом є два списки, що розкриваються, — елементів керування та подій. У лівому виберіть елемент `labell` (припустимо, що напис `labell` містить текст `Гвоздика`).
6. У правому списку виберіть подію `MouseHover`.
7. В події `Labell.Mousehover` атрибуту `Visible` поля `PictureBoxl` надайте значення `True`, а атрибутам `Visible` трьох інших полів — значення `False`.
8. Зробіть те саме для решти трьох написів, відповідно змінюючи видимість зображень.
9. У результаті програма має показувати лише одне зображення в тому випадку, коли покажчик миші наведений на його опис.

Додаткове завдання

Модифікуйте програму так, щоб у разі вибору певної квітки автоматично змінювався фоновий колір форми.

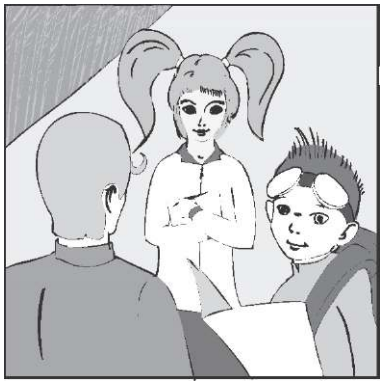
Нарешті обрано маршрут, яким друзі подорожуватимуть містом. Та виникла ще одна проблема: щоб вийти з будинку, потрібно відкрити кодовий замок.

Кодовий замок

1. З якою метою використовують код програми?
   а) для зчитування значень атрибутів елементів керування;
   б) для встановлення значень атрибутів елементів керування;
   в) і для того, і для іншого.

2. Який рядок коду забезпечує зчитування та виведення на екран значення атрибута `Checked` перемикача?
   а) `MessageBox.Show(RadioButton1.Checked);`
   б) `MessageBox.Show(Radio.Property);`
   в) `MessageBox.Show(RadioButton1.Selected).`
3. У який спосіб виконуються оператори присвоєння в мові Visual Basic?
   а) ліва частина присвоюється правій;
   б) права частина присвоюється лівій;
   в) будь-яким із вказаних способів, оскільки вони еквівалентні.
4. Який оператор установлює прапорець `CheckBoxl`?
   а) `CheckBoxl.Checked = Yes;`
   б) `CheckBoxl.Unchecked = False;`
   в) `CheckBoxl.Checked = True.`
5. Який із наведених рядків некоректний?
   а) `MessageBox.Show(TextBox1.Visible);`
   б) `TextBoxl.Width = l00;`
   в) `TextBoxl.Text = True.`
6. Коли виконується код обробника події `Forml_Load`?
   а) ніколи;
   б) у разі виникнення помилкових ситуацій;
   в) перед відкриттям форми `Forml`.
7. Для чого призначений засіб IntelliSense?
   а) для виправлення помилок у введеному коді;
   б) для прискорення набору команд;
   в) для прискорення виконання програми.
8. Які дії виконує метод `TextBox. Focus ()` ?
   а) шукає текст у текстовому полі;
   б) видаляє текст із текстового поля;
   в) встановлює курсор у текстове поле.

День 6
Змінні



Після будь-якої мандрівки, а тим паче мандрівки незнайомим і цікавим містом, страшенно хочеться спати. Але навчання — понад усе. Тому вранці, як завжди, друзі зібрались у кімнаті ВВ.

— У нас сьогодні складна тема, — попередив ВВ Михася та Даринку, коли ті вже сиділи в його кімнаті, готові до чергового уроку.

— А що ми вивчатимемо сьогодні? — запитала Даринка.

— Те, що лежить в основі будь-якого програмування, — відповів ВВ.

— Функції? — з виглядом експерта запитав Михась.

— Ні, без функцій можна обійтися, — заперечив ВВ.

— Цикли? — Михась був уже менш упевненим.

— І без них теж програма може існувати, — відповів ВВ.

— То що ж це?! — нетерпляче запитала Даринка.

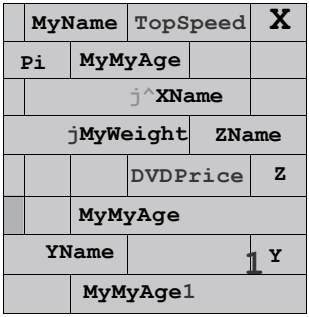
— Змінні. Сьогодні ми говоритимемо про основу будь-якого програмування — змінні, — урочисто сказав ВВ.

— Але ж ми писали програми без них! — ображено зауважила Даринка.

— Ні, вони були, просто ви про них тоді ще не знали, — відповів ВВ. — Пригадуєте атрибути? Це є ті самі змінні!

— Ну добре, починай, — погодилася Даринка.

— Спочатку я розповім вам про типи даних (рядки, цілі числа тощо). Коли ви задаєте значення атрибута форми, використовуючи код, то мусите переконатися, що воно правильного типу. Потім ми будемо обговорювати змінні та їхнє використання в коді. Змінні в коді застосовуються для зберігання різнотипних даних. Вони є «цеглинами», з яких складається код.



Присвоєння у Visual Basic

Минулого разу ми почали писати код Visual Basic .NET. Тепер ви вже знаєте, як за допомогою коду зчитувати та задавати значення атрибутів. Також ви маєте пам'ятати, що для встановлення значення атрибута застосовують оператор *присвоєння,* що надає певне значення певній змінній. Код оператора присвоєння подібний до простого рівняння:

```
TextBox1.Text = "Правила кодування"
```

Тепер поговоримо про правила використання операторів присвоювання. Насамперед потрібно знати, що спочатку обчислюється значення виразу, який записаний праворуч від символу «=». Потім це значення надають змінній, що розміщена ліворуч від символу «=». Це нібито зворотний порядок, але саме так працює програма. У наведеному вище рядку коду спочатку «обчислюється» рядок "Правила кодування", а потім атрибуту TextBoxl.Text надається обчислене значення.

Типи даних

Вам також необхідно знати, що ліва частина оператора присвоєння вимагає, щоб права частина мала певний *тип* (була текстом, числом, кольором тощо). Я розкажу вам, що маю на увазі, на прикладі рядка коду, який ми вже розглядали:

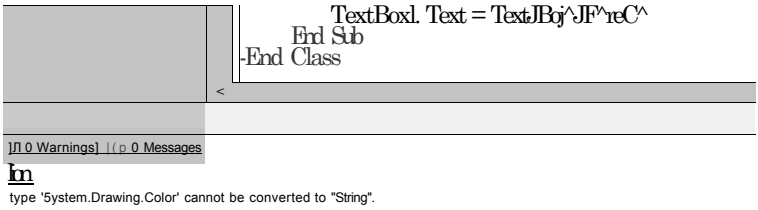
```
TextBox1.Text = "Основи правил кодування"
```

У лівій частині присвоєння — атрибут Text елемента керування TextBoxl, який може мати лише текстове значення, тому

в правій частині міститься текст. Якщо права частина оператора присвоєння буде іншого типу (не текстом), виникне помилка, коли ви закінчите вводити рядок чи коли програма спробує його виконати. Оскільки `"Основи правил кодування"` — це текст, помилки не виникає.

Компіляція наступного рядка не буде успішною, тому що ми намагаємося надати текстовому атрибуту `Text` значення атрибута `ForeColor`, яке є кольором:

```
TextBox1.Text = TextBox1.ForeColor 'виникає помилка
```



Наведений далі код коректний, оскільки права частина є цілим числом, а атрибуту `Height` можна присвоїти лише ціле число:

```
TextBoxl.Height = 200
```

А от такий код буде некоректним, оскільки права частина присвоєння є дробовим числом:

```
TextBoxl.Height = 200.5
```

Пам'ятайте: коли використовуєте оператор присвоєння, його права частина має бути правильного типу, тобто такого типу, на який очікує ліва частина.

Тип, який ми досі позначали словом «текст», насправді в системі .NET називають «рядком». Коли задаєте рядкове значення атрибута, не забудьте взяти текст у лапки: `Button.Text = "Клацніть тут"`.

Основні типи даних

Система .NET підтримує певну кількість основних типів даних, які називають примітивними. Ви вже користувалися рядковим типом для атрибута `Text` кнопки, цілочисловим типом для атрибута `Height` форми та навіть булевим типом для атрибута

`Visible` текстового поля (цей тип має два значення: істинно чи хибно).

До примітивних типів, які найчастіше використовують, належать: `String` (текстовий рядок), `Integer` (ціле число), `Single` (десяткове число) та `Boolean` (істинно чи хибно, або `True` чи `False`). Такі примітивні типи є спільними для всіх мов програмування .NET.

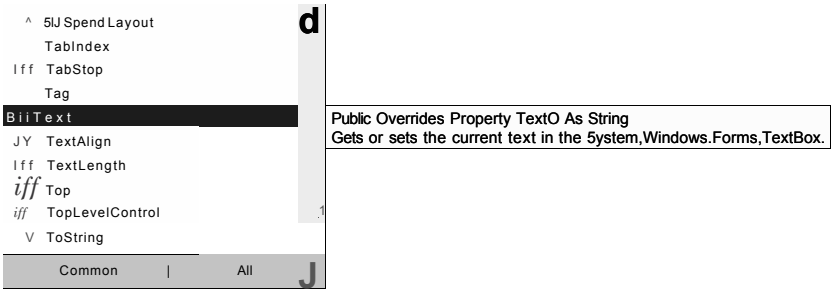
Ось приклади значень, які відповідають кожному з цих примітивних типів.

Тип	Приклад	Примітки
`String`	`"Привіт"`	Текст має бути взятий у лапки
`Integer`	`123`	Цілі числа без десяткової крапки
`Single`	`55.12`	Числа з десятковою крапкою
`Boolean`	`True`	Є лише два значення цього типу: `True` та `False`

Погляньте на приклади операторів присвоєння, в яких значення надаються атрибутам різних типів:

```
TextBox1.Text = "Посміхніться"
TextBoxl.Visible = True
TextBoxl.Width = l000
```

Коли ви обираєте атрибут або метод зі списку IntelliSense, відображається підказка, що повідомляє, значення якого типу слід присвоїти атрибуту або аргументи якого типу потрібні методу.



Поняття змінної

Під час написання коду *змінні* використовують для зберігання даних, які можуть знадобитися пізніше. Змінні у програмуванні схожі на змінні в математиці. Їм надають значення, а також їх можна використовувати замість значень.

Отже, змінні застосовують у програмі для «запам'ятовування» даних. Якщо ви в коді не використовуватимете змінні, програмі доведеться зупинятися та пропонувати користувачеві вводити інформацію кожного разу, коли вона знадобиться. Уявімо, що у програмі тричі використовується вік користувача. Чи не буде це дратувати, якщо програма тричі зупинятиметься та пропонуватиме користувачеві ввести свій вік?

```
Введіть свій вік.
Введіть свій вік.
Введіть свій вік.
```

Якщо застосувати змінну, програма запропонує користувачу ввести свій вік лише один раз. Значення віку буде збережено у змінній, яка використовуватиметься пізніше, коли програмі знову знадобиться вік користувача.

Призначення змінних

Як уже зазначалося, використання змінних дає можливість заощадити час користувача. За допомогою змінних можна значно скоротити обсяг інформації, що потрібно вводити, а також зменшити кількість помилок. Якби вам, наприклад, довелося вводити довгі числа знову й знову, ви могли б наробити помилок.

Окрім зберігання введених користувачем даних, змінні мають ще й інше призначення. Їх застосовують для зберігання результатів обчислень; здійснення порівнянь, що визначають порядок

виконання коду; підрахунку певних речей, наприклад того, скільки разів потрібно виконувати обчислення.
Змінні використовують в усіх мовах програмування. Вони необхідні для виконання обчислень і керування роботою програми. Вміння працювати зі змінними — основна навичка програміста.

Локальні й глобальні змінні

Перш ніж у програмі можна буде використовувати змінні, їх треба оголосити. *Оголошення змінної* — це рядок коду, в якому зазначено ім'я змінної та її тип. У Visual Basic .NET необхідно оголошувати всі змінні. Це змушує замислюватися над тим, як використовуватиметься змінна та якого вона буде типу. Оголошення змінних прискорює компіляцію програми та робить її виконання ефективнішим, а також запобігає неправильному написанню імен змінних у коді.
Щоб оголосити змінну у Visual Basic .NET, слід використовувати оператор `Dim`, що має такий синтаксис:

```
Dim Ім'я Змінної As Тип Змінної
```

Ось кілька прикладів оголошення змінних:

```
Dim MyName As String
Dim MyWeight As Integer
Dim NoBrainer As Boolean
Dim DVDPrice As Single
```

Змінну `MyName` оголошують як змінну типу `String`, їй можна надавати лише рядкові значення. Змінна `MyWeight` має тип `Integer`, їй можна надавати лише цілі числа. Змінній `NoBrainer` можна надати лише значення типу `Boolean`, а змінній `DVDPrice` — лише значення типу `Single`.

Важливим є ім'я, яке ви оберете для своєї змінної. Імена мають бути описовими, але короткими. Скорочуйте їх до абревіатур лише в тому випадку, коли це конче потрібно. Багато програмістів користується стандартом найменування, що отримав назву «верблюжий». За ним кожне слово у змінній починається з великої літери, наприклад, MyName, TopSpeed, LocalSpeedLimit. Система автоматично робить першу літеру імені великою.

У коді Visual Basic .NET ви можете оголосити стільки змінних, скільки потрібно. Але зробіть це так, щоб вони були чи то глобальними, чи то локальними. Досі весь код ми писали в обробниках подій, наприклад, в обробнику події клацання кнопки. Усі змінні, які оголошуються в коді обробників подій, є *локальними*. Локальні змінні можуть зчитуватися або змінюватися лише тим обробником події, в якому вони оголошені. До них не можуть отримувати доступ обробники інших подій або код з будь-яких інших форм. Значення локальних змінних зберігаються лише доти, доки виконується код обробника події. Після його виконання значення локальних змінних втрачаються.

Для того щоб оголосити локальні змінні, оператор `Dim` потрібно розмістити всередині коду обробника події. Змінна, що використовується обробником події, має бути оголошена на початку його коду, тобто перед будь-якими іншими операторами. Наприклад:

```
Private Sub Button1 Click(ByVal sender As System.Object,
  ByVal e As System.EventArgs) Handles Button1.Click
    Dim MyName As String MyName = "Білл"
    TextBox1.Text = MyName
End Sub
```

Досі ми не користувалися *глобальними* змінними. До них може отримати доступ будь-який обробник подій і весь код певної

форми. Значення глобальних змінних зберігаються доти, доки відкрито форму. Прикладом такої змінної є лічильник, який підраховує загальну кількість клацань усіх кнопок форми.
Щоб оголосити глобальну змінну, яку зможе використовувати ввесь код форми, потрібно вставити оператор `Dim` у тіло коду форми. Найкраще оголошувати змінні в коді форми відразу після такого рядка:

```
Windows Form Designer generated code
```

Цей рядок коду обрамлений прямокутником, а ліворуч від нього розташований знак +. Погляньмо на приклад:

```
Windows Form Designer generated code
Dim TotalButtonClicks As Single
```

Змінна `TotalButtonClicks` є глобальною. Її значення може бути зчитане та задане в коді форми, зокрема, у коді обробників подій. Якщо ви оголосите всі змінні на початку коду форми, вони будуть глобальними. Їх значення зможуть зчитувати або задавати всі оператори форми та обробників подій, пов'язаних з її використанням.

— А навіщо стільки числових форматів? Та що буде з числом, якщо його зберегти в різних форматах? — запитав Михась.

— Саме це ми зараз і перевіримо, — відповів ВВ. — Відкрийте Visual Studio .NET, виконайте команду **New Project** (Новий проект) і створіть проект PiValue.

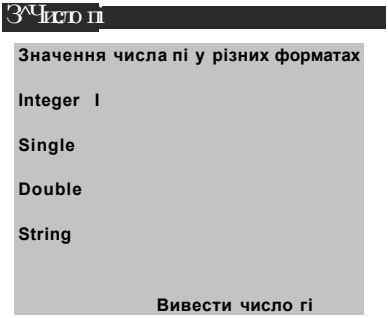
Вправа 6.1. Числові формати

Відобразьте число π як значення змінних різних типів. Файли, які вам можуть знадобитися для виконання цієї вправи, містяться в папці **Вправа_6.1.**

Створіть таку форму, як показано далі на рисунку.

У коді події клацання кнопки оголосіть змінні таких типів:

- `Integer`;
- `Single`;
- `Double`;
- `String`.



За допомогою вбудованого калькулятора Visual Basic надайте значення числа п кожній змінній. Для отримання цього значення достатньо записати `Math.PI`.

Присвойте кожну змінну відповідному текстовому полю.

Ініціалізація змінних

Коли змінну оголошено, ви можете використовувати її в коді. Перед усім, змінній слід надати початкове значення. Ним може стати будь-яке значення відповідного типу (тобто того типу, який має сама змінна). Щоб надати змінній значення, використовують оператор присвоєння. Перше присвоєння змінній значення називають її *ініціалізацією*. Звернімося до прикладів.

```
Dim MyName As String
MyName = "BB/550"

Dim MyWeight As Integer
MyWeight = 60

Dim DrivingDistance As Single
DrivingDistance = 12.8
```

Змінну `MyName` оголошують як змінну типу `String` і їй надають рядок `"BB/550"`. Для змінної `MyWeight` обрано тип `Integer`, тому їй надають ціле число `60`, а змінній `DrivingDistance`, що має тип `Single`, — дробове значення `12.8`. Дуже зручно поєднувати оголошення змінної з її ініціалізацією. Оголошуючи змінну та задаючи її початкове значення в

одному рядку коду, ви гарантуєте, що змінна міститиме певне значення.

```
Dim MyAge As Integer = 100
```

Використання змінної, якій не було присвоєне початкове значення, може призвести до збою програми.

Змінні в коді

Оголосивши змінну та надавши їй початкове значення, ви можете використати значення змінної в коді там, де захочете. Розглянемо приклади.

```
Dim MyName As String
MyName = "Джо Кокер"
MessageBox.Show(MyName)
```

Як ви гадаєте, що відображатиметься у вікні повідомлення? Звичайно, «Джо Кокер»! Інший варіант:

```
Dim MyName As String
MyName = "ВВ/550, Михась і Даринка"
TextBoxl.Text = MyName
```

Ви можете змінити значення змінної, використавши додаткові оператори присвоєння:

```
Dim MyName As String
MyName = "ВВ/550"
MyName = "Михась"
MyName = "Даринка"
TextBoxl.Text = MyName
```

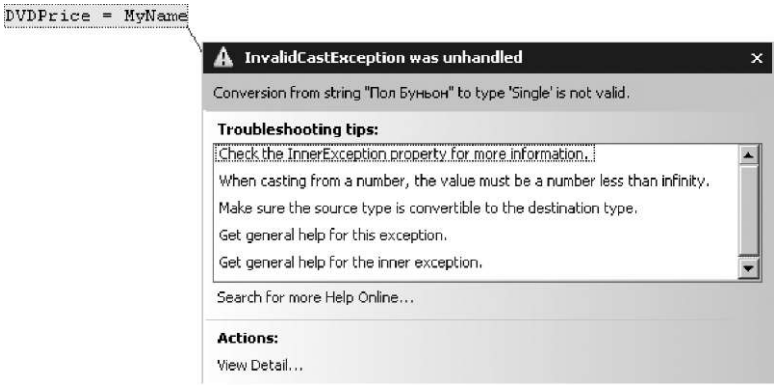
Яке ім'я відображатиметься в текстовому полі TextBoxl? Правильна відповідь — Даринка. Змінній MyName надається значення ВВ/550, потім Михась і, нарешті, Даринка. Оскільки останнє значення, присвоєне змінній MyName, — Даринка, саме воно і відобразиться у полі TextBoxl.
Ви можете надати значення однієї змінної іншій.

```
Dim MyName As String
Dim YourName As String
MyName = "Ален Гінсберг"
```

```
YourName = MyName
MessageBox.Show(TourName)
```

Ім'я Ален Гінсберг відобразиться у вікні повідомлення.
У прикладі, що наведений нижче, допущено помилку. Через те, що ми намагаємося надати змінній одного типу значення змінної іншого типу, такий код не компілюватиметься.

```
Dim MyName As String
Dim DVDPrice As Single
MyName = "Грейс Хоппер"
DVDPrice = MyName
```



Пам'ятайте, що ви оголошуєте змінну певного типу, якій можна надавати значення лише того самого типу.

Розробка програми з використанням змінних

Я розповів вам про типи змінних, про їх використання та про те, як їх оголошувати. Тепер спробуйте написати програму, використовуючи змінні!

Створіть Windows-програму в середовищі Visual Studio .NET і назвіть її ShowXY. Відкрийте вікно **Toolbox** (Панель інструментів), додайте кнопку та два текстових поля до форми `Forml`. Видаліть значення атрибута `Text` текстових полів `TextBoxl` та `TextBox2`. Надайте атрибуту `Text` кнопки `Buttonl` значення

Показати XY, атрибуту Readonly текстових полів TextBox1 та TextBox2 — значення True. Двічі клацніть кнопку Button1, щоб відредагувати код обробника події клацання кнопки. Введіть такі рядки коду.

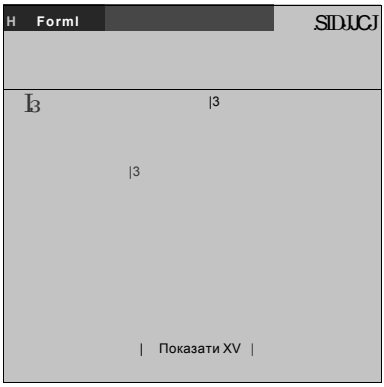
```
Dim XName As String
Dim YName As String
XName = "X — це моє ім'я"
YName = "Y — це моє ім'я"
TextBox1.Text = XName
TextBox2.Text = YName
```

Побудуйте проект та запустіть його. Клацніть кнопку **Показати XY.** У програмі оголошуються дві рядкові змінні, XName та YName, яким надають рядки тексту. Потім атрибуту Text текстового поля TextBox1 присвоюють значення, що зберігається у змінній XName, а атрибуту Text текстового поля TextBox2 — значення, що зберігається у змінній YName. Спробуйте змінити значення змінних XName та YName і знову запустити програму. Тепер модифікуємо цю програму. Додайте третє текстове поле. Видаліть значення атрибута Text текстового поля TextBox3, а атрибуту Readonly цього ж поля надайте значення False. Двічі клацніть кнопку Button1, щоб відредагувати код події клацання кнопки. Змініть код у такий спосіб:

```
Dim XName As String
Dim YName As String
Dim ZName As String
ZName = TextBox3.Text
XName = ZName
YName = XName
TextBox1.Text = XName
TextBox2.Text = YName
```

Побудуйте та запустіть проект. Клацніть кнопку **Показати XY.** Що сталося? У коді оголошуються три рядкові змінні: XName, YName та ZName. Змінній ZName надається значення атрибута Text текстового поля TextBox3, змінній XName — значення змінної ZName (якій вже було надано значення атрибута Text текстового поля TextBox3), а змінній YName — значення змінної Xname (якій щойно було надано значення змінної

ZName). Після цього атрибуту Text текстового поля TextBoxl надається значення змінної XName, а атрибуту Text текстового поля TextBox2 — значення змінної YName. Отже, я показав, як надавати значення однієї змінної іншій.



Обмін значеннями між змінними

— Мені здається, що програма ShowXY не дуже корисна, — зауважив Михась. — Було б цікавіше, якби текстові поля обмінювалися своїм вмістом.

— Так, — відповів ВВ, — обмін значеннями між двома змінними, наприклад між двома текстовими полями, будь-якому програмісту доводиться кодувати майже щодня. Це, мабуть, найпоширеніший прийом у програмуванні.

— Але ж це дуже просто, — втрутилася в розмову Даринка. — Спочатку атрибуту TextBoxl.Text слід присвоїти значення атрибута TextBox2.Text, а потім атрибуту TextBox2.Text присвоїти значення атрибута TextBoxl.Text — і обмін значеннями здійснено!

— Ця простота оманлива, — посміхнувся ВВ. — Ніякого обміну значеннями в тебе не відбулося. Припустимо, що поле TextBoxl містило текст Textl, а поле TextBox2 — текст Text2. Що станеться після виконання першого присвоєння?

```
TextBoxl.Text = TextBox2.Text
```

Поля TextBoxl і TextBox2 міститимуть один і той самий текст — Text2. Тобто ми замінимо значення атрибута TextBoxl . Text значенням атрибута TextBox2 . Text, і тому друге присвоєння, TextBoxl. Text = TextBox2 . Text, вже ні на що не буде впливати. Після його виконання обидва поля, як і раніше, міститимуть текст Text2.

Для розв'язання цієї задачі нам знадобиться допоміжна змінна, скажімо, змінна tmp типу String. Замініть код обробника події клацання кнопки Buttonl у програмі ShowXY таким:

```
Dim tmp As String
tmp = TextBoxl.Text
TextBoxl.Text = TextBox2.Text
TextBox2.Text = tmp
```

Побудуйте проект і запустіть його. Введіть у поля TextBoxl і TextBox2 різний текст, скажімо числа 1 і 2. Клацніть кнопку Buttonl — вміст цих полів буде поміняно місцями.

Як цей код працює? Перший оператор присвоєння записує вміст поля TextBoxl у змінну tmp:

```
tmp = TextBoxl.Text
```

Потім ми замінюємо значення атрибута TextBoxl.Text значенням атрибута TextBox2.Text:

```
TextBoxl.Text = TextBox2.Text
```

— Але ж попереднє значення атрибута TextBoxl.Text зберігається в змінній tmp! — вигукнув Михась.

— Його ми й надаємо атрибуту TextBox2.Text у третьому присвоєнні:

```
TextBox2.Text = tmp
```

Використання глобальних змінних

А зараз на прикладі я покажу вам, як можна використовувати глобальні змінні. Створіть новий проект, що називатиметься TotalButtonClicks (Усього клацань кнопки). Додайте три кнопки до форми Forml. У вікні коду знайдіть такий рядок:

```
Windows Form Designer generated code
```

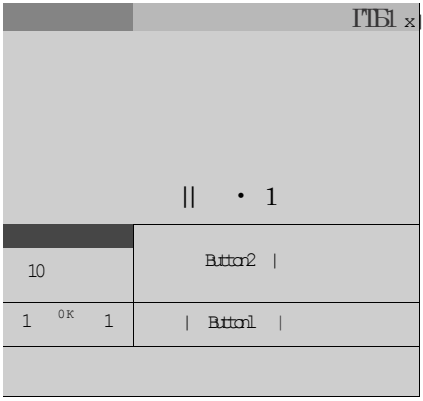
Після нього додайте такий рядок:

```
Dim TotalButtonClicks As Integer
```

Двічі клацніть кнопку Button1 та додайте до обробника події клацання кнопки такий код:

```
TotalButtonClicks = TotalButtonClicks + 1
MessageBox.Show(TotalButtonClicks)
```

Додайте той самий код до обробників подій клацання кнопок Button2 та Button3. Побудуйте та запустіть проект. Клацніть усі три кнопки по черзі. Кожного разу загальна кількість клацань кнопок виводитиметься у вікні повідомлення. Цей код працює, оскільки ми оголосили глобальну змінну TotalButtonClicks у коді форми. Глобальну змінну використовують для збереження загальної кількості клацань кнопок форми. Кожного разу, коли користувач клацає кнопку, значення, що зберігається у змінній TotalButtonClicks, збільшується на одиницю. Значення глобальної змінної не втрачається, доки не буде закрито форму.



Типи даних .NET

У мові Visual Basic .NET ви можете оголошувати змінні не лише примітивних типів, до яких належать String, Integer, Boolean, Single. Є й інші вбудовані до системи .NET типи, які

можна використовувати в будь-яких мовах родини .NET. Багато з них — це атрибути системних класів, які містять код, що забезпечує основні функціональні можливості мов .NET. Вивчення системних класів та їх функціональних можливостей — це один із найважливіших етапів навчання програмування однією з мов .NET, проте вам не доведеться робити це саме зараз. Я лише надам вам код, у якому використовуються атрибути або методи системних класів для визначення фонового кольору форми. Тож було б добре, якби ви змогли їх розпізнати.

Відкрийте проект ShowXY та додайте другу кнопку до форми `Forml`. Змініть значення атрибута `Text` кнопки `Button2` на `"Color"`. Двічі клацніть кнопку `Button2`, щоб відредагувати код обробника події клацання кнопки.

Додайте такий код:

```
Dim MyColor As System.Drawing.Color
MyColor = System.Drawing.Color.Blue
Forml.ActiveForm.BackColor = MyColor
```

Побудуйте та запустіть проект. Клацніть кнопку **Color** — форма стане синьою. Чому так відбувається? У коді оголошується змінна `MyColor` типу `System.Drawing.Color`. Їй було надано значення `System.Drawing.Color.Blue`, що є одним із допустимих значень типу `System.Drawing.Color`. Атрибуту `BackColor` форми `Forml` було надано значення змінної `MyColor`. Зауважте, що цьому атрибуту надається значення типу `System. Drawing.Color`.

Ви можете самостійно вивчити системні класи. Для цього у вікні коду введіть `System.` й IntelliSense відобразить список усіх атрибутів і методів класу `System`. Виберіть атрибут або метод і введіть ще одну крапку (`.`), щоб побачити, чи є в нього свої атрибути та методи.

— Ну, як вам тема? Не надто складно? — запитав ВВ.

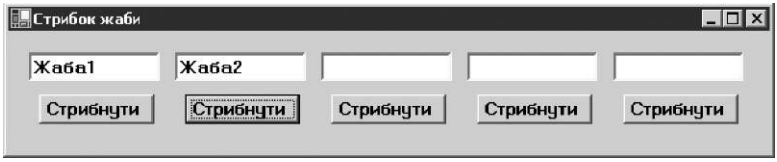
— Ні, але замало практичних завдань, — відповів Михась.

— Це дуже легко виправити: перш ніж вирішувати, як нам розважитися сьогодні ввечері, давайте розв'яжемо ще кілька задач.

Вправа 6.2. Стрибок жаби

Ставок із жабами, який побачили ВВ, Михась і Даринка цього ранку (на жаль, тільки по телевізору), надихнув їх на написання програми FrogLeap (Стрибок жаби).

Створіть форму такого вигляду, як зображено на рисунку, або використайте готовий шаблон із папки **Вправа_6.2.**



Атрибуту `TextBox1.Text` надайте рядок `"Жаба1"`, а атрибуту `TextBox2.Text` — рядок `"Жаба2"`.

Коли користувач клацає кнопку **Стрибнути,** вміст текстового поля над кнопкою надається полю, що розташоване на два поля правіше. Не забудьте очистити поточне поле.

Завдання 6. Робота над помилками

Виконуючи це завдання, ви шукатимете помилки у програмі. У кожному рядку коду програми з розробки **splat.sln,** що записана в папці **Завдання_6,** є помилка. Загалом таких помилок десять. Їх необхідно виправити, щоб побудувати та виконати проект. Ви можете набрати код самостійно, якщо запропонований варіант є зовсім недоречним.

Додаткове завдання

Чи знаєте ви, хто така Грейс Хоппер? І яке відношення вона має до програмування? Щоб знайти цю інформацію, можете скористатися Інтернетом.

Коли вправи та завдання було виконано, усі троє почали вигадувати, чим би зайнятися, адже залишалося багато місць, куди б вони ще хотіли потрапити.

— Давайте підемо в кіно! — запропонувала Даринка.

— Та ну, там мало що змінилося: лише вдосконалили ефект тривимірного простору і додали ще кілька дрібниць, — заперечив ВВ.

— А комп'ютерні клуби у вас є? — запитав Михась.

— Після створення тотальної безпроводової мережі ці клуби мало хто відвідує, — відповів ВВ, а потім додав: — Друзі, а ви каталися коли-небудь на аероциклах? Я не пам'ятаю, чи були вони у вашому часі, чи ні?

— На чому? — запитав здивовано Михась.

— А що це таке? — не менше здивувалася Даринка.

— Ага, — зрадів ВВ, вже точно знаючи чим вони займуться сьогодні. — Вам це обов'язково має сподобатися. Аероцикл подібний до мотоцикла, але він літає. Хочете спробувати прокататися на ньому?

— А це безпечно? — засумнівалася Даринка.

— Цілком, — запевнив її ВВ. — То що — по циклах?

— Пішли! — вигукнув Михась.

— Але спочатку нам потрібно пройти тест... — буденно заявила Даринка.

Кодовий замок

1. Скільки елементів може бути вказано ліворуч від символу «=» в операторі присвоєння?
   а) один;
   б) два;
   в) скільки завгодно.
2. Який тип має атрибут `Text` текстового поля?
   а) `Single`;
   б) `Integer`;
   в) `String`.
3. Для чого використовують змінні?
   а) для зміни значень атрибутів;

б) для запам'ятовування значень;

в) для завершення програми.

4. Що потрібно зробити, перш ніж використовувати змінну в програмі?

а) оголосити змінну;

б) обчислити значення змінної;

в) змінну можна використовувати, не виконуючи жодних попередніх дій.

5. Чи може змінюватися тип змінної?

а) так, за допомогою функції `Val`;

б) так, у разі надання змінній значення іншого типу;

в) ні, не може.

6. Чи обмежена кількість змінних у програмі?

а) ні, можна створювати скільки завгодно змінних;

б) у кожній мові програмування є своє обмеження на кількість змінних;

в) єдиним обмеженням є обсяг пам'яті, яка може бути виділена для зберігання значень змінних.

7. У чому полягає головна відмінність між локальними і глобальними змінними?

а) локальні змінні доступні лише в межах підпрограм, де вони оголошені, натомість глобальні змінні доступні в будь-якому місці коду;

б) імена глобальних змінних мають починатися тільки з великої букви, а імена локальних змінних — з будь-якої;

в) зберігання значень локальних змінних потребує менше пам'яті, ніж глобальних.

8. Чим відрізняється ініціалізація від присвоєння?

а) ініціалізація — це перше присвоєння значення змінній;

б) ініціалізація виконується швидше, ніж присвоєння;

в) немає жодної відмінності: будь-яке присвоєння можна вважати ініціалізацією і навпаки.

День 7
Програма — набір операцій

Цілу ніч Михасю та Даринці снилися польоти й падіння. Вони прокидались із криками, тому виспатися добре їм не вдалося: все-таки важко звикнути до того, що легесенький апарат, який різко реагує на кожен твій рух, є надійним і безпечним.

Михась почав літати на аероциклі відразу після того, як ВВ пояснив йому принцип керування цим механізмом. А потім вони довго вмовляли Даринку хоча б спробувати посидіти на задньому сидінні. Нарешті, коли ВВ і Михась уже дві години каталися містом (на аероциклі було набагато зручніше та швидше оглядати місто майбутнього), Даринка згодилася спочатку політати разом із ВВ, а потім спробувала сама покерувати цим незвичайним видом транспорту.

Наступного ранку Михась із Даринкою ледве-ледве зосередившись, приступили до роботи. А ВВ, як і завжди, бадьоро розпочав свою чергову розповідь.

Учора ви вже навчилися користуватися змінними та операціями присвоєння, але в мовах програмування є й багато інших операцій, про які ми поговоримо сьогодні. Операції — один з основних інструментів програмування. Коли ви навчитеся використовувати операції у Visual Basic .NET, то зможете зробити це в будь-якій мові програмування. Наприкінці я покажу

вам, як користуватися деякими засобами налагодження програм, що є частиною Visual Studio .NET. Вони допоможуть вам уникати помилок під час кодування.

Операції та операнди

Прикладами операцій є додавання, віднімання, множення та ділення. Ви користуєтеся ними з першого класу. У мовах програмування існують й інші операції, зокрема це операції, призначені для виконання складних математичних обчислень та поєднання рядків тексту.

Операції зазвичай виконують над двома значеннями, хоча деякі — лише над одним. Значення називають *операндами*. Ось основний синтаксис використання операцій:

```
операнд1 операція операнд2
```

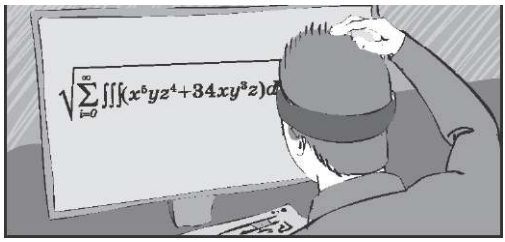
Наприклад,

```
3 + 4
```

У виразі 3 + 4 — два операнди (3 та 4), що з'єднуються операцією додавання (+). Інакше кажучи, операція (у цьому випадку додавання) виконується над двома операндами.
Працювати з операціями легко та цікаво. Використовуючи їх, ви можете будувати складні вирази. Почнемо з розгляду арифметичних операцій: додавання, віднімання, множення та ділення. Потім я покажу, як з'єднувати текстові рядки, і ми розглянемо спеціальну операцію `NOT`.

Арифметика у Visual Basic

— Математичні правила! Мені подобається математика! — вигукнув Михась.



— Я б також залюбки вирішував математичні задачі цілими днями, — підтримав Михася ВВ,

а потім додав: — Але коли я втомлююся, то починаю робити помилки. Комп'ютерні програми чудово справляються з математичними задачами: вони не допускають помилок і ніколи не втомлюються виконувати ту саму операцію повторно. Але, відверто кажучи, є невеличка проблема. Хоча програми чудово знаходять відповіді, вони не можуть формулювати задачі. Це ваше завдання як програміста. Тому важливо, щоб ви зрозуміли, як використовувати арифметичні операції в коді. За їх допомогою ви повідомлятимете програмі, які математичні проблеми хочете вирішити. Усі мови програмування забезпечені стандартним набором арифметичних операцій. Оскільки ці операції часто застосовують, символи, що використовуються для їхнього відображення в коді, майже однакові в усіх мовах програмування.

Ось список найбільш вживаних арифметичних операцій, звичайних символів для їх позначення та символів, що використовуються у програмуванні. Вони відрізняються лише для операції множення: у коді використовують символ «*» замість «x». А от ділення в коді позначається лише символом «/».

Операція	Звичайний символ	Символ у коді
Додавання	+	+
Віднімання	-	-
Множення	x	*
Ділення	/ або :	/

Погляньмо на кілька прикладів застосування арифметичних операцій у коді:

```
3 + 4;
18 - 2;
33 * 3;
66 / 6.
```

Конкатенація рядків

Не всі програми, які ви створюватимете, виконуватимуть обчислення. Іноді програмі потрібно лише обробляти текстові

рядки. Вам відомо, що у Visual Basic .NET та інших мовах програмування ви можете «додавати» текст? Таке додавання не схоже на звичайне додавання чисел. Додавання тексту називають *конкатенацією*. Це лише чудернацька назва операції об'єднання двох рядків. У мові Visual Basic .NET операція конкатенації позначається як символом «&», так і символом «+». Конкатенацію використовують так само, як і операцію додавання, але з текстовими рядками замість чисел.
Ось кілька прикладів:

```
"Богдан" + "Ігор"

"1" & "2"
```

Коли ви застосовуєте операцію конкатенації для об'єднання двох рядків, Visual Basic .NET автоматично не додає пробілів. Вам потрібно додати їх до тексту самостійно. Для цього можете використовувати будь-який із таких методів:

```
"Богдан" & " " & "Ігор"
```

або

```
"Богдан " & "Ігор"
```

Логічне заперечення

Операція `Not` працює лише з одним операндом булевого типу (він має значення `True` або `False`). Операція `Not` змінює значення `False` на `True`, а `True` на `False`. Погляньте на такі приклади:

```
Not  (True)              'отримуємо  False
Not  (Not  (True))       'отримуємо  True
```

Операції в операторах присвоєння

Як застосовувати операції в коді? Найчастіше їх використовують в операторах присвоєння. Пригадайте, що присвоєння — це вираз із символом «=». Операції та їхні операнди розміщуються праворуч від цього символу. Спочатку обчислюється значення виразу з правого боку від символу «=», а потім лівій частині оператора присвоюється це значення. Результат обчислення правої частини має бути значенням того ж типу,

що й змінна в лівій частині, інакше компіляція коду не буде успішною. Ось кілька прикладів застосування арифметичних операцій і операцій конкатенації у присвоєннях.

```
Dim FormWidth As Integer
FormWidth = 200 + 300

Dim FormHeight As Integer
FormHeight = 1000 / 2

Dim TransportName As String
TransportName = "Аеро" & "цикл"
```

Конструювання виразів

Приклади, які я щойно навів, досить прості. У кожному з них використовується лише два операнди та одна операція. Насправді вам часто доведеться перекладати формули на мову програмування чи виконувати обчислення, в яких використовується багато операцій і операндів. На щастя, у більшості мов програмування можна утворювати вирази зі стількох операцій і операндів, скільки потрібно для розв'язання задачі. Крім того, ви можете користуватися змінними операндами. Саме так! Змінні також можуть бути операндами, якщо належать до відповідного типу. Більше того, ви навіть можете присвоїти змінній вираз, що містить ту саму змінну. Тож погляньмо на кілька складніших прикладів. У першому прикладі обчислюється довжина кола шини автомобіля.

```
Dim TireCircum As Single
Dim TireDiam As Integer
Dim PiValue As Single
PiValue = 3.14159
TireDiam = 18
TireCircum = TireDiam * PiValue
```

У наступному прикладі обчислюються середні витрати на електроенергію за березень та квітень. Тут використовуються дужки для того, щоб спочатку виконувалося додавання, а потім — ділення.

```
Dim MarchCost As Single = 123.66
Dim AprilCost As Single = 231.45
Dim AvgCost As Single
AvgCost = (MarchCost + AprilCost) / 2
```

Використовуйте дужки для керування порядком обчислень так само, як і під час розв'язування звичайних математичних задач. Якщо брати вираз у дужки, він буде обчислюватися першим. Наприклад:

```
(5 + 7)/(1 + 5) = 2
```

У цьому прикладі спочатку обчислюється значення виразу (5 + 7), а потім (1 + 5). Після цього виконується ділення. У результаті отримуємо 2.

Якщо виконувати обчислення без дужок, то відповідь буде іншою:

```
5 + 7/1 + 5 = 17
```

Збільшення значення змінної

Пригадуєте, я казав, що можна присвоїти змінній вираз, який містить ту саму змінну? Вас цікавить, як така конструкція виконуватиметься? Подивіться на приклад і спробуйте визначити, що буде відображено у вікні повідомлення.

```
Dim MileCounter As Integer
MileCounter = 100
MileCounter = MileCounter + 200
MileCounter = MileCounter + 400
MsgBox (MileCounter)
```

— Мабуть, число 700? — невпевнено запитав Михась.

— Так, — впевнено відповів ВВ. — У двох перших рядках коду оголошується змінна `MileCounter`, якій надається початкове значення — 100. У третьому рядку коду до поточного значення змінної `MileCounter` (100) додається 200, і ця змінна набуває значення 300. Згадайте: права частина оператора присвоєння (`MileCounter + 200`) завжди обчислюється першою. Лівій частині оператора присвоєння (змінній `MileCounter`) надається значення правої частини оператора присвоєння (300). У четвертому рядку коду до поточного значення змінної `MileCounter` (тепер воно становить 300) додається 400, отже, отримуємо 700. Змінній `MileCounter` надається

це значення, яке потім виводиться у вікні повідомлення. Ліва частина оператора присвоєння «нічого не знає» про спосіб обчислення правої частини, важливо лише, щоб типи частин оператора присвоєння були узгоджені. Ідентифікатор `MileCounter` у правій частині — це не ім'я лівої частини, а ім'я незалежної змінної.

У програмуванні часто практикується надання змінній значення виразу, що містить саму змінну. Так ми уникаємо потреби створювати ще одну змінну, яка б тимчасово зберігала проміжне значення. Як показано у прикладі, що наведений вище, цим методом можна користуватися для обчислення послідовності значень. Ви знаєте, що в будь-якому коді найчастіше трапляється рядок типу `змінна = змінна + 1`?

— Знаю! — вигукнув Михась. — Тато мені колись казав, що така конструкція в деяких мовах замінюється на простіші, щоб скоротити написання! Цей рядок зазвичай застосовується для визначення кількості ітерацій циклу.

— Так, — запевнив їх ВВ. — Наприклад, у мові C і всіх похідних від неї цей рядок спрощується до `змінна++`.

Застосування логічних і рядкових операцій

— Щось забагато математики! — застогнала Даринка.

— Добре, спочатку я розкажу вам, як діє операція конкатенації, а потім наведу приклад із операцією `Not`, — погодився ВВ. — У першому прикладі значення текстових змінних об'єднуються, а результат виводиться в текстовому полі:

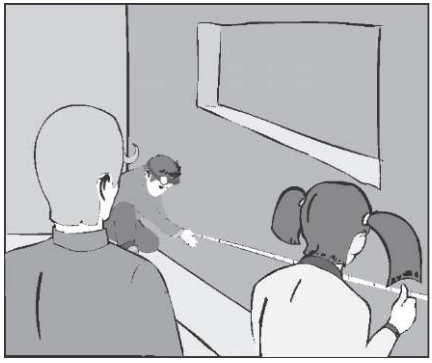
```
Dim FirstName As String
Dim LastName As String
FirstName = "Боб"
LastName = "Марлі"
TextBox1.Text = FirstName & " " & LastName
```

— Ви помітили, що я вказав Visual Basic .NET на необхідність додати пробіл між значеннями змінних `FirstName` та `LastName`? — запитав ВВ.

— Так, ти говорив, що операція конкатенації автоматично не додає пробілів, — відповіла Даринка.

У наступному прикладі показано, як операція `Not` використовується, щоб приховати одне текстове поле, коли з'являється інше. Пам'ятайте, що ця операція застосовується до значень булевого типу.

```
TextBox1.Visible = True
TextBox2.Visible = Not  (TextBox1.Visible)
```



— Ми вже стільки часу провели в цій кімнаті, — зауважив якось ВВ. — Що ви можете про неї сказати?

— Ну, — зніяковіло почала Даринка, — ми можемо розповісти, як розташовано деякі речі та на чому сидимо.

— А також про розміри кімнати, — додав Михась.

— У мене виникла ідея! — вигукнув ВВ. — А давайте обчислимо її периметр і площу!

— І об'єм! — захоплено додала Даринка.

— Я думаю, що саме час виконати вправу, — задоволено посміхнувся ВВ.

Вправа 7.1. Параметри кімнати

Розробіть програму, за допомогою якої можна обчислити периметр, площу та об'єм кімнати:

1. Створіть форму. Додайте до неї такі елементи:
   - три текстових поля для введення вхідних значень:
     - довжини;
     - ширини;
     - висоти;

- три текстових поля для виведення результату:
  - периметра;
  - площі;
  - об'єму;
- шість написів — по одному для кожного текстового поля;
- кнопку з написом, після клацання якої здійснюватимуться обчислення.

2. В обробнику події клацання кнопки обчисліть периметр, площу та об'єм кімнати. Формули, за допомогою яких це можна зробити, вам відомі з уроків математики.
3. Створивши форму, побудуйте проект. Якщо під час компіляції не виникло помилок, виберіть команду **Debug •Start Debugging,** щоб запустити програму.

Додаткове завдання

Спробуйте вдосконалити програму так, щоб вона знаходила відношення об'єму однієї кімнати до об'єму іншої. Для цього достатньо поділити знайдені значення об'ємів. Щоб зробити це, не змінюючи вигляду форми, результат виведіть у повідомленні.

Налагодження програми

— ВВ, як ти вважаєш, ми все правильно робимо? — запитав Михась.

— Ви чудово працюєте! — відповів ВВ, побачивши, як старанно Михась із Даринкою виконують завдання. — Написали багато коду, розробили та запустили чимало проектів. Можливо, не всі проекти вам удалося відкомпілювати відразу, але ви вже починаєте краще розуміти, як це робиться. Зараз я дам вам кілька підказок, за допомогою яких ви зможете швидко усувати будь-які помилки. Процес усунення помилок називають *налагодженням коду*. Він необхідний для того, щоб ваша програма виконувалася і давала правильні результати. Спочатку я покажу, як Visual Studio допомагає налагоджувати код під час його написання.

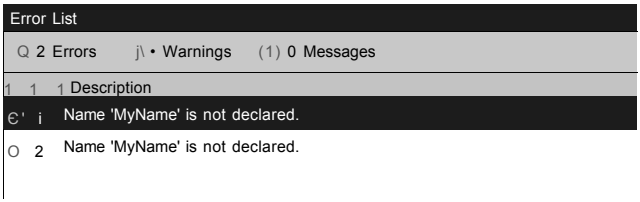
1. Створіть нову Windows-програму і назвіть її DebugView. Додайте одну кнопку та одне текстове поле до форми `Form1`. Двічі клацніть кнопку `Button1` та введіть код:

```
MyName = "Михась"
TextBox1.Text = MyName
```

2. Зауважте, що перед тим, як ви створите проект, у Visual Studio .NET відобразиться хвиляста синя лінія під словом `MyName` в першому та другому рядках коду. До чого все це? Середовище Visual Studio .NET досить «розумне», щоб «усвідомити» (навіть ще до розробки проекту), що ви не оголосили змінну `MyName`. Вас попереджують про те, що із кодом щось негаразд.

```
Private Sub Button1_Click(ByVal
    MyName = "Михаєі
    TextBox1.Text = MyName
End Sub
```

3. Побудуйте та запустіть проект. У вікні **Error List** (Список помилок) відобразиться повідомлення про те, що побудова не вдалася. У списку помилок пояснюється, чому це трапилося: два повідомлення про помилки вказують на те, що ім'я `MyName` не було оголошене **(Name 'MyName' is not declared).**



4. Двічі клацніть рядок **Name 'MyName' is not declared** у списку помилок. Буде виділено фрагмент коду з помилкою.

```
Private Sub Button1_Click(ByVal
    MyName = "Михась"
    TextBox1.Text = MyName
End Sub
```

5. Зауважте, що виділено змінну `MyName`. З нею і пов'язана проблема, яку потрібно усунути. А саме, в обробнику події клацання кнопки `Button1` перед операторами, які використовують змінну `MyName`, слід записати такий рядок коду:

   ```
   Dim MyName As String
   ```

   Хвиляста синя лінія зникне.

6. Побудуйте проект і запустіть його на виконання. Проект буде скомпільовано успішно — отже, ви завершили налагодження програми.

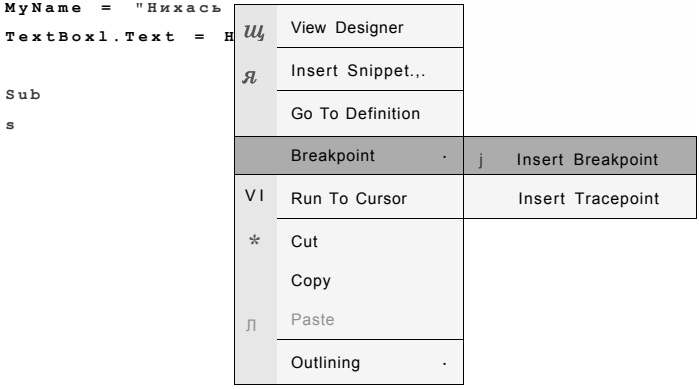
Виконання програми в покроковому режимі

Visual Studio .NET може визначити не всі помилки в коді. Програма може бути скомпільована без помилок, але результати її роботи будуть хибними. Це означає, що помилки містяться не в синтаксисі, а в логіці програми, або виникають під час її виконання. Такі помилки не розпізнаються автоматично, бо вони не порушують синтаксичні правила написання коду.
Для усунення цих помилок необхідно виконати програму в *покроковому режимі,* рядок за рядком. Це дає змогу відстежити, які саме оператори коду виконуються та як модифікуються значення змінних. Зазвичай ви задаєте в коді *точку переривання,* а потім запускаєте програму в покроковому режимі. Точка переривання діє як знак зупинки: на рядку коду, що позначений такою точкою, виконання програми переривається. Коли програма зупиниться, натисніть клавішу **F11,** щоб виконати наступний рядок коду. Зараз я покажу вам, як працює ця технологія.

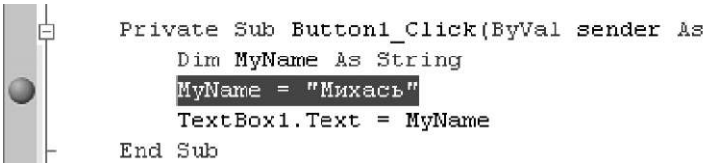
1. Знайдіть у коді обробника події клацання кнопки `Button!` рядок

   ```
   MyName = "Михась"
   ```

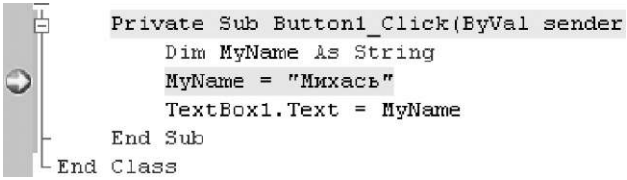
   Установіть курсор де-небудь на цьому рядку коду. Клацніть правою кнопкою миші та виберіть із контекстного меню команду **Insert Breakpoint** (Вставити точку переривання).



2. Зауважте, що рядок коду було виділено червоним кольором, а ліворуч від цього рядка відобразилося червоне коло. Побудуйте й запустіть проект, а після появи форми клацніть кнопку.



3. Код виконується та добігає рядка, який ви позначили точкою переривання (`MyName = "Михась"`). Цей рядок коду досі не виконувався. Його виділено жовтим, а в червоному колі біля нього тепер міститься жовта стрілка.



4. У вікні коду перемістіть курсор до змінної `MyName`. Буде відображено підказку, з якої ви дізнаєтеся значення цієї змінної. Зараз `MyName = Nothing`, оскільки змінну було

оголошено, але значення їй не надано. Клацнувши значення змінної, його можна змінити, навіть якщо в коді немає відповідного оператора.

```
Private Sub Buttonl_Click(ByVal
    Dim MyName As String
```

```
Private Sub Buttonl_Click(ByVal
```

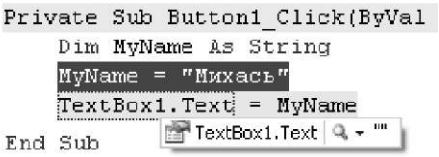
5. У вікні коду перемістіть курсор до атрибута `Text` текстового поля `TextBoxl`. З'явиться підказка зі значенням атрибута `Text`. Зараз `Text = ""` (пустий рядок).

```
Private Sub Buttonl_Click(ByVal sender As
```

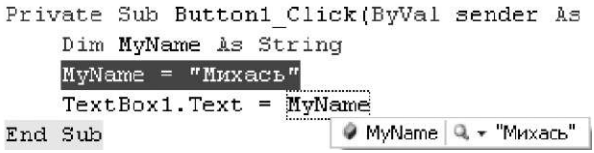
6. Натисніть клавішу **F11.** Буде виконано поточний рядок коду (`MyName = "Михась"`) і виділено жовтим кольором наступний рядок. Знову перемістіть курсор до змінної `MyName`. З'явиться підказка, яка покаже, що `MyName = "Михась"`, оскільки змінній було тільки-но надано це значення.

```
Private Sub Buttonl_Click(ByVal

                ^ it MyName     ' "Михась" |
```

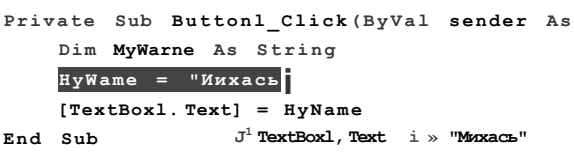
7. У вікні коду перемістіть курсор до атрибута `Text` текстового поля `TextBoxl`. Значення цього атрибута не змінилося: `TextBoxl.Text = ""`. Знову натисніть клавішу **F11.** Так буде виконано поточний рядок (`TextBoxl.Text = MyName`) та виділено жовтим наступний рядок.



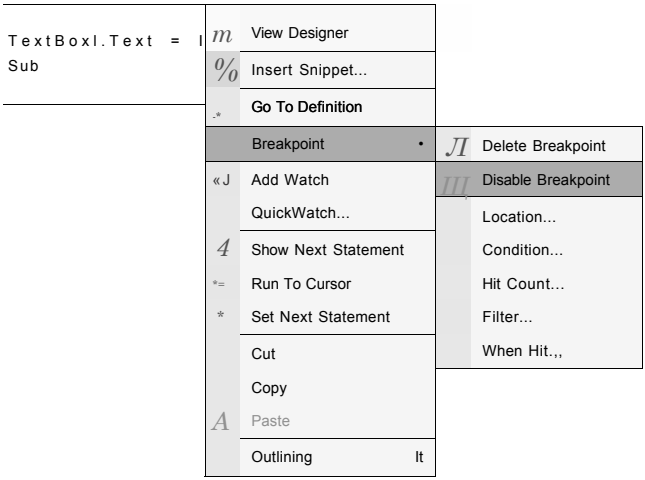
8. У вікні коду перемістіть курсор до змінної `MyName`. Підказкою буде `MyName = "Михась"`, оскільки значення цієї змінної не було змінене.



9 У вікні коду перемістіть курсор до атрибута `Text` поля `TextBoxl`. Підказка покаже, що `Text = "Михась"`, оскільки ми щойно надали цьому атрибуту значення змінної `MyName`. Натисніть клавішу **F11.** Знову з'явиться форма, адже виконання обробника події клацання кнопки було завершено. Щоб закрити форму, клацніть кнопку X в її правому верхньому куті.



10. У вікні коду клацніть правою кнопкою миші рядок із точкою переривання та виберіть команду **Disable Breakpoint** (Відключити точку переривання) з контекстного меню.



Ви завершили сеанс налагодження. Як бачите, переглядати код рядок за рядком зручно. Ви можете простежити шлях виконання коду та виявити, що вказали неправильний порядок виконання операторів або використали хибне ім'я змінної в операторі присвоєння. Можна також контролювати значення атрибутів і змінних, встановлюючи курсор на їхніх іменах. Значення оновлюються після виконання кожного рядка коду.

Чим складніший код, тим імовірніше виникне потреба в налагодженні. Налагодження складного коду потребує більших витрат часу. Згодом ви дізнаєтесь, як контролювати порядок виконання операторів `IF` та циклів. Стежити за виконанням цих операторів за допомогою інструментів налагодження надзвичайно цікаво.

— Щоб закріпити вивчений матеріал, — підсумував ВВ, — давайте розглянемо ще одну вправу та виконаємо завдання.

Вправа 7.2. Налагоджувач

У папці **Вправа_7.2** ви знайдете повноцінну програму. Вам не доведеться нічого до неї дописувати. Просто відшукайте файл **DebugMe.sln** та двічі клацніть його.

Ваше завдання полягає в тому, щоб записати в текстовому файлі, документі Word або у старому зошиті для домашніх робіт відповіді на три питання.

1. Яке значення ви ввели до текстового поля форми?
2. Яким є кінцеве значення змінної **AnswerOne?**
3. Яким є кінцеве значення змінної **AnswerTwo?**

Код цієї програми був ускладнений навмисне, щоб ви не змогли обчислити відповіді вручну. Вам потрібно лише помістити коричневе коло точки переривання в лівому вертикальному полі, потім запустити програму, і після виконання рядка коду, що помічений точкою переривання, навести курсор на змінну, значення якої ви хочете переглянути.
Налагодження — найцінніший навик, який ви можете застосовувати в усіх своїх майбутніх проектах.

— Пам'ятаєте кришталеві прес-пап'є, які ми бачили вчора в магазині подарунків? Яку форму вони мали? — спитав Михась.

— Так, одне було кубічної, а інше сферичної форми, — відповів ВВ.

— Мені хотілося б знати, яке з них важче, — пояснив Михась причину своєї зацікавленості.

— Оскільки вони обидва зроблені з однакового матеріалу, важчим буде те, що має більший об'єм, — спробував ВВ дати пояснення Михасеві, а потім додав: — А щоб краще в цьому розібратися, давайте розробимо програму, що обчислюватиме об'єм куба та сфери.

Завдання 7. Об'єм куба та сфери

Обчисліть об'єм куба та сфери. Файли, необхідні для виконання цього завдання, містяться в папці **Завдання_7.**
Створіть таку форму, як показано далі на рисунку.
Довжину грані куба потрібно записати в перше поле, а радіус сфери — у друге.

Якщо ви були не дуже уважними на уроках геометрії, нагадую вам деякі формули:

- Об'єм куба дорівнює довжині грані, піднесеній до третього степеня (тобто тричі помноженій на саму себе).
- Об'єм сфери становить чотири третіх числа `п`, помноженого на радіус в третьому степені. Ви можете використати 3,14 як значення числа `п` або застосувати функцію `Math.PI`.

Додаткове завдання

Знайдіть значення довжини радіуса сфери та грані куба, за яких значення об'ємів будуть однаковими принаймні до чотирьох знаків після коми в десятковому дробі.

Якщо зможете, додайте до форми другу частину, що даватиме можливість за об'ємом обчислити відповідні довжину грані та радіус.

— Усе. На сьогодні досить, — вимовив ВВ, побачивши, що його учні вже не здатні більше нічого сприймати, а потім додав: — Ну, чого ви такі... ніякі? Що з вами?

— Та то вчорашні польоти, — відповів Михась. — Не звикли ми до такого способу пересування.

— Це моя провина, — сказав ВВ. — Не потрібно було вам дозволяти стільки часу кататися на аероциклах. То чим ми займемося сьогодні?

— Нічим! — відповів Михась. — Бо скрізь встановлені якісь тести...

— Але тестів не уникнути в будь-якому випадку, — відповів ВВ, — а втрачений час не повернеш. Пропоную відразу виконати тест, а потім піти в басейн, який розміщується неподалік від нашого будинку.

Кодовий замок

1. Які символи використовують для змінення послідовності виконання операцій?
   а) точки;
   б) дужки;
   в) символ «&».
2. Яку операцію застосовують для об'єднання двох рядків?
   а) `And;`
   б) =;
   в) `&`.
3. Яким буде результат операції `"55" & "Один"`?
   а) 56;
   б) 55Один;
   в) операція некоректна.
4. Яким буде результат виконання операції `5 + 4 / 2 - 1`?
   а) 3,5;
   б) 9;
   в) 6.
5. Яким буде результат виконання операції `Not (54)`?
   а) 45;
   б) -54;
   в) операція некоректна.
6. Яка з цих помилок не розпізнається автоматично?
   а) помилка в послідовності запису операторів;

б) помилка в імені змінної;

в) помилка в типі значення, що присвоюється змінній.

7. Для чого виконують програму в покроковому режимі?

а) для виправлення синтаксичних помилок;

б) для виправлення логічних помилок;

в) для виправлення помилок, пов'язаних із введенням користувачем некоректних значень.

8. Яким кольором під час покрокового виконання виділяється рядок коду, що виконуватиметься наступним?

а) синім;

б) червоним;

в) жовтим.

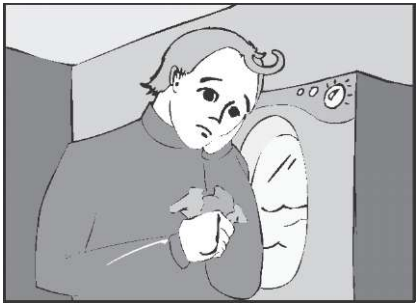
День 8
Прийняття рішень

Як відомо, вода має цілющі властивості. Тому після вчорашнього купання в басейні всі зранку відчували себе бадьоро та були готові до роботи. Доброю новиною для Михася та Даринки було те, що тато ВВ, ВВ/548, нарешті дізнався, коли вони зможуть повернутися додому — у майбутньому часі їм залишалося перебувати 6 днів. Діти дуже зраділи цій звістці, бо вже скучили за сім'єю, хоча й важко було звикнути до думки, що рідні навіть не встигнуть помітити їхньої відсутності. Натомість ВВ ця новина настрою не додала, тому він відразу зосередив увагу на роботі:

— Навряд чи ви встигнете пройти все! Вам потрібно наполегливо працювати, бо далі йтимуть складніші теми! — розпочав чергову розповідь ВВ.

— Ми будемо старатися, — пообіцяв Михась.

Минулого разу ви дізналися про застосування операцій для створення в коді аналогів математичних рівнянь. Сьогодні ви займатиметесь більше логікою, ніж математикою. Логіка є основою процесу прийняття рішень. Ми приймаємо різноманітні рішення щодня: позитивні (якось я вирішив зателефонувати на радіостанцію — та виграв квитки на концерт групи «Будь-що») та негативні (потім я надумав випрати джинси: вкинув їх до пральної машини, а після цього згадав, що забув вийняти ці квитки з кишені).



Приймати рішення доводиться всім. Людям у цьому плані пощастило, адже коли вони приймають рішення, у них є багато варіантів вибору. Наприклад, яке морозиво купити, яку музику слухати або з ким спілкуватися. Комп'ютерним програмам пощастило менше: вони мусять «приймати рішення», виходячи з істинності (`True`) або хибності (`False`) чогось. Прийняття рішень для них — це велика гра у відгадування.

Наведу приклад. Уявіть, що ви намагаєтеся замовити в кафе гамбургер, смажену картоплю, безалкогольне пиво, а офіціант задає вам запитання, на які можна відповісти лише «так» чи «ні».

- Ви будете щось пити? (Так).
- Ви будете каву? (Ні).
- Ви будете чай? (Ні).
- Ви будете содову? (Ні).
- Ви будете колу? (Ні).
- Ви будете безалкогольне пиво? (Так).

Було поставлено шість запитань, а ви ще не замовили гамбургер та смажену картоплю! Просуваєтеся повільно, але врешті-решт ви зробите замовлення! Приблизно так само відбувається прийняття рішення комп'ютерною програмою. Лише «так» чи «ні» (`True/False`). Такий тип логіки називають *булевою*.

— А ще, здається, є цілий підрозділ математики, який розглядає поняття логіки, зокрема, булевої, — перебив Михась розповідь ВВ.

— Саме так, — відповів ВВ. — Ми теж розглянемо деякі поняття цієї науки.

Логічні задачі

Комп'ютерні програми вправно розв'язують математичні задачі та задачі з булевої логіки. У таких задачах відповіддю завжди буде «істинно» (`True`) або «хибно» (`False`). Для цієї

логіки не існує ніяких «майже», «ніби», «здається». Подивіться на такі приклади з булевої логіки.

4 = 4? Так, істинно.

3 = 1 + 3? Ні, хибно.

3 = (6 + 12)/(1 + 5)? Так, істинно.

Бачите, в кожній із цих логічних задач відповіддю є «істинно» чи «хибно». Відповіддю не є числа, кольори або текст. Так комп'ютерна програма приймає рішення. Якщо відповідь «істинно», програма виконує одну дію, якщо відповідь «хибно» — іншу або не виконує жодної.

Операції булевої логіки

Як і в математиці, в булевій логіці є свої операції. Але вони виконуються не над числами, а лише над булевими операндами «істина» (`True`) та «хибність» (`False`). Їх можна застосовувати для поєднання логічних тверджень, щоб створювати складніші *логічні,* або *булеві вирази.*

Існує три основних булевих операції: `AND`, `OR` та `NOT`. Операції `AND` та `OR` виконуються над двома операндами, а операція `NOT` — над одним. Усі три дають булевий результат. Кожна операція обчислює результат за певними правилами. Щоб зрозуміти їх, розгляньте булевий вираз.

```
3 = 1 + 2 AND 2 = 1 + 1
```

Результат `True`.

Маємо два окремих логічних вирази, що поєднані операцією `AND`. Спочатку обчислюється значення першого (3 = 1 + 2), яке дорівнює `True`, а потім — другого (2 = 1 + 1), що також становить `True`. Після цього обчислюється вираз `True AND True`, значенням якого також є `True` в силу дії правила, за яким виконується операція `AND`: якщо обидва операнди операції `AND` мають значення `True`, її результатом також буде `True`. Подивіться на ще один приклад.

```
3 = 1 + 2 AND 5 = 2 + 2
```

Результат `False`.

Знову маємо два окремих логічних вирази, що поєднані операцією AND. Спочатку визначається результат першого виразу (3 = 1 + 2), який становить True, потім — другого (5 = 2 + 2), який становить False. Результатом операції True AND False є False в силу іншого правила виконання операції AND: якщо хоча б один операнд операції AND становить False, її результатом буде False.

Останній приклад.

```
3 = 1 + 2 OR 5 = 2 + 2
```

Результат True.

Спочатку обчислюється значення виразу 3 = 1 + 2, яке становить True, а потім виразу 5 = 2 + 2, яке становить False. Значенням виразу True OR False є True, згідно з правилом виконання операції OR: якщо хоча б один операнд операції OR становить True, її результатом буде True.

Щоб краще засвоїти булеві операції, вам потрібно попрацювати з ними. Усі правила виконання операції AND можна звести до простої таблиці.

Лівий операнд	Правий операнд	Лівий операнд AND Правий операнд
True	True	True
True	False	False
False	True	False
False	False	False

Так само ми можемо дослідити й усі можливі комбінації значень операндів операції OR.

Лівий операнд	Правий операнд	Лівий операнд OR Правий операнд
True	True	True
True	False	True
False	True	True
False	False	False

Операція Not не поєднує двох виразів. Вона просто перетворює значення True на False, а False на True.

Операнд	Not Операнд
True	False
False	True

Ось два приклади використання операції Not.

```
Not(3 = 1 + 2)
```

Результат False: 3 = 1 + 2 становить True, а Not True становить False.

```
Not(5 = 2 + 2)
```

Результат True: 5 = 2 + 2 становить False, а Not False становить True.

Операції порівняння

Пригадуєте, ми говорили, що програми вправно розв'язують математичні задачі, але не можуть їх формулювати? Те саме й з логічними задачами. Комп'ютерні програми чудово вміють розв'язувати дуже складні задачі з булевої логіки, але вони не можуть їх створювати. Це, знову ж таки, завдання програміста. Коли виконується код, програма «розв'язує» задачу. Відповіддю задачі з булевої логіки буде True або False. Залежно від неї програма «приймає рішення». Символ «=» можна використовувати для позначення операції присвоєння, а також застосовувати як операцію порівняння. У прикладах, що були наведені вище, за допомогою символу «=» формулюється питання «Істинно чи хибно?». Існують також інші операції порівняння. Вони використовуються для визначення того, більший (>) чи менший (<) лівий операнд ніж правий. У кожному випадку за допомогою таких операцій формулюється питання, відповіддю на яке буде True або False. Розгляньмо приклади.

```
3 < 4
```

Питання: Чи 3 менше 4?

Так, відповідь True.

```
2 > 5
```

Питання: Чи 2 більше 5?

Ні, відповідь `False`.

У таблиці наведено перелік операторів порівняння, які використовують найчастіше.

Оператор	Назва	Запитання
=	Дорівнює	Лівий операнд дорівнює правому?
>	Більше	Лівий операнд більший за правий?
<	Менше	Лівий операнд менший за правий?
>=	Більше або дорівнює	Лівий операнд більший або дорівнює правому?
<=	Менше або дорівнює	Лівий операнд менший або дорівнює правому?
<>	Не дорівнює	Лівий операнд не дорівнює правому?

Можливо, вам не знайомі операції >=, <= та <>. Це дві операції в одній, які поєднують два булевих вирази операцією `OR`. Якщо значення принаймні одного виразу дорівнює `True`, то значення всього виразу становитиме `True`. Наприклад, >= означає: (лівий операнд більший за правий?) `OR` (лівий операнд дорівнює правому?). Два окремих питання з'єднані операцією `OR`. Операція <> означає: (лівий операнд менший за правий?) `OR` (лівий операнд більший за правий?).

Розглянемо кілька прикладів із використанням операцій порівняння для створення булевих виразів. Можливо, вам доведеться звертатися до таблиць, щоб побачити, як працює операція `OR` у кожному випадку.

```
3 <= 3
```

3 < 3 дає `False`, а 3 = 3 дає `True`. `False OR True = True`.

Результат `True`, 3 менше або дорівнює 3.

```
2 >= 3
```

2 > 3 дає False і 2 = 3 дає False. False OR False = False. Результат False, 2 не більше і не дорівнює 3.

```
3 <> 4
```

3 < 4 дає True, а 3 > 4 дає False. True OR False = True. Результат True, 3 не дорівнює 4.

```
3 <> 3
```

3 < 3 дає False і 3 > 3 дає False. False OR False = False. Результат False, 3 не дорівнює 3; отже, 3 дорівнює 3.

Приклади, які ми розглядали, стосуються чисел. Але ви можете порівнювати рядки тексту або значення інших типів, зокрема булевого.

Розглянемо такі приклади.

```
"ABC" = "DEF"
```

Чи "ABC" = "DEF"? Ні, результат False.

```
True = True
```

Чи True = True? Так, результат True.

```
True = False
```

Чи True = False? Ні, результат False.

Ви можете застосовувати операції порівняння для порівняння рядків так само, як використовуєте їх для порівняння чисел. Меншим вважається той рядок, який в алфавітному переліку слів розміщувався б вище.

```
"ABC" < "DEF"
```

Чи "ABC" < "DEF"? Так, результат True.

```
"DEF" >= "ABC"
```

Чи "DEF" >= "ABC"? Так, результат True.

Моделювання прийняття рішень

Тепер за допомогою операцій порівняння та булевої логіки ви можете зробити так, щоб ваша програма ставила запитання,

які припускають відповідь `True` або `False`. Отже, що програма буде робити з відповіддю? Вона «прийме рішення». Виходячи з того, якою була відповідь: `True` чи `False`, програма виконуватиме одну операцію чи іншу. Часто програма не має вибору: якщо відповідь — `True`, виконується одна операція; якщо відповідь `False`, то нічого не виконується. Далі наведено простий приклад такої поведінки.

- Якщо я помию мамину машину перш ніж тато повернеться додому, він заплатить мені 5 кредитів.
- Тато повернувся додому. Я помив машину? Так. Тато платить мені 5 кредитів.
- Тато повернувся додому. Я помив машину? Ні. Тато не платить мені 5 кредитів.

У схожий спосіб прийняття рішень моделюється у програмі.

- Якщо прапорець **Великий розмір** установлено, буде виведено зображення великого розміру.
- Прапорець **Великий розмір** установлено? Так. Виводиться зображення великого розміру.
- Прапорець **Великий розмір** установлено? Ні. Не буде виведено зображення великого розміру.

Наведемо інший приклад.

- Якщо змінна `TotalLiters` більше нуля, обчислюємо вартість пального.
- Змінна `TotalLiters` > 0? Так. Обчислюємо вартість пального.
- Змінна `TotalLiters` > 0? Ні. Не обчислюємо вартість пального.

У прикладах із реального життя є спільні риси із прикладами з моїх програм. У них використовується булева логіка для

формулювання питань із відповідями True або False. Операції порівняння також «задають» питання з відповідями True або False. Якщо відповідь на таке питання буде True, виконуються певні дії; якщо False — ні.

Умовний оператор

Ви, мабуть, помітили, що я в прикладах застосовую слова «якщо» (If) та «тоді» (Then)? Якщо певна умова виконується (True), тоді щось відбувається; а якщо умова не виконується (False), нічого не відбувається. Коли ви хочете запрограмувати процес прийняття рішення, то також використовуєте оператор If...Then. У Visual Basic .NET його синтаксис має такий вигляд:

```
If умова Then
   оператор
End If
```

Ось приклад коду Visual Basic .NET:

```
If myAge = 3 Then
   MessageBox.Show("Мені 3 роки.")
End If
```

Зверніть увагу на слово If. Це ключове слово мови Visual Basic .NET, і тому середовище Visual Studio виділяє його в тексті програми синім кольором. Умова зазначається відразу після слова If. У ній формулюється питання, що припускає відповідь True чи False: «чи змінна myAge дорівнює 3?» Якщо твердження істинне, треба щось зробити. Зауважте, що слово Then записане після умови. Це також ключове слово Visual Basic .NET, яке в тексті програми виділяється синім кольором. Після цього слова записується оператор, що виконуватиметься в разі істинності умовного твердження. Тепер перед цим оператором ми зробили відступ, щоб легше було читати код. Якщо умова виконується, програма виводить на екран вікно повідомлення з текстом "Мені 3 роки. ". Зауважте, що після оператора виведення повідомлення стоять слова End If. Це ключові слова, якими закінчується оператор If.

Тепер ваша черга! Напишіть нову Windows-програму, що матиме назву IfThen. Додайте до форми Form1 текстове поле і кнопку. Двічі клацніть кнопку, щоб змінити код обробника події клацання. Додайте такі рядки коду:

```
Dim myAge As Integer
myAge = 3
If myAge = 3 Then
   TextBox1.Text = "Мені 3 роки."
End If
```

Побудуйте та запустіть проект. Клацніть кнопку. У коді оголошується myAge, якій присвоюється значення 3. В операторі If формулюється питання, що припускає відповідь True або False: чи myAge = 3? Відповіддю буде True, бо ви щойно надали відповідне значення змінній myAge: myAge = 3. Оскільки відповідь на питання — True, виконується оператор, записаний після слова Then. У текстовому полі буде виведено текст "Мені 3 роки.".

Тепер змінимо оператор присвоєння, надавши змінній myAge значення 2:

```
myAge = 2
```

Побудуйте та запустіть проект знову. Клацніть кнопку. Що відбувається? Цього разу змінній myAge присвоїли значення 2. В операторі If міститься питання: чи myAge дорівнює трьом? Цього разу відповідь — False, тому оператор після слова Then не виконується. Нічого не відбувається. Текст у текстовому полі не змінився, бо умова є хибною.

Розглянемо інший приклад. До форми Form1 додайте прапорець. Змініть код обробника події клацання кнопки. Замініть наявний код таким:

```
If CheckBox1.Checked = True Then
   TextBox1.Text = "Мене встановлено."
End If
```

Побудуйте та запустіть проект. Установіть прапорець та клацніть кнопку. Умова CheckBox1.Checked = True є істинною (бо ви встановили прапорець), тому в текстовому полі відобразиться текст "Мене встановлено.".

Побудуйте та запустіть проект знову. Цього разу зніміть прапорець та клацніть кнопку. Що сталося? Умова є хибною (тому що ви зняли прапорець), тому текст залишився без змін. Ваша програма «вирішувала», чи змінювати значення атрибута `Text`, спираючись на результат перевірки умови (встановлено прапорець чи ні).

Вам слід виділяти відступами всі оператори між рядками `If...Then` та `End If` — тоді код буде легше читати та розуміти. Навіть якщо такий оператор один, його слід виділити. За стандартних настройок Visual Studio .NET створює відступи автоматично.

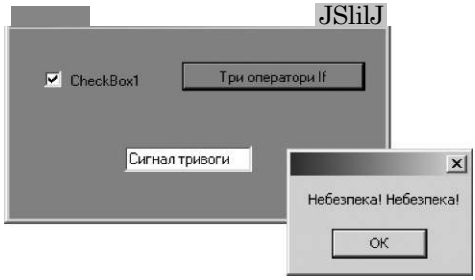
Застосування кількох умовних операторів

Імовірно, вам знадобиться, щоб під час виконання програми було прийнято кілька рішень щодо подальших дій. Наприклад, вам потрібно, перевіривши стан прапорця, а після цього перемикача, порівняти два числа та визначити, чи користувач залишив текстове поле порожнім. Багато ви хочете від програми! Чи може програма все це зробити? На щастя, може. Ви можете додати стільки операторів `If...Then`, скільки потрібно. Як і будь-які інші оператори, вони виконуються в тому порядку, в якому розташовані в коді. Напишемо програму, в якій використовуватиметься більше ніж один оператор `If...Then`. Припустімо, що в нашій програмі є прапорець **Відображати всі аварійні сигнали.** Якщо його встановлено, програма змінює фоновий колір форми на червоний та текст у полі на «Сигнал тривоги», а потім відображає вікно повідомлення з текстом «Небезпека! Небезпека!». Один із варіантів такої програми передбачає використання трьох окремих операторів `If...Then`. В умові кожного з них перевірятиметься, чи встановлено прапорець **Відображати всі аварійні сигнали.** Якщо умова є істинною, буде виконано певний код.

Створіть новий проект і назвіть його FireAlarm (Пожежна тривога). Додайте прапорець, текстове поле та командну кнопку до форми `Form1`. Змініть значення атрибута `Text` кнопки `Button1` на `Три оператори If`. Двічі клацніть кнопку `Button1`, щоб змінити обробник події клацання цієї кнопки. Додайте такий код.

```
If CheckBox1.Checked = True Then
   Form.ActiveForm.BackColor = System.Drawing.Color.Red
End If
If CheckBox1.Checked = True Then
   TextBox1.Text = "Сигнал тривоги"
End If
If CheckBox1.Checked = True Then
   MessageBox.Show("Небезпека! Небезпека!")
End If
```

Побудуйте та запустіть проект. Установіть прапорець і клацніть кнопку **Три оператори If.** По черзі буде виконано всі оператори `If...Then`, а всі умови будуть істинними. Під час виконання першого оператора `If...Then` колір форми буде змінено на червоний; під час виконання другого — текст у полі буде змінено на «Сигнал тривоги»; і нарешті третій оператор виведе вікно повідомлення з текстом «Небезпека! Небезпека!».



Готуючись до від'їзду друзів, ВВ вирішив розробити програму, за допомогою якої він міг би обрати для них подарунки. Допоможіть йому в цьому.

Вправа 8.1. Подарунки

Напишіть програму вибору подарунків. Користувач буде обирати подарунок зі списку, після чого у списку поруч пропонуватиметься кілька ідей, пов'язаних з обраним подарунком. Файли, потрібні для виконання цієї вправи, містяться в папці **Вправа_8.1.**

1. Створіть форму та назвіть її `Подарунки`. Додайте до неї написи `Основний подарунок` та `Супутній подарунок`.
2. Помістіть до форми поле зі списком (згадуємо день 4). Надайте його атрибуту `Text` значення `Оберіть подарунок`. Заповніть колекцію `Items` подарунками, список яких наведено нижче, або скористайтеся власними ідеями щодо подарунків.
3. Додайте до форми текстове поле. Зробіть його багаторядковим.
4. Зробіть так, щоб після того, як подарунок буде обрано, в полі зі списком текстове поле заповнювалося назвами принаймні двох інших речей, що доповнюватимуть основний подарунок. Ви можете використати наведені нижче варіанти або вигадати власні.

Основний одарунок	Супутні подарунки
Капелюх	Стрічка для капелюха Краватка
Черевики	Крем для черевиків
Опудало шакала	Карта таємної шахти, де сховано золото Справжня золотоносна руда
Листівки	Ручка з написом Футболка
Перець чілі	Соус чілі Гострий соус «Торнадо»

Значення атрибута `SelectedIndex` дорівнює номеру обраної речі (починаючи з нуля).

Атрибут `SelectedItem` містить назву обраної речі.

За потреби відображати фрази в окремих рядках текстового поля вставте між ними спеціальну константу `vbNewLine`:

```
TextBox1.Text = "A" & vbNewLine & "B"
```

Звичайно, `A` та `B` слід замінити назвами подарунків.

Код, вкладений в умовний оператор

Ви помітили, що у програмі FireAlarm для всіх операторів `If...Then` використовувалася одна умова? А саме:

```
If CheckBox1.Checked = True Then
```

Різними були лише оператори всередині конструкції If... End If. Чи не забагато повторень? Може є простіший спосіб запису того самого коду? Звичайно, є! За допомогою Visual Basic .NET і більшості сучасних мов програмування можна записувати довільну кількість будь-яких операторів усередині оператора If...End If. Для цього використовують такий синтаксис:

```
If умова Then
   оператор 1
   оператор 2
   оператор N
End If
```

Знаючи це, ми можемо переписати код, що використовувався у програмі FireAlarm. Спростимо його шляхом видалення операторів, які повторюються. Відкрийте програму FireAlarm. Додайте другу командну кнопку до форми Form1. Змініть значення атрибута Text цієї кнопки на Один оператор If і двічі клацніть її, щоб відредагувати обробник події клацання. Додайте такий код:

```
If CheckBox1.Checked = True Then
   Form.ActiveForm.BackColor = System.Drawing.Color.Red
   TextBox1.Text = "Сигнал тривоги"
   MessageBox.Show("Небезпека! Небезпека!")
End If
```

Побудуйте та запустіть проект. Установіть прапорець і клацніть кнопку **Один оператор If.** Перевіряється одна умова. Вона є істинною, тому всі оператори всередині If...End If виконуються так само, як і раніше, по черзі. Програма працює, як і раніше, проте виконання трьох операторів залежить від істинності лише однієї умови! Використовуючи одну умову між словами If та Then із кількома операторами між Then та End If можна скоротити обсяг коду та зробити його зрозумілішим.

Вибір із кількох операторів

Пригадуєте приклад про замовлення безалкогольного пива, коли необхідно було відповідати на питання лише «так» або «ні»? Офіціант задавав багато питань із двома варіантами

відповідей замість того, щоб задати одне питання з кількома варіантами відповідей. Хоча й знадобився певний час, але замовлення зрештою було зроблено. Ви також можете сформулювати питання з кількома варіантами відповідей у коді, використовуючи низку операторів If...Then. На підтвердження цього в мене є чудовий приклад.

Створіть новий проект, що називатиметься ChooseAColor (Вибір кольору). Додайте три перемикачі та командну кнопку до форми Form1. Використайте вікно **Properties** (Властивості), щоб задати значення атрибутів елементів керування. Змініть значення атрибута Text перемикача RadioButton1 на Червоний, значення атрибута Text перемикача RadioButton2 — на Жовтий, значення атрибута Text перемикача RadioButton3 — на Зелений, значення атрибута Text кнопки Button1 — на Зробити кольоровою. Атрибуту Checked перемикача RadioButton! присвойте значення True. Потому двічі клацніть кнопку Button1, щоб змінити код обробника події клацання цієї кнопки. Введіть такі рядки коду:

```
Dim myColor As System.Drawing.Color
If RadioButton1.Checked = True Then
   myColor = System.Drawing.Color.Red
End If
If RadioButton2.Checked = True Then
   myColor = System.Drawing.Color.Yellow
End If
If RadioButton3.Checked = True Then
   myColor = System.Drawing.Color.Green
End If
Form.ActiveForm.BackColor = myColor
```

Побудуйте та запустіть проект. Зауважте, що у групі перемикачів обрано перемикач **Червоний.** Клацніть кнопку **Зробити кольоровою.** Форму та кнопки буде пофарбовано червоним. Установіть інший перемикач та клацніть кнопку **Зробити кольоровою.** Колір форми та кнопок буде змінено! Погляньте на код. За допомогою операторів If...Then можна відповісти на питання, що має кілька варіантів відповіді: «Який перемикач установлено?». Послідовно виконуючи код, програма врешті-решт визначає, який із трьох перемикачів установлено,

а потім здійснює відповідне присвоєння, щоб змінити колір форми та елементів керування.

Застосування логічних операторів

Ми вже говорили про те, як можна використовувати булеві операції для з'єднання виразів булевої логіки. Погляньмо на приклад, в якому застосовується булева операція AND для створення складної умови. Ми змінимо програму ChooseAColor, додавши прапорець, що активуватиме та дезактивуватиме функцію змінення кольору. Для створення складнішої умови, що враховує стан прапорця, коли приймається рішення про необхідність змінення кольору, потрібно застосувати булеву операцію AND.

Відкрийте програму ChooseAColor. Додайте до форми Form1 прапорець. Змініть значення його атрибута Text на Активувати кольори. Замініть код обробника події клацання кнопки Button1 таким кодом:

```
Dim myColor As System.Drawing.Color
If RadioButton1.Checked = True AND CheckBox1.Checked Then
   myColor = System.Drawing.Color.Red
End If
If RadioButton2.Checked = True AND CheckBox1.Checked Then
   myColor = System.Drawing.Color.Yellow
End If
If RadioButton3.Checked = True AND CheckBox1.Checked Then
   myColor = System.Drawing.Color.Green
End If
Form.ActiveForm.BackColor = myColor
```

Створіть та запустіть проект. Клацніть кнопку **Зробити кольоровою.** Нічого не відбувається! Колір не змінився. Тепер установіть прапорець **Активувати кольори** та знову клацніть кнопку **Зробити кольоровою.** Форма та елементи керування зробляться червоними. Чому це відбувається? До умови в кожному виразі з оператором If...Then ми додали такий код:

```
AND CheckBox1.Checked
```

Цей код еквівалентний такому:

```
AND CheckBox1.Checked = True
```

Так вводиться додаткова умова. Тепер щоб колір змінився, обидві умови мають бути істинними. Наприклад:

```
RadioButton2.Checked = True AND CheckBox1.Checked = True
```

Пригадуєте правила виконання операції AND? Обидві умови мають бути істинними для того, щоб істинним став увесь вираз. Якщо хоча б одна з умов, об'єднаних оператором AND, є хибною, весь вираз має значення False, а колір форми не змінюється.

Пам'ятайте, операція AND використовується тоді, коли істинні дві умови. Якщо для виконання певної дії достатньо, щоб лише одна із двох умов була істинною, поєднуйте їх операцією OR.

— ВВ, може досить? — запитав Михась, коли помітив, що той ще не збирається завершувати свою розповідь. — Ми ж не зможемо запам'ятати такий обсяг інформації!

— А інакше ви не встигнете все вивчити! — заперечив ВВ.

— А що краще: пройти все і нічого не засвоїти, чи запам'ятовувати по частинах? — запитала Даринка.

— Ну що ж, ви мене переконали. Тоді, мабуть, на сьогодні досить. Але, для того щоб закріпити вивчений матеріал, виконайте кілька вправ. А завтра продовжимо розглядати цю саму тему, але дещо в іншому аспекті, — відповів ВВ, а потім додав: — ви ще не були в нашому парку атракціонів, тому пропоную піти туди.

— Давайте швидше виконувати вправи! — з радістю вигукнув Михась.

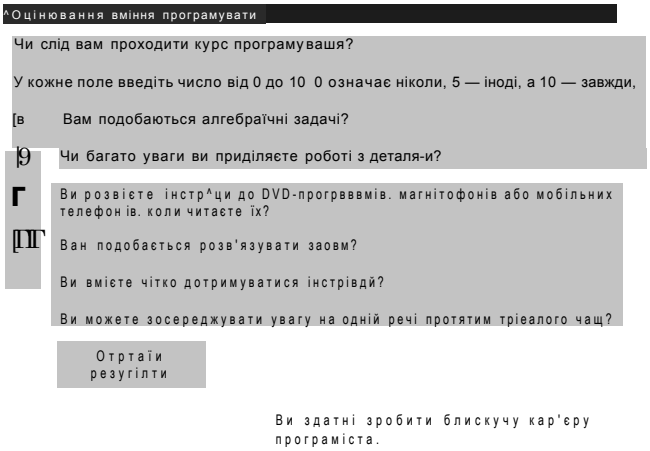
— А вони сьогодні дуже цікаві, — зауважив ВВ. — Ми розробимо програму-тест, яка визначить рівень ваших знань!

Вправа 8.2. Тест для вибору майбутньої професії

Створіть інтерактивну форму оцінювання. Користувачі матимуть змогу оцінювати рівень знань з програмування за допомогою 10-бальної шкали. Програма додасть усі бали та виведе оцінку за вміння програмувати. Файли, необхідні для виконання цієї вправи, містяться в папці **Вправа_8.2.**

Коли користувач клацає кнопку **Оцінити результати,** додаються числа, що містяться в шести текстових полях. Оскільки вміст текстового поля є рядком, а не числом, для виконання операції додавання вам доведеться перетворити рядок на число. У Visual Basic .NET зробити це можна кількома способами. Ось один із них.

```
X = Val(TextBox1.Text)
```



Функція `Val` перетворює значення, що записане як текстовий рядок, на число. Ви можете надати це значення змінній, наприклад, змінній `x`. Потім, аналізуючи значення змінної `x` за допомогою операторів `If...Then`, визначте, яке повідомлення виводитиметься в текстове поле:

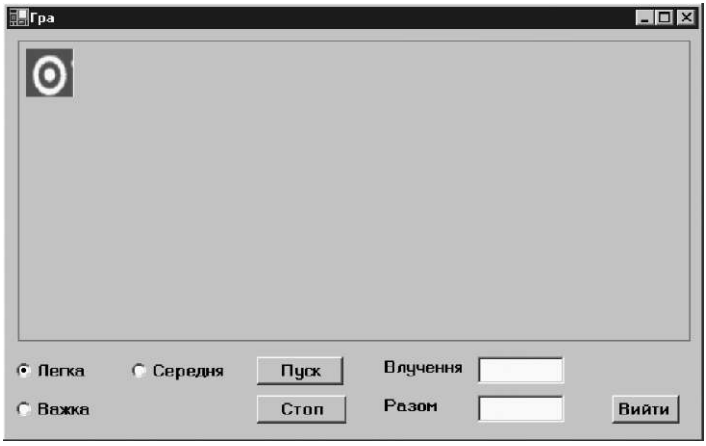
- Якщо `x` більше або дорівнює 40, виведіть у текстове поле повідомлення про те, що користувач здатен зробити блискучу кар'єру програміста.
- Якщо `x` належить діапазону від 20 до 39, порадьте користувачу пройти курс програмування.
- Якщо `x` менше 20, порадьте користувачу знайти іншу професію.

А скільки балів набрали ви?

Перед подорожжю до парку атракціонів Михась розповів ВВ про одну просту та цікаву гру, що дуже поширена в їхньому часі. І для того щоб пояснити в чому полягає її зміст, він вирішив написати програму, яка б ілюструвала гру. У цій програмі використовуватимуться конструкції, які ще не було розглянуто, тому уважно прочитайте вказівки, що наводяться нижче.

Завдання 8. Влучний постріл

Створіть гру, зміст якої полягатиме в тому, щоб клацнути кнопкою миші мішень, що рухається. Файли, необхідні для виконання цього завдання, містяться в папці **Завдання_8.** Змініть висоту форми `Form1` на 300, а ширину — на 600 пікселів. Надайте формі ім'я **Гра,** а потім додайте до неї елементи керування, щоб вона набула такого вигляду, як на рисунку. Зокрема додайте поле зображення, атрибуту `Image` якого присвойте ім'я файлу зображення мішені **001_Challenge_Target.gif** з папки **Завдання_8.** Атрибуту `SizeMode` поля зображення надайте значення `StretchImage`, щоб розмір мішені дорівнював розміру поля.



Мета гри — клацнути мішень, що рухається. Код забезпечуватиме переміщення зображення. Гра починається з клацання

кнопки **Пуск,** а завершується клацанням кнопки **Стоп.** За допомогою кнопки **Вийти** можна закрити вікно програми. Три перемикачі використовуються для визначення рівня складності гри, а поле малюнка містить зображення мішені.
У цій програмі застосовується таймер, для того щоб код виконувався через певні проміжки часу. Намалювавши форму, знайдіть у вікні **Toolbox** (Панель інструментів) елемент керування **Timer** (Таймер) та двічі клацніть його. Він відобразиться під формою. Таймер є частиною форми, але користувачам його не видно.

Настроївши атрибути форми та елементів керування, переходьте до введення коду.

1. Оголосіть змінну `Hits` типу `Integer`, в якій зберігатиметься кількість влучень, під таким рядком коду:

   ```
   Public Class Form1
   ```

2. Під оголошенням змінної `Hits` оголосіть змінну `Total` типу `Integer`, в якій зберігатиметься загальна кількість спроб. Спробою вважатимемо кожне пересування мішені.
3. До обробника події клацання кнопки `Пуск` додайте код, який забезпечуватиме виконання таких дій:
   - надання нульового значення лічильникам `Hits` (кількість влучень) та `Total` (загальна кількість спроб);
   - надання параметру `Enabled` таймера `Timer1` значення `True`.
4. До обробника події клацання кнопки `Пуск` додайте код, який забезпечуватиме виконання таких дій:
   - якщо установлено перемикач `RadioButton1`, параметру `Interval` таймера надається значення 900, а ширині та висоті зображення — 40;
   - якщо установлено перемикач `RadioButton2`, параметру `Interval` таймера надається значення 850, а ширині та висоті зображення — 35;
   - якщо установлено перемикач `RadioButton3`, параметру `Interval` таймера надається значення 800, а ширині та висоті зображення — 30.

Стан перемикачів RadioButton1-RadioButton3 аналізуватимуть три схожих фрагменти коду. Щоб не вводити однотипний код тричі, скористайтеся буфером обміну.

5. У коді обробника події клацання кнопки Стоп параметру таймера Enabled присвойте значення False.
6. У коді обробника події клацання кнопки Вийти забезпечте завершення програми за допомогою команди End.
7. До обробника події клацання поля зображення додайте код, що забезпечуватиме додавання одиниці до лічильника Hits та присвоєння значення цього лічильника текстовому полю Влучення.
8. До обробника події таймера (двічі клацніть його, щоб дістатися коду), додайте код, що пересуває мішень.

```
Dim MyRandomGenerator As System.Random
MyRandomGenerator = New System.Random
Dim RandomX As Integer
Dim RandomY As Integer

RandomX = MyRandomGenerator.Next(1, 550)
RandomY = MyRandomGenerator.Next(1, 165)
PictureBox1.SetBounds(RandomX, RandomY, Me.Width, Me.Height,
System.Windows.Forms.BoundsSpecified.Location)
```

9. В обробнику події таймера після коду, що було наведено вище, до загальної кількості спроб додайте 1 та виведіть результат у полі Разом.
10. Двічі клацніть у полі зображення, щоб відредагувати обробник події Onclick, пов'язаної з цим полем. У коді обробника до лічильника Hits додайте 1.

Додаткове завдання

Зробіть так, щоб спочатку фоновий колір форми був сірим, а у разі влучення в мішень він ставав червоним.

— Все, ми виконали всі завдання! — зрадів Михась.

— От і добре. Тепер ми можемо йти в парк атракціонів, але за однієї умови, — сказав ВВ.

— Якої? — здивовано запитала Даринка.

— Якщо ви відкриєте кодовий замок, — посміхнувся ВВ.

Кодовий замок

1. За якою схемою «приймають рішення» комп'ютери?
   а) істинно/хибно;
   б) ймовірно/неймовірно;
   в) комп'ютери не приймають рішень, оскільки їхні дії запрограмовані.
2. У чому полягає зміст операції, позначеної символами <>?
   а) піднесення до степеня;
   б) не дорівнює;
   в) менше або дорівнює.
3. Коли операція `AND` повертає значення `True`?
   а) коли обидва її операнди мають значення `True`;
   б) коли принаймні один з її операндів має значення `True`;
   в) відповідь неоднозначна.
4. Коли операція `OR` повертає значення `True`?
   а) коли обидва її операнди мають значення `True`;
   б) коли принаймні один з її операндів має значення `True`;
   в) відповідь неоднозначна.
5. Який буде результат операції `a>a`?
   а) `True`;
   б) `False`;
   в) результат залежить від значення змінної `a`.
6. Таблиця значень якої операції має найбільше значень `False`?
   а) `AND`;
   б) `OR`;
   в) `NOT`.
7. Який із наведених нижче виразів не еквівалентний виразу `x<>y`?
   а) `x>y AND x<y`;
   б) `x>y OR x<y`;
   в) `NOT(x=y)`.

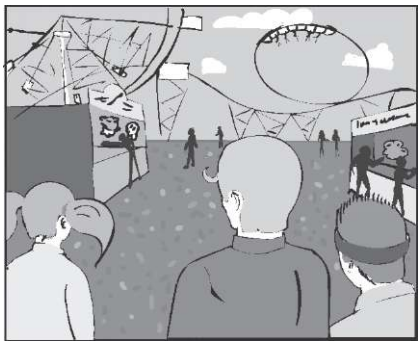
8. Умова `a OR NOT (a)` :

а) є завжди істинною;

б) є завжди хибною;

в) може бути як істинною, так і хибною, залежно від значення змінної `a`.

День 9
Варіативність — основа інтелекту



Зранку, як і домовлялися, друзі зібралися, щоб продовжити вчорашню розмову. Атракціони, які вони відвідали, справді виявилися цікавими та захоплюючими. Більшість із них були такими самими, як і в тому часі, з якого прибули Михась і Даринка, проте траплялися й такі конструкції, які Михась навіть уявити не міг!

Після ввімкнення ноутбуків ВВ розпочав свою розповідь. Отже, вчора я розказував вам про оператори `If...Then`, які ще називають умовними операторами. Тепер наші програми можуть приймати рішення! Вони майже навчилися міркувати! Ми вже говорили про те, як застосовувати операції порівняння, працювати з булевою логікою та використовувати в коді Visual Basic .NET оператори `If...Then`. Сьогодні я покажу вам кілька прийомів, за допомогою яких можна зробити оператори `If...Then` ще кориснішими.

Вкладання умовних операторів

Вам уже відомо, як застосовувати оператор `If...Then` у Visual Basic .NET. Завдяки цьому оператору ви можете розробити достатньо складні програми. А якщо використати кілька операторів `If...Then`, програма зможе приймати цілу низку рішень

будь-якого рівня складності. Проте писати такий код іноді вкрай непросто, вже не кажучи про те, що це нудно. Тож я навчу вас кільком прийомам, за допомогою яких можна буде надати простому оператору `If...Then` потужні функціональні можливості. Спершу розглянемо, як вкладати оператори `If... Then` один в один, а потім навчимося писати код операторів `If...Then...Else`.

Вчора ми не встигли дізнатися про одну маленьку таємницю операторів `If...Then`: між словами `Then` та `End If` може записуватися інший оператор `If...Then`! Тільки подумайте! Другий оператор `If...Then` виконується лише тоді, коли умова першого оператора є істинною. Це називають *вкладанням,* тому що один оператор `If...Then` вкладається в інший. Слова `End If`, якими завершується внутрішній оператор, передуватимуть словам `End If` зовнішнього оператора. Розглянемо приклади з реального життя.

Якщо у мене є 200 кредитів, то, якщо магазин відкрито, я придбаю MP3-плеєр.

У псевдокоді це мало б такий вигляд:

```
If у мене є 200 кредитів Then
   If магазин відкрито Then
      Я придбаю MP3-плеєр
   End If
End If
```

Один оператор `If...Then` містить у собі другий. Перший оператор перевіряє умову, чи є у мене 200 кредитів, у разі істинності якої виконується другий оператор `If...Then`. Він містить власну умову (якщо магазин відкрито), у разі істинності якої приймається рішення «я придбаю MP3-плеєр». Отже, ви бачите, що в мене має бути 200 кредитів, а магазин має бути відкрито, інакше я не придбаю ніякого MP3-плеєра. Потрібно, щоб виконувались обидві умови. Якщо в мене не буде 200 кредитів, то умова першого

оператора If...Then не буде істинною, тому вже неважливо, чи відкрито магазин, чи ні — MP3-плеєр я все одно не придбаю! Розгляньмо ще один приклад коду. Напишіть нову програму та дайте їй назву Nesting (Вкладання). Додайте два прапорці та командну кнопку до форми Forml. Змініть значення атрибута Text кнопки Buttonl на Який прапорець установлено?. Двічі клацніть кнопку Buttonl і до обробника події клацання кнопки додайте такий код:

```
If CheckBoxl.Checked = True Then
   If CheckBox2.Checked = True Then
      MessageBox.Show("Усі прапорці встановлено.")
   End If
End If
```

Напишіть та запустіть програму. Встановіть перший прапорець і клацніть кнопку **Який прапорець установлено?** Нічого не відбудеться. Тепер установіть другий прапорець і знову клацніть ту саму кнопку. Цього разу відобразиться повідомлення «Усі прапорці встановлено». Як бачите, перший рядок коду містить оператор If...Then:

```
If CheckBoxl.Checked = True Then
```

У цей оператор вкладено інший оператор If...Then, який має власну умову:

```
If CheckBox2.Checked = True Then
```

У вкладений оператор If...Then, у свою чергу, вкладено оператор виведення вікна повідомлення:

```
MessageBox.ShowC^^ прапорці встановлено.")
```

Що ж відбувається? Якщо виконується перша умова (встановлено прапорець CheckBoxl), виконується й оператор перевірки стану прапорця CheckBox2. Оскільки цей оператор є другим оператором If...Then, то, якщо виконується друга умова (встановлено прапорець CheckBox2), виконується й оператор виведення вікна повідомлення. Отже, для того щоб відкрилося вікно повідомлення, мають виконуватися обидві умови. Тобто для виведення вікна повідомлення потрібно встановити обидва прапорці Checkboxl і Checkbox2.

Зауважте, що в коді, який було наведено вище, другий оператор If...Then цілком міститься в першому. Слова End If першого оператора If...Then розміщуються після слів End If другого оператора If...Then. Тобто другий оператор вкладено в перший. Ви можете користуватися вкладанням, щоб об'єднати кілька операторів If...Then. Тоді певна дія виконуватиметься лише тоді, коли буде виконано ряд умов.

У вкладені оператори If...Then, у свою чергу, можна вкладати інші оператори. При цьому вкладати їх можна на стільки рівнів, на скільки вам потрібно. Однак, якщо застосовувати більше трьох рівнів, код стає заплутаним і його важко налагоджувати. Програма стане зрозумілішою, якщо ви виділятимете вкладені твердження відступами.

Вибір з двох альтернатив

Досі наші умови були дещо однобічними: якщо умова виконується, виконується й певний оператор, а якщо умова не виконується, нічого не відбувається. А що нам робити, аби щось відбувалось, коли умова не виконується? Можливо, нам потрібно, щоб у цьому разі виконувалися певні рядки коду. Як цього досягти?

По-перше, можна спробувати застосувати ще одну конструкцію If...Then, що міститиме протилежну умову, а також оператор, що виконуватиметься в разі її істинності. Напишемо такий код! Створіть новий проект із назвою IfThenOtherwise (Якщо тоді у протилежному випадку). Додайте командну кнопку та прапорець до форми Forml. Змініть значення атрибута Text кнопки Buttonl на IfThen. Помістіть такі рядки коду до обробника події клацання кнопки Buttonl:

```
If CheckBoxl.Checked = True Then
   Form.ActiveForm.BackColor = System.Drawing.Color.Red
End If
If CheckBoxl.Checked = False Then
   Form.ActiveForm.BackColor = System.Drawing.Color.Blue
End If
```

Побудуйте та запустіть проект. Клацніть кнопку **IfThen.** В результаті форма набуде блакитного кольору. Встановіть прапорець та знову клацніть кнопку **IfThen.** Форма стане червоною.

Як працює такий код? У першому операторі If...Then перевіряється умова CheckBoxl. Checked=True. Якщо вона виконується, атрибуту BackColor форми присвоюється значення Red. У другому операторі перевіряється умова CheckBox1. Checked=False (протилежна умова). Якщо вона виконується, атрибуту BackColor форми присвоюється значення Blue. Код трохи повторюється, але спрацьовує! У ньому один недолік: доводиться писати майже ідентичний код для протилежної умови. Це забирає час та може призвести до виникнення помилок.

Оператор If...Then...Else

Visual Basic .NET пропонує альтернативу, яка дає можливість не повторювати код та робить його простішим і зручнішим для читання. Цією альтернативою є оператор If ...Then...Else. У ньому записується одна умова та два можливих варіанти дій за результатами її перевірки (коли умова істинна та коли хибна). Цей оператор використовується замість двох операторів If...Then із протилежними умовами. Розгляньмо його синтаксис:

```
If умова Then
   оператор1
Else
   оператор2
End If
```

Як працює такий код? Спочатку перевіряється умова. Якщо вона є істинною, виконується оператор1, а інакше (Else) — оператор2.

Тепер давайте переробимо програму IfThenOtherwise, замінивши в ній два незалежних і протилежних оператори If...Then одним оператором If...Then...Else. Додайте другу кнопку до форми Forml. Змініть значення її атрибута Text на IfThenElse. До коду клацання кнопки Button2 додайте таке:

```
If CheckBoxl.Checked = True Then
   Form.ActiveForm.BackColor = System.Drawing.Color.Red
Else
   Form.ActiveForm.BackColor = System.Drawing.Color.Blue
End If
```

Побудуйте та запустіть проект. Клацніть кнопку **IfThenElse.** Форма набуде блакитного кольору. Встановіть прапорець, а потім знову клацніть цю кнопку — форма стане червоною. Програма за допомогою оператора `If...Then...Else` робить те саме, що й тоді, коли використовувала два оператори `If...Then`. Код працює так: перевіряється умова `CheckBoxl.Checked=True;` якщо вона виконується, форма набуває червоного кольору; інакше (`Else`) виконується другий вкладений оператор (після слова `Else`), і форма стає блакитною.

Зверніть увагу на виділення умовних тверджень відступами після фраз `If...Then` та `Else`. Visual Studio .NET робить його автоматично, підтримуючи охайність і читабельність коду.

Розгляньмо ще приклад використання оператора `If...Then... Else`. Змінимо програму IfThenOtherwise так, щоб між словами `Then` та `Else` і `Else` та `End If` містилося кілька рядків коду. Відкрийте програму IfThenOtherwise. Замініть код обробника події клацання кнопки `Button2` таким кодом:

```
If CheckBox1.Checked = True Then
   Form.ActiveForm.BackColor = System.Drawing.Color.Red
   MessageBox.Show('^ червона. ")
   TextBox1.Text = "Я червона. "
Else
   Form.ActiveForm.BackColor = System.Drawing.Color.Blue
   MessageBox.Show('^ блакитна. ")
   TextBox1.Text = "Я блакитна."
End If
```

Побудуйте та запустіть проект. Клацніть кнопку **IfThenElse.** Відбудуться три події: форма і текст у полі набудуть червоного кольору, а також буде виведено повідомлення «Я червона». Тепер установіть прапорець і знову клацніть кнопку **IfThenElse.** Відбудуться три інші події.

Покрокове виконання програм з умовними операторами

Тепер скористаємося налагоджувачем Visual Studio .NET для виконання оператора `If...Then...Else` в покроковому режимі.

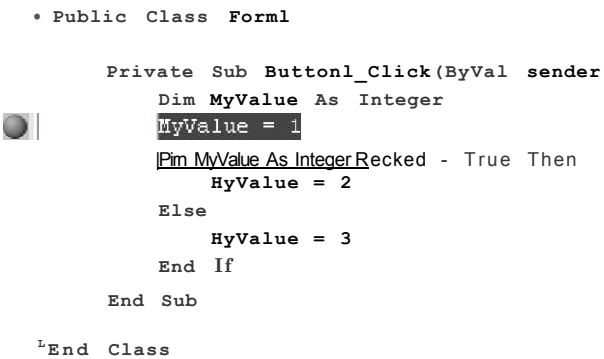
Так ви зможете спостерігати за його роботою «зсередини», з точки зору коду.

Створіть новий проект і назвіть його StepInlf (Покрокове виконання If). Додайте кнопку та прапорець до форми Forml. Присвойте атрибуту Checked прапорця значення True. Коли ви запустите програму, прапорець буде встановлено.

Двічі клацніть кнопку, щоб змінити код обробника події клацання. Додайте такі рядки коду:

```
Dim MyValue As Integer
MyValue = 1
If CheckBoxl.Checked = True Then
   MyValue = 2
Else
   MyValue = 3
End If
```

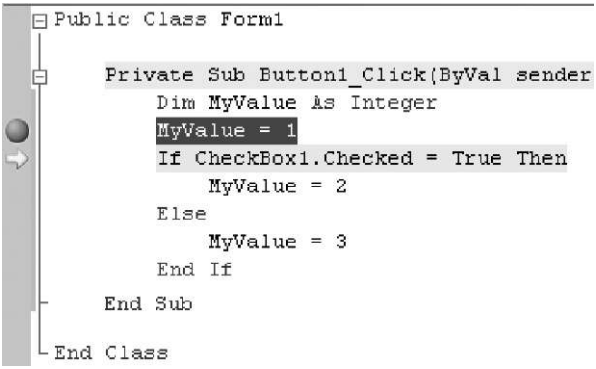
1. Установіть точку переривання на другому рядку коду (MyValue = 1), клацнувши навпроти цього рядка на сірій ділянці, що розташована ліворуч від коду, або клацнувши рядок правою кнопкою миші та обравши в контекстному меню команду **Insert Breakpoint.**



2. Побудуйте та запустіть проект. Після того як з'явиться форма, клацніть кнопку. Код буде виконано до рядка, що містить точку переривання.

```
• Public Class Form1

    Private Sub Button1_Click(ByVal sender
        Dim HyValue As Integer
        HyValue = 1
        If CheckBox1.Checked = True Then
            HyValue = 2
        Else
            HyValue = 3
        End If
    End Sub

End Class
```

3. Щоб виконати рядок коду `MyValue=1`, натисніть клавішу **F11.** Буде виділено наступний рядок коду.



4. Знову натисніть клавішу **F11,** щоб виконати поточний рядок коду: `If CheckBox1.Checked = True Then`.

Оскільки прапорець CheckBoxl встановлено, наступним буде виділено рядок MyValue=2.

```
• Public Class Forml

    Private Sub Buttonl_Click(ByVal sender
        Dim HyValue As Integer
        HyValue = 1
        If iCheckBoxl. Checked = True Then
            HyValue = 2 US' CheckBoxl, Checked True
        Else
            HyValue = 3
        End If
    End Sub

End Class
```

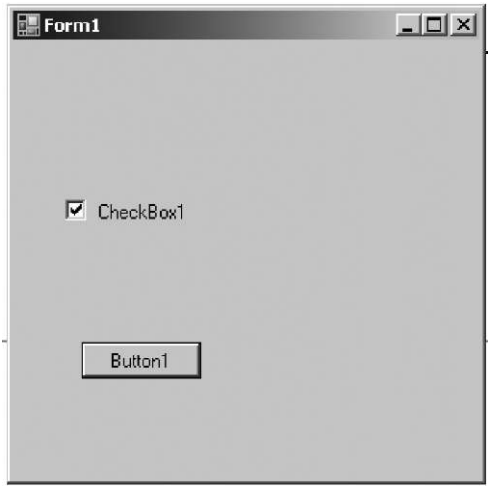
5. Щоб виконати рядок коду MyValue=2, натисніть клавішу **F11.** Буде виділено наступний рядок коду. Зауважте, що ним є рядок із фразою End If.

```
• Public Class Forml

    Private Sub Buttonl_Click(ByVal sender
        Dim HyValue As Integer

        If CheckBoxl.Checked = True Then
            piyVaiud = 2
        Else      ^ MyValue 2
            HyValue = 3
        End If
    End Sub

└End Class
```

6. Для того щоб виконати рядок коду End If, натисніть клавішу **F11.** Буде виділено наступний рядок коду.

```
Public Class Form1

    Private Sub Button1_Click(ByVal sender
        Dim MyValue As Integer

        If CheckBox1.Checked = True Then
            MyValue = 2
        Else
            MyValue = 3
        End If
    End Sub

End Class
```

7. Щоб виконати рядок коду `End Sub`, натисніть клавішу **F11.** Знову з'явиться форма.



8. Зніміть прапорець **CheckBox1** і клацніть кнопку. Повторіть процес виконання коду в покроковому режимі, натискаючи клавішу **F11** та спостерігаючи, які рядки коду

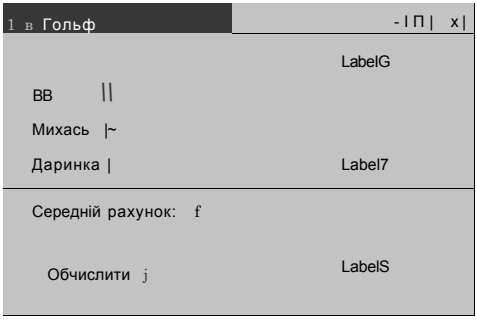
виділяються, коли виконується код. Цього разу буде виконано код за словом `Else`, тобто оператор `MyValue=3`.

```
• Public Class Forml

    Private Sub Buttonl_Click(ByVal sender
        Diin HyValue As Integer

        If CheckBoxl.Checked = True Then
            HyValue = 2
        Else
            HyValue = 3
        End If
    End Sub

-End Class
```

Ось тепер ви змогли пересвідчитись, як насправді працює оператор `I f ...Then...Else`!

Вправа 9.1. Визначаємо вправних гравців

Михась і Даринка зранку, після розмови про програмування, вирішили показати ВВ гру в гольф. Уявіть собі: ВВ не знав про цю гру зовсім нічого! Щоб простіше було обчислювати рахунок, друзі вирішили розробити програму. Допоможіть їм! Створіть форму, що відображатиме рахунок гри в гольф та деякі пов'язані з ним обчислення. Файли, необхідні для виконання цієї вправи, містяться в папці **Вправа_9.1.**



Після того як ви клацаєте кнопку **Обчислити,** визначається середній рахунок гри. Оскільки вмістом текстового поля є рядок, а не число, то цей рядок вам доведеться перетворити на число. У Visual Basic .NET для цього існує кілька способів. Наведемо один із них:

```
x = Val(TextBoxl.Text)
```

За допомогою функції `Val` обчислюється значення, що записане в текстовому рядку. Потім ви можете присвоїти його змінній, наприклад змінній `x`.

Кожному гравцеві на формі Гольф відповідає напис **(Label6-Label8).** Якщо бал гравця вищий за середній, перетворіть відповідний напис на «Добрий рахунок», інакше використайте текст «Продовжуйте практикуватися».

Коли програма розпочинає роботу, поля написів мають бути порожніми.

— Це все, що я хотів розповісти про умовні оператори. Тепер ви знаєте про кілька потужних інструментів програмування і можете використовувати вкладені оператори `If...Then` та оператори `If...Then...Else`.

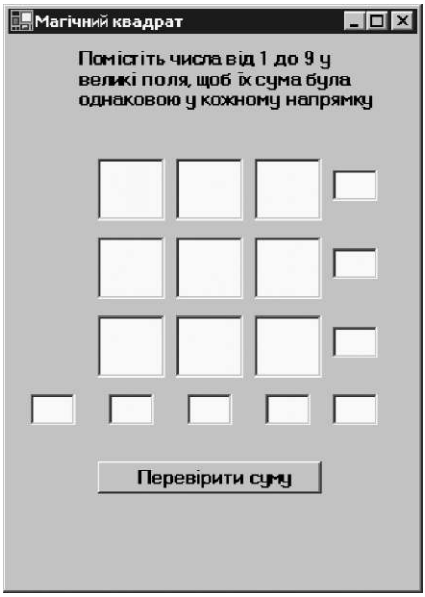
— Отже, наші програми можуть приймати рішення! — вигукнув Михась.

— Так, — підтвердив слова свого друга ВВ, а потім додав: — А тепер розгляньмо математичну вправу.

Я не фокусник, як той робот-артист із парку атракціонів, але знаю один магічний трюк: я можу записати числа від 1 до 9 у клітинки таблиці 3x3 так, що їх сума в будь-якому рядку, стовпці або по діагоналі буде однаковою. Ви можете написати програму, що перевірятиме, чи справді ці суми рівні?

Вправа 9.2. Магічний квадрат

Розробіть таку форму, як показано на рисунку. Вона призначена для заповнення магічного квадрата, а також для повідомлення гравців про успіх або невдачу. Якщо хочете заощадити час, можете використати форму з папки **Вправа_9.2.**



Коли ви клацаєте кнопку **Перевірити суму,** обчислюється сума у трьох рядках, трьох стовпцях і за двома діагоналями, а результати з'являються в текстових полях `TextBox10-TextBox17` (рядок полів знизу та стовпець полів праворуч).

Оскільки текстове поле містить рядкові дані, а не числа, вам доведеться перетворити їх на числа. Це можна зробити, зокрема, за допомогою функції `Val`:

```
x = Val(TextBoxl.Text)
```

За допомогою функції `Val` можна перетворити текстовий рядок на записане в ньому числове значення. Потім ви зможете присвоїти це значення змінній або використати його в операторі `If`.

Усі суми мають дорівнювати 15. Сума чисел у всіх дев'яти текстових полях становитиме 45. Напис **Label2** у нижній частині форми можна використати для привітання гравця з успіхом, заохочення його до нових спроб, або для повідомлення про некоректність вхідних даних.

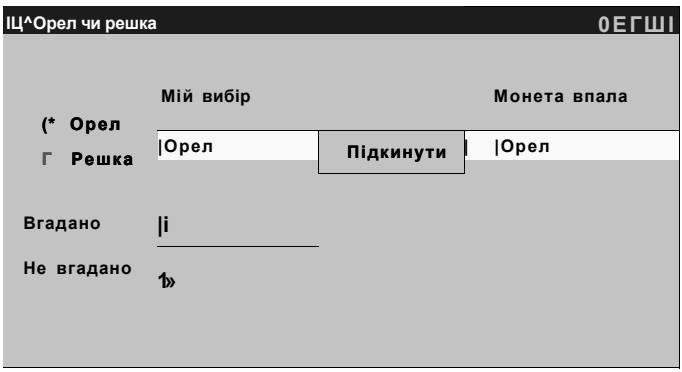
— А якщо ми не дійдемо згоди стосовно того, що слід вводити? — запитала Даринка.

— У такому випадку, — сказав Михась, — можна підкинути монету. Принаймні, я бачив, що так роблять у фільмах.

— Прекрасна ідея! Але давайте зробимо так, щоб це за нас робив комп'ютер! — зауважив ВВ. — Побачимо, чи зможемо ми вгадати, що випаде — орел або решка.

Завдання 9. Підкидаємо монету

Перевірте, чи вмієте ви вгадувати, якою стороною впаде монета. Створіть таку форму, як показано на рисунку, або застосуйте форму з папки **Завдання_9.** Ви встановлюватимете перемикач в одне з двох положень, **Орел** або **Решка.** Ваш вибір буде відображено в полі **Мій вибір,** а в полі **Монета впала** текст «Орел» або «Решка» відображатиметься після натискання кнопки **Підкинути.** Рахунок результатів підкидання монети слід вивести в полях **Вгадано** та **Не вгадано.**



Коли установлюється відповідний перемикач, атрибуту `TextBox1.Text` слід присвоїти значення `"Орел"` або `"Решка"`. Коли хтось клацає кнопку **Підкинути,** генерується випадкове число. Цю дію виконує такий код:

```
Dim MyRandomGenerator As System.Random
MyRandomGenerator = New System.Random
```

```
    Dim RanNum As Integer

' Генеруємо випадкове ціле число від 0 до 2, виключаючи 2
RanNum = MyRandomGenerator.Next(0, 2)
```

Якщо випадкове число дорівнює 0, атрибуту `TextBox2 .Text` присвоюється значення `"Орел"`, інакше — `"Решка"`.
Якщо значення атрибута `TextBox1.Text` дорівнює значенню атрибута `TextBox2.Text`, одиницю буде додано до кількості вгаданих результатів, інакше — до кількості не вгаданих.

Ви вже знаєте, що не можна плутати числа із рядками. Атрибут `Text` завжди має рядкове значення. У Visual Basic .NET можна додавати значення до текстового поля, якщо у ньому вже міститься число, але краще так не робити. Визначте одну цілочислову змінну для запам'ятовування кількості виграшів, а іншу — для кількості програшів. Збільшуйте значення цих змінних, а потім записуйте їх до текстових полів.

Додаткове завдання

Додайте поле, в якому буде показано процентне співвідношення виграшів і програшів.

— А що ти нам запропонуєш цього разу? — запитала Даринка, коли всі завдання вже було виконано.

— Я ще не знаю, але зараз щось придумаю, — невпевнено відповів ВВ, а після невеликої паузи додав: — Ми ще не були в Музеї початку XXI століття! Не маєте бажання туди сходити?

— Мені здається, що нам варто там побувати, адже цікаво дізнатися, що думають нащадки про наш час! — справедливо зауважив Михась.

— То що, йдемо? — перепитав ВВ.

— Пішли! — відповіла Даринка.

— Ні, краще полетіли! — заперечив Михась.

— Не боїтеся? — з іронією запитав ВВ.

— Та вже якось переживемо. По аероциклах! — крикнув Михась і першим побіг до виходу. Але біля дверей він згадав, що не зможе їх відчинити, доки вони не дадуть правильні відповіді на всі тестові запитання кодового замка.

Кодовий замок

1. Як потрібно виділяти вкладений оператор?

а) ніяк не виділяти;

б) брати у квадратні дужки;

в) виділяти відступом.

2. Скільки у коді має бути операторів `End If`, коли один оператор `If` міститься в іншому?

а) 0;

б) 1;

в) 2.

3. За яким словом записують код, що виконується, коли умова в операторі `If` хибна?

а) `Then;`

б) `Else;`

в) `Otherwise.`

4. Скільки рядків коду може бути між операторами `Else` та `End If`?

а) 0;

б) 1;

в) скільки завгодно.

5. Що відбувається із вказівником на рядок, який виконується, під час виконання оператора `If` у покроковому режимі?

а) вказівник залишається перед оператором `If;`

б) вказівник переміщується всіма рядками, але при цьому виконуються лише деякі рядки коду;

в) вказівник на рядок, що виконується, «перестрибує» рядки, які не потрібно виконувати.

6. Чи можна присвоїти число текстовій змінній?

а) так, можна, звичайним оператором присвоєння;

б) ні, не можна в жодний спосіб;

в) так, можна, скориставшись функцією зведення типів.

7. Чи можна всередині конструкції `If умова1 Then...Else` знову використовувати вираз `If умова1 Then`?

   а) ні, не можна — це призведе до помилки;

   б) так, можна, хоч і це не логічно;

   в) так, можна, це звична ситуація у програмуванні.

8. Якого значення набуде змінна `x` після виконання такого коду?

```
Dim x As Integer
x = 1
If x > 0 Then
    If x > 1 Then
        x = 2
    End If
    x = 3
End If
```

   а) 1;

   б) 2;

   в) 3.

День 10
Циклічність — крок до оптимізації

Цього разу (всупереч побоюванням Михася) Даринка та Михась відчували себе після вчорашньої мандрівки на аероциклах значно краще, ніж тоді, коли вони вперше сіли за кермо повітряного транспорту. ВВ показав їм картинг-трек, на якому було так весело! Шкода тільки, що вони туди потрапити не змогли — не було вільних картингів. Але всі вирішили, що це місце потрібно обов'язково відвідати знову.

Зранку друзі зібралися в кімнаті ВВ, який розповів Михасю та Даринці, чим відрізняються сучасні картинги від тих, які були раніше, пояснив правила поведінки, а потім, скориставшись мережею, зробив замовлення на 3 картинги. Після цього ВВ розпочав черговий урок.

Повторення коду

У житті ми часто робимо одні й ті самі речі знову й знову, — розмірковував ВВ. — Пригадую, як я тричі фарбував свій аероцикл, щоб він мав чудовий вигляд.

— Мама каже, що їй доводиться повторювати мені п'ять разів, щоб я виніс сміття, перш ніж я це зроблю, — навів приклад Михась. — Але насправді вона повторює це лише чотири рази, я підраховував.

— Так само і з програмами, — відповів ВВ. — Часто виникають ситуації, коли потрібно, щоб програма виконувала одну й ту саму дію знову й знову, доки вона не виконає її певну кількість разів. Тому в усіх мовах програмування використовують цикли, код у яких виконується декілька разів. Цикли — дуже зручний засіб, завдяки якому обсяг коду, що вам доводиться писати, зменшується. Коли ви вставляєте оператор у цикл, вам необхідно написати його лише один раз, а виконуватися він може багато разів. Зазвичай такий підхід дає змогу зменшити кількість помилок.

Ось приклад неефективного кодування. Уявіть, що вам потрібно додати всі числа між 1 та 100. Це можна зробити, написавши одне довге присвоєння, наприклад:

```
Dim TotalCount As Integer
TotalCount = 0

TotalCount = 1 + 2 + 3 + 4  'тощо
```

Важкувато?! Чим би це замінити?

```
Dim TotalCount As Integer
TotalCount = 0
TotalCount = TotalCount + 1
TotalCount = TotalCount + 2
TotalCount = TotalCount + 3
TotalCount = TotalCount + 4  'тощо
```

Ще більше коду, який повторюється! Один рядок вам доведеться повторювати сто разів!

А якщо йтиметься про числа від 1 до 1000? Як ви їх збираєтеся додавати?

Оператор визначеного циклу

Ось тут і стануть у нагоді цикли. Як я вже згадував, у всіх мовах програмування є цикли, за допомогою яких код можна виконувати знову й знову. Часто вам відомо, скільки саме разів він має виконуватися. Наприклад, щоб додати всі числа від 1 до 1000, код повинен виконуватися 1000 разів. Якщо відомо, скільки разів має виконуватися код, ви застосовуєте

визначений цикл. Його так називають, бо в операторі циклу ви визначаєте, скільки саме разів виконуватиметься певний фрагмент коду.

У Visual Basic .NET визначений цикл утворюється за допомогою оператора `For...Next`. Код, який має виконуватися визначену кількість разів, міститься всередині цього оператора. Синтаксис циклу `For...Next` у Visual Basic .NET такий:

```
Dim LoopCounter As Integer
For LoopCounter = ПочатковеЗначення To КінцевеЗначення
   оператор 1
   оператор 2
   оператор N
Next
```

Зауважте, що слова `For`, `Next` та `To` є ключовими. У середовищі Visual Studio вони виділяються синім кольором і пишуться з великої літери. `LoopCounter` — це змінна цілочислового типу, яка має використовуватися в циклі `For...Next`. Цю змінну називають лічильником циклу (вона не обов'язково повинна мати ім'я `LoopCounter`). За допомогою лічильника програма відстежує, скільки разів виконався цикл. Зсередини циклу ви можете отримати до цієї змінної доступ.
`ПочатковеЗначення` та `КінцевеЗначення` мають бути цілими числами, змінними цілочислового типу або виразами, результатами обчислення яких є цілі числа (наприклад, `3 + 1` або `X + 1`). `ПочатковеЗначення` — це початкове значення змінної `LoopCounter`. Кожного разу після виконання тіла циклу (так називають групу операторів всередині циклу) значення змінної `LoopCounter` збільшується на одиницю. Коли значення змінної `LoopCounter` стає більшим за кінцеве значення, виконання циклу припиняється.

Давайте напишемо код, у якому використовуватиметься визначений цикл. Спочатку створіть цикл, що виконується двічі та відображає значення лічильника під час кожного виконання. Створіть нову Windows-програму та назвіть її ForNextTwo. Додайте командну кнопку до форми `Form1`. Відредагуйте обробник події клацання кнопки. Додайте до нього наведений далі код.

```
Dim LoopCounter As Integer
For LoopCounter = 1 To 2
   MessageBox.Show(LoopCounter)
Next
```

Побудуйте та запустіть проект. Клацніть кнопку. Відкриється вікно повідомлення з текстом «1» — це поточне значення змінної `LoopCounter`. Якщо клацнути кнопку **OK**, то відобразиться повідомлення з текстом «2»: тепер поточне значення змінної `LoopCounter` дорівнює двом. Клацніть кнопку **OK**. Виконання циклу завершилося, тому повідомлень більше не виводиться.

Розглянемо виконання коду уважніше. Ви оголосили цілочислову змінну з іменем `LoopCounter`, у рядку `For` надали цій змінній початкове значення 1, а верхньою межею зробили число 2. Оператор, що виконуватиметься двічі, ви розмістили між операторами `For` і `Next`. Кожного разу, коли виконувався цикл, значення змінної `LoopCounter` збільшувалося на один, а у вікні повідомлення виводилося поточне значення цієї змінної (спочатку 1, потім 2). Коли значення `LoopCounter` сягнуло трьох, цикл завершив свою роботу, і вікно повідомлення не відкрилося.

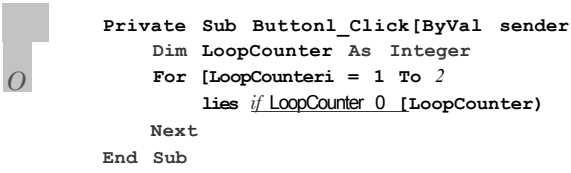
Покрокове виконання визначеного циклу

Давайте дослідимо код циклу `For...Next`, який ви щойно написали, за допомогою налагоджувача Visual Studio .NET. Простежте за виконанням коду в покроковому режимі.

1. Встановіть точку переривання на рядку, що містить оператор `For`. Побудуйте та запустіть програму.

```
Private Sub Button1_Click(ByVal sender
    Dim LoopCounter As Integer
    For LoopCounter = 1 To 2
        MessageBox.Show(LoopCounter)
    Next
End Sub
```

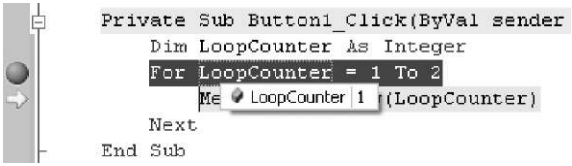
2. Коли відкриється форма, клацніть кнопку `Button1`. Код буде виконано до точки переривання. Рядок коду, помічений точкою переривання, буде виділено. Підведіть курсор до змінної `LoopCounter`. Буде відображено її значення, що дорівнює 0.



3. Натисніть клавішу **F11.** Буде виконано виділений рядок коду:

```
For LoopCounter = 1 To 2
```

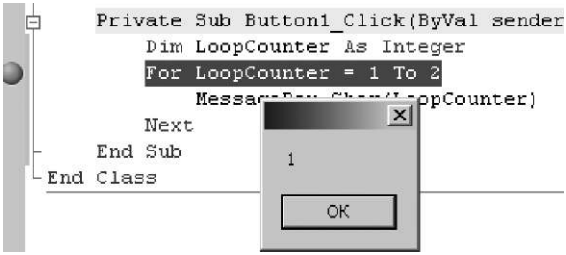
Перемістіть курсор до змінної `LoopCounter`. Значення цієї змінної становить 1.



4. Натисніть клавішу **F11.** Буде виконано виділений рядок коду:

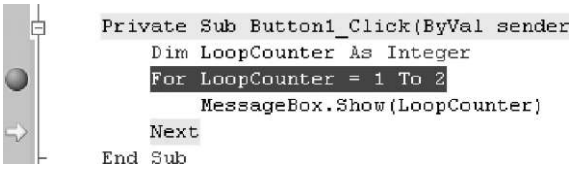
```
MessageBox.Show(LoopCounter)
```

У вікні повідомлення буде виведено одиницю. Клацніть кнопку **OK,** щоб закрити вікно повідомлення.



5. Натисніть клавішу **F11**. Буде виконано виділений рядок коду:

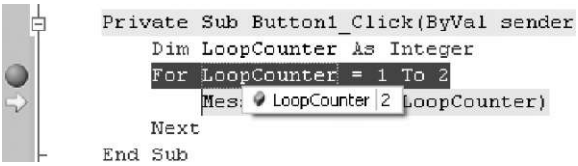
```
Next
```



6. І знову виконується цикл `For`:

```
For LoopCounter = 1 To 2
```

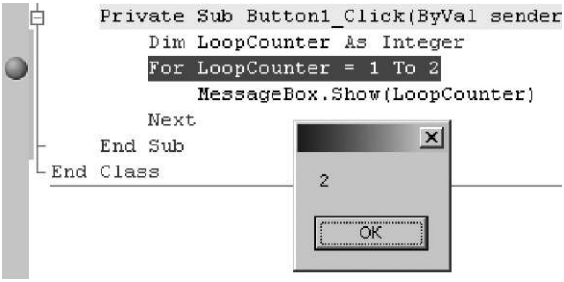
Перемістіть курсор до змінної `LoopCounter`. Тепер її значення становить 2.



7. Натисніть клавішу **F11**. Буде виконано виділений рядок коду:

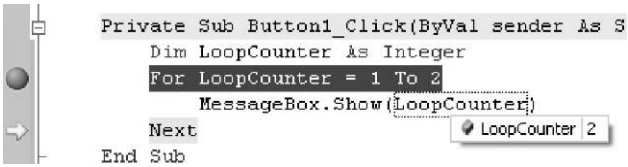
```
MessageBox.Show(LoopCounter)
```

У вікні повідомлення відображатиметься число 2. Клацніть кнопку **OK**, щоб закрити вікно повідомлення.



8. Натисніть клавішу **F11**. Буде виконано виділений рядок коду:

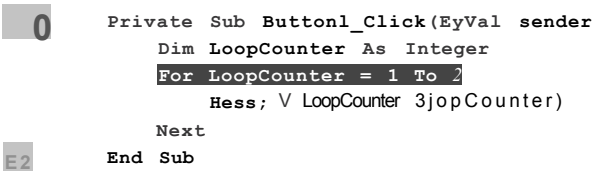
```
Next
```



9. Знову виконується цикл `For`.

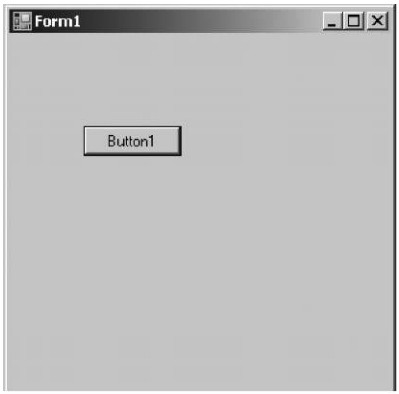
```
For LoopCounter = 1 To 2
```

Перемістіть курсор до змінної `LoopCounter` в операторі `For`. Значення `LoopCounter` становить 3.



10. Натисніть клавішу **F11**. Буде виконано виділений рядок коду:

```
End Sub
```



Форма з'явиться знову. Зауважте, що останнього разу оператор всередині циклу не виконувався, бо значення лічильника перевищило кінцеве значення.

Ви щойно використали інструменти налагодження, щоб переглянути роботу циклу `For... Next` у покроковому режимі, і таким чином дізналися, як виконується код.

Застосування визначеного циклу

— У мене є ідея! — із захопленням продовжив ВВ.

— Яка? — з цікавістю запитала Даринка.

— Давайте напишемо код, що додаватиме цілі числа від 1 до 1000. Це легко зробити за допомогою циклу `For...Next`. Напишіть Windows-програму, що називатиметься AddUp (Додавання). Додайте кнопку до форми `Form1`. Змініть значення атрибута `Text` кнопки `Button1` на `Додати`. Двічі клацніть цю кнопку, щоб змінити обробник події її клацання, і додайте до нього такий код:

```
Dim LoopCounter As Integer
Dim TotalCount As Integer = 0

For LoopCounter = 1 To 1000
   TotalCount = TotalCount + LoopCounter
Next
MessageBox.Show(TotalCount)
```

Напишіть та запустіть програму. Клацніть кнопку **Додати.** У вікні повідомлення буде виведено суму всіх цілих чисел від 1 до 1000. Швидко? Ви написали кілька рядків коду, а програма автоматично виконала 1000 додавань.

Як працює цей код? Спершу оголошуються дві змінні цілочислового типу: `LoopCounter` і `TotalCount`. Змінна `LoopCounter` «стежить» за тим, скільки разів виконується цикл, а у змінній `TotalCount` зберігається сума чисел. У циклі `For` змінній `LoopCounter` надається початкове значення 1, а кінцеве значення покладається рівним 1000, ми ж хочемо, щоб цикл виконувався 1000 разів. Оператор, що повторюється, додає поточне зна-

чення змінної LoopCounter до поточного значення TotalCount і зберігає обчислене значення в змінній TotalCount. Отже, кожної ітерації циклу значення змінної LoopCounter додається до змінної TotalCount (0 + 1 = 1, 1 + 2 = 3, 3 + 3 = 6, 6 + 4 = 10 тощо). Коли значення змінної LoopCounter сягне 1000, цикл завершиться. Програма «виходить» із циклу, та виконується наступний після оператора Next рядок коду — виведення значення змінної TotalCount у вікні повідомлення.

Приклад програми з циклом

Тепер подивимось, як визначений цикл виконує не математичні обчислення, а щось інше. Пам'ятайте: ви можете записати будь-який код усередині циклу For...Next, наприклад код для автоматичного створення довгих текстових рядків із повідомленнями, що виводитимуться знову й знов, або для змінення значень атрибутів з урахуванням значення лічильника циклу. Отже, розглянемо приклад застосування циклу для створення довгого текстового рядка.

Створіть нову Windows-програму з назвою GarbageOut (Виносимо сміття). Додайте до форми Form1 кнопку. Змініть значення атрибута Text кнопки Button1 на Мама говорить. Двічі клацніть кнопку Button1, щоб змінити код обробника події клацання кнопки. Додайте такий код:

```
Dim MomMessage As String
Dim i As Integer = 0
MomMessage = "Будь-ласка, винеси сміття. "
For i = 1 To 5
   MessageBox.Show(MomMessage)
   MomMessage = MomMessage & vbNewLine & MomMessage
Next
```

Побудуйте та запустіть проект. Клацніть кнопку **Мама говорить,** і ви побачите мамині слова. Клацніть кнопку **OK.** Зверніть увагу, що мамине повідомлення щоразу стає все більшим.

— Добре, добре! Здається, я зрозумів! — зрадів Михась. — Цикл є одним із засобів поєднання тексту.

— Розглянемо інший приклад, — продовжив ВВ. — Додайте ще одну кнопку до форми `Form1`. Змініть значення атрибута `Text` кнопки `Button2` на `Зробити кольоровою знову`. Двічі клацніть кнопку `Button2`, щоб змінити обробник події клацання кнопки. Додайте до нього такий код:

```
Dim i As Integer = 0
For i = 2 To 6
   If i < 3 Or i > 5 Then
      MessageBox.Show(i)
      Form1.ActiveForm.BackColor = System.Drawing.Color.Red
   Else
      MessageBox.Show(i)
      Form1.ActiveForm.BackColor = System.Drawing.Color.Blue
   End If
Next
```

Створіть та запустіть проект. Клацніть кнопку **Зробити кольоровою знову.** Буде виведено вікно повідомлення з поточним значенням лічильника циклів. Зауважте, що тепер початкове значення лічильника дорівнює двом, а кінцеве — шести. Оператор `If...Then...Else` в циклі `For...Next` «вирішує», як змінити колір форми, залежно від значення лічильника циклу. Якщо це значення становить 2 або 6, форма буде червоною, інакше — синьою.

Вправа 10.1. Маленький острів

Якось ВВ запитав у Михася та Даринки:

— От я не розумію: яка різниця між «зайцем» і «кроликом»? Як на мене, то це одна й та сама тварина.

Михась із Даринкою розсміялись і пояснили відмінність, а ВВ запропонував створити програму, яка буде рахувати кроликів:

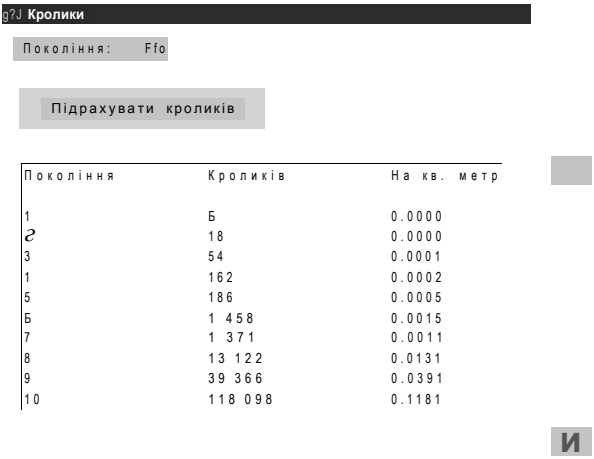
— Припустимо, на острів потрапляють два кролики та дають приплід — чотири тваринки. Кроликів стає шестеро. Усі вони теж дають приплід і через два покоління на острові житиме вже 18 кроликів (6 + 12 = 18).

— Але ж тоді, за твоїми розрахунками виходить, що кролики живуть вічно! — зауважила Даринка.

— А тобі це не подобається? — з усмішкою запитав Михась у Даринки, а тоді відповів ВВ: — Давай спробуємо порахувати кроликів!

Як зростатиме популяція кроликів із появою кожного нового покоління?

Намалюйте форму, як показано на рисунку, або використайте форму з папки **Вправа_10.1.**



Виведіть номер покоління, відповідну загальну кількість кроликів всього та кількість кроликів на квадратний метр, припускаючи, що площа острова дорівнює одному квадратному кілометру, а у квадратному кілометрі 1 000 000 квадратних метрів.

Зробіть так, щоб користувач міг обирати кількість поколінь.

Для легшого читання даних, які виводяться на екран, використайте константи Visual Basic — vbTab і vbNewLine.
Щоб відформатувати вихідні дані, скористайтеся функцією Format. Наприклад:

```
VariableX = Format(VariableX, "###, ##0.0000")
```

Вкладені цикли

Вам відомо, що можна вставити один цикл For...Next в інший? Внутрішній цикл For...Next — це код, що багаторазово виконується на кожній ітерації зовнішнього циклу For... Next (кожне виконання тіла циклу зветься його ітерацією). Тож скільки разів виконуватиметься внутрішній код? Щоб відповісти на це питання, давайте подивимося на приклад.
Створіть нову Windows-програму під назвою LoopTheLoop (Зацикліть цикл). Додайте кнопку до форми Form1, двічі клацніть її, щоб змінити код обробника події клацання кнопки. Додайте такий код:

```
Dim OuterLimit As Integer = 3
Dim InnerLimit As Integer = 4
Dim OuterCounter As Integer
Dim tonerCounter As Integer
Dim Total As Integer = 0
For OuterCounter = 1 To OuterLimit
    For InnerCounter = 1 To InnerLimit
        Total = Total + 1
    Next
Next
MessageBox.Show("Разом: " & Total)
```

Побудуйте проект і запустіть його. Клацніть кнопку Button1. Буде виведено вікно повідомлення з текстом "Разом: 12". Це загальна кількість ітерацій внутрішнього циклу. Знову погляньте на код. Ви помітили, що 12 — це 4 х 3 (кількість ітерацій внутрішнього циклу, помножена на кількість ітерацій зовнішнього циклу)? Правильно! Кожного разу, коли виконується зовнішній цикл, внутрішній цикл виконується чо-

тири рази. Якщо зовнішній цикл виконується тричі, то внутрішній цикл — 4 x 3 = 12 разів.

Зауважте, що в цьому прикладі ми використали дві змінних, `OuterLimit` та `InnerLimit`, щоб визначити кількість ітерацій зовнішнього та внутрішнього циклів. Кількість ітерацій не обов'язково має бути цілим числом. Вона може бути змінною або виразом, що має цілочислове значення. Зверніть увагу також на відділення внутрішнього циклу відступами.

Переривання визначеного циклу

— Сьогодні будемо кататися на картингах, — сказав ВВ. — Якщо ви їздитимете по колу, може запаморочитися голова. Тоді, за правилами техніки безпеки, необхідно з'їхати з траси, поки ви не почуватимете себе добре. Програми, як і люди, також можуть втомлюватися.

— Хіба? — сміючись запитали Михась і Даринка.

— Ну, їм то все одно, але вийти з циклу, не закінчивши його, інколи потрібно, — сказав ВВ. — Ну, хоча б навіть тоді, коли у програміста замиготить в очах через неправильно написаний цикл, який мав виводити слова мами! Отже, іноді цикл потрібно зупиняти, перш ніж лічильник циклів сягне максимального значення.

Для цього застосовують оператор `Exit For`. Він записується всередині оператора `If...Then` у тілі циклу. Для прикладу напишемо відповідний код.

Створіть нову Windows-програму, що називатиметься ExitSign (Ознака виходу). Додайте кнопку до форми `Form1`. Змініть значення атрибута `Text` кнопки `Button1` на `Вийти зараз`. Двічі клацніть цю кнопку, щоб змінити обробник події її клацання, додавши наведений далі код.

```
Dim i As Integer = 0
For i = 1 To 5
   MessageBox.Show("i внутрішнє = " & i)
   If i = 3 Then
      Exit For
   End If
Next
MessageBox.Show("i зовнішнє = " & i)
```

Побудуйте і запустіть проект. Клацніть кнопку **Вийти зараз.** Буде виведено вікно повідомлення зі значенням лічильника циклу. Коли лічильник циклу набуде значення 3, виконається умова оператора `If...Then`, а отже, й оператор `Exit For`. Цикл буде завершено без виконання решти ітерацій. Зауважте, що текст «I внутрішнє = 4» ніколи не буде виведено на екран монітора. Коли цикл завершиться, виконуватиметься код після оператора `Next`, і тому останнє вікно повідомлення міститиме текст «I зовнішнє = 3».

Ось інший приклад використання оператора `Exit For`. Додайте до форми `Form1` у програмі ExitSign прапорець, а також ще одну кнопку. Потім змініть значення атрибута `Text` кнопки на `Установити прапорець`. Щоб змінити код обробника події клацання кнопки, двічі клацніть кнопку `Button2`. Додайте такий код:

```
Dim LoopIndex As Integer
For LoopIndex = 1 To 5
   If LoopIndex = 4 Then
      CheckBox1.Checked = True
   End If
   If CheckBox1.Checked = True Then
      MessageBox.Show("Вихід після " & LoopIndex & "-х
                      ітерацій.")
      Exit For
   End If
Next
```

Побудуйте та запустіть проект. Клацніть кнопку **Установити прапорець.** Відкриється вікно повідомлення з текстом «Вихід після 4-х ітерацій». Що відбулося? Цикл `For...Next` виконував-

ся, доки значення змінної LoopIndex не дорівнювало чотирьом. Потім атрибуту Checked прапорця CheckBox1 було надано значення True, а отже умова CheckBox1.Checked = True стала істинною і були виконані такі оператори:

```
MessageBox.Show("Вихід після " & LoopIndex & "-х ітерацій.")
Exit For
```

В результаті цих дій у вікні повідомлення відобразилося значення змінної LoopIndex і виконався оператор Exit For. Цикл завершено.

Тепер, коли прапорець установлено, клацніть кнопку **Установити прапорець** знову. Що станеться цього разу? Оператор Exit For буде виконано, коли значення змінної LoopIndex становитиме 1, і відобразиться повідомлення про вихід після однієї ітерації.

— Ну, напевно, все, — сказав ВВ. — Давайте виконаємо ще кілька вправ, а тоді ми вільні.

— Добре, як можна зробити це найшвидше? — запитав Михась у ВВ.

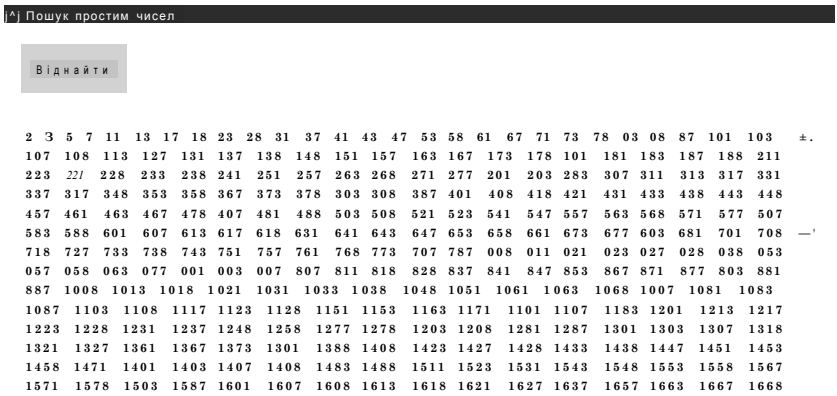
— Якщо будеш замислюватися над тим, що робиш, — відповів ВВ із серйозним виразом обличчя. Усі знову розсміялися. Такого веселого дня в них ще не було.

— Хочу сьогодні ввечері замість піци з'їсти простий біфштекс! — Завершував урок зголоднілий ВВ. — До речі, говорячи про просте, я згадав урок з математики, на якому ми вивчали прості числа. Просте число — це число, що ділиться без залишку лише на себе та 1. Ось кілька прикладів таких чисел: 2, 3, 5, 7. Числа 4, 6, 8 та 9 не є простими, оскільки 4, 6 та 8 діляться на 2, а 9 на 3.

Ви можете написати програму, що обчислить усі прості числа, більші за одиницю та менші від 5000?

Вправа 10.2. Що є простішим за число?

Створіть таку форму, як показано на рисунку, або використайте форму з папки **Вправа_10.2.**



Коли користувач клацне кнопку **Віднайти,** програма має віднайти всі менші від 5000 прості числа та записати їх у текстове поле. Ось кілька підказок.

Насамперед вам слід використати один цикл `For` у тілі іншого циклу `For`. Зовнішнім циклом ви забезпечите перебирання чисел від 2 до 5000. Значенням лічильника зовнішнього циклу буде число, яке перевіряється на простоту. У внутрішньому циклі перебиратимуться потенційні дільники цього числа. Лічильник внутрішнього циклу може змінюватися від 2 до значення, що на 1 менше від значення лічильника зовнішнього циклу. Чому у внутрішньому циклі не потрібно перевіряти числа, більші за значення лічильника зовнішнього циклу? А тому, що число не може ділитися без остачі на число, більше за себе. Чому «на 1 менше»? Тому, що будь-яке число ділиться саме на себе без остачі, тож навіщо це перевіряти?

Вам може знадобитися булева змінна, за допомогою якої можна відстежувати, чи не ділиться значення лічильника зовнішнього циклу на значення лічильника внутрішнього циклу націло. Надайте їй значення `True` перед внутрішнім циклом, а потім змініть його на `False`, якщо з'ясується, що число має дільник.

Отже, у внутрішньому циклі ви маєте перевіряти, чи потенційно просте число ділиться на лічильник внутрішнього

циклу націло. Припустімо, ваші числа — це X та Y. Звідки ви знатимете, чи X ділиться на Y без остачі? Існує кілька способів, за допомогою яких можна визначити це. Оскільки я намагаюся перевірити ваші вміння застосовувати цикли, а не знання з математики, підкажу один із таких способів. Легше за все зробити це за допомогою оператора Mod: вираз X Mod Y дорівнюватиме нулю тоді і тільки тоді, коли X ділиться на Y без остачі.

Після внутрішнього циклу ви можете вивести просте число в текстовому полі, якщо булева змінна, про яку згадувалося вище, все ще має значення True. Звичайно, прості числа мають відокремлюватися пробілами.

— А ви ніколи не думали про пенсію? — запитав ВВ, коли вправу було виконано.

— А навіщо? — розсміялися Михась і Даринка. — Що від цього може змінитися?

— Ну, не знаю, як там у вашому часі, — відповів ВВ, — але у нас нещодавно було прийнято закон, що відкриває деякі цікаві можливості.

— І що ж це за можливості? — запитав Михась.

— Давайте розглянемо все послідовно, — відповів ВВ.

Завдання 10. Майбутнім пенсіонерам

— Я в майбутньому хочу працювати програмістом, проте настане час, коли потрібно йти на пенсію, — почав свою розповідь ВВ. — Тому я вже міркував над тим, як заробити собі пенсійний капітал. Мені відомо: якщо я почну заощаджувати кошти замолоду, пенсія в мене буде значно більшою.

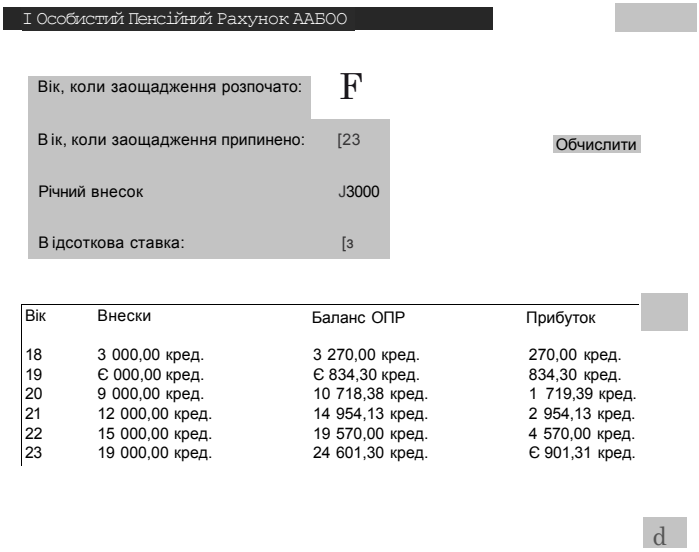
— А чому це? — запитала Даринка.

— Я прочитав про ОПР АА600, — пояснив ВВ. — ОПР — особистий пенсійний рахунок, а АА600 — це ім'я депутата, який запропонував відповідний законопроект. У ньому передбачено здійснення інвестицій у спеціальному податковому режимі. Вам ніколи не доведеться платити податок із заробітку в пенсійний фонд. Ви заощаджуєте гроші, доки вам не виповниться

п'ятдесят дев'ять із половиною років, або доки ви не використаєте їх для навчання чи придбання власного будинку.

Тож давайте створимо калькулятор для ОПР AA600.

Створіть таку форму, як показано на рисунку, або використайте форму з папки **Завдання_10.**



У формі зазначають вік, коли ви починаєте та коли припиняєте заощаджувати кошти, обсяг річного внеску та відсоткову ставку. Ми вивчали оператор циклу, і зараз у нас є чудова можливість використати все, що знаємо про нього. Не забудьте, що в текстових полях містяться рядки, тому треба перетворити їх на числа за допомогою функції `Val`. Наприклад:

```
X = Val(TextBox1.Text)
```

Виведіть вік, загальну суму внесків від початку заощадження, баланс ОПР і прибуток, який дорівнює різниці між сумою внесків і балансом.

Для того щоб відформатувати вихідні значення грошових сум, використайте функцію `FormatCurrency`. Наприклад:

```
TextVariable = FormatCurrency(VariableX)
```

Підказка 1: додавайте суми внесків, перш ніж підраховувати баланс і прибуток.

Підказка 2: коефіцієнт, на який множать суму коштів, визначають за формулою: 1 + відсоткова ставка/100 (наприклад, ставці 9 % відповідає коефіцієнт 1.09).

Додаткове завдання

Ви не сплачуєте податок з ОПР. Що відбуватиметься, якщо ви зробите внесок на звичайний банківський рахунок? Зі звичайного банківського рахунку уряд хоче отримати свою частку. Припустимо, що податок становить 12 % зі щорічного заробітку. Яку суму коштів ви зможете заощадити в такому разі?

— Ну що, важкою була сьогоднішня тема? Ви все зрозуміли з неї? — запитав ВВ.

— Так, — упевнено відповіли Михась із Даринкою.

— Може, підемо кататися на картингах? — запропонував ВВ. Усі погодилися з ВВ і направилися до виходу. Але, як завжди, тут на них чекав кодовий замок.

Кодовий замок

1. Які дії виконує цикл?
   а) повертається до початку блоку коду;
   б) перериває виконання програми;
   в) виконує код знову й знову.
2. Які дії виконує визначений цикл?
   а) приймає рішення знову й знову;
   б) виконує певний набір коду задану кількість разів;
   в) визначає змінні.
3. Якого типу має бути змінна лічильника у циклі `For`?
   а) `Integer`;
   б) `Single`;
   в) `String`.

4. Яким буде значення змінної `LoopCounter`, коли завершиться виконання циклу `For LoopCounter = 1 To 2`?

а) 1;

б) 2;

в) 3.

5. Що відбуватиметься, якщо початкове значення лічильника циклу `For...To` більше за кінцеве?

а) виникне помилка;

б) цикл не виконається жодного разу;

в) виконається лише одна ітерація циклу.

6. Що відбуватиметься, якщо на кожній ітерації циклу `For...To` зменшувати лічильник?

а) програма «зациклиться»;

б) цикл виконається меншу кількість разів;

в) виникне помилка.

7. Припустимо, що певний фрагмент коду містить вкладені цикли. Як у зовнішньому, так і у внутрішньому циклі лічильник змінюється в межах від 1 до 5. Скільки всього разів виконається внутрішній цикл під час виконання такого фрагмента коду?

а) 5;

б) 10;

в) 25.

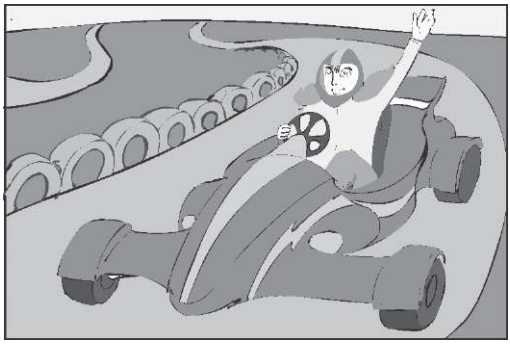
8. Скільки ітерацій у циклі `For LoopCounter = -4 To 3`?

а) 6;

б) 7;

в) 8.

День 11
Невизначена циклічність



Картинг справді був захоплюючим, через що друзі повернулися пізно ввечері. А на ранок усі були повні сил і енергії, а тому ВВ, без жодних відтягувань, перейшов до справи.

Учора ми розбиралися з визначеними циклами — тими, що виконуються задану кількість разів. Цього ж разу ми продовжимо розмову про цикли! Я покажу, як користуватися іншими типами циклів, які називають *невизначеними*. Скільки разів виконуються невизначені цикли, наперед невідомо. Такий цикл завершується тоді, коли виконується певна умова.

Ви дізнаєтеся, що невизначені цикли — це ще один зручний інструмент, яким повинен вміти користуватися кожен програміст. Є випадки, коли ту чи іншу задачу за допомогою визначеного циклу розв'язувати не зручно — тоді в нагоді стає невизначений цикл.

Код, що виконується знову й знову

Багато речей нам доводиться робити знову й знову, доки не відбудеться певна подія. Ця подія є умовою, за виконання якої ми припиняємо повторення тих чи інших дій. Отже, невизначені цикли виконуються доти, доки певна умова не стане істинною чи хибною, потім їх виконання переривається.

Пам'ятаєте, коли ми вирішили прогулятися містом, нам довелося майже годину літати на аероциклах, доки ми, нарешті, не знайшли вільного місця для паркування.
Так само і з кодом. Іноді вам потрібно, щоб один і той самий код виконувався знову й знову, доки певна умова не стане істинною. Наприклад, ваш код може запитувати в користувача пароль доти, доки той не введе його правильно. Або програма може відтворювати фонову музику, поки виконує обчислення.

Є два типи невизначених циклів: «виконувати, поки» та «виконувати, доки не». Перший виконується, поки певна умова залишається істинною, а другий — доки певна умова не стане істинною. Головне — обрати, який саме цикл вам потрібен: «виконувати, поки» чи «виконувати, доки не».

Цикл «виконувати, поки»

Цикл `Do While...Loop` («виконувати, поки») виконує блок коду знову й знову, поки певна умова залишається істинною. Тільки-но умова стає хибною, цикл завершується, а виконання коду всередині циклу припиняється. Умова, що керує циклом, зазначається в дужках після ключового слова `While`. Ось синтаксис циклу `Do While...Loop` у Visual Basic .NET:

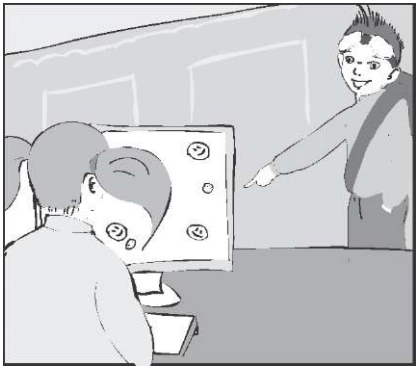
```
Do While (умова)
   оператор 1
   оператор 2
   оператор N
Loop
```

Зауважте, що слова `Do`, `While` та `Loop` є ключовими. У Visual Studio .Net вони виділяються синім кольором. Умова береться в дужки. Нею може бути будь-який булевий вираз. Наприклад, `X < 4`. Код, який має виконуватися повторно, записується між фразами `Do While` та `Loop`. Його називають тілом циклу, а кожне виконання цього коду — ітерацією циклу. На початку кожної ітерації перевіряється вказана після слів `Do While` умова. Якщо вона є істинною, тіло циклу виконується знову. Інакше цикл завершується.

Тепер я продемонструю вам приклад використання циклу `Do While...Loop`. Створіть нову Windows-програму під назвою DoWhileLoop. Додайте командну кнопку до форми `Form1`, двічі клацніть її, і до обробника події клацання кнопки додайте такий код:

```
Dim WhileValue As Integer
WhileValue = 0
Do While (WhileValue < 2)
   MessageBox.Show(WhileValue)
   WhileValue = WhileValue + 1
Loop
```

Побудуйте та запустіть проект. Клацніть кнопку **Button1.** У вікні повідомлення буде виведено значення 0. Клацніть кнопку **OK.** У вікні повідомлення буде виведено значення 1. Якщо ще раз клацнете цю кнопку, то вікно повідомлення закриється.



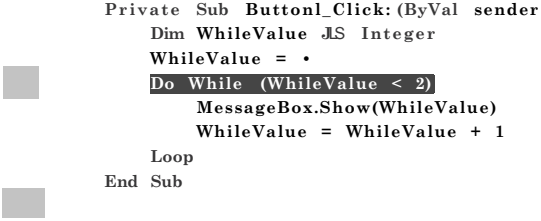
Як працює цей код? Ми оголосили змінну `WhileValue` та присвоїли їй початкове значення 0. Кожної ітерації циклу значення змінної `WhileValue` збільшується на 1. На початку циклу перевіряється, чи `WhileValue < 2`. Якщо ця умова виконується, відкривається вікно повідомлення, а значення змінної `WhileValue` збільшується на одиницю. Якщо умова не виконується, цикл завершує свою роботу. Цикл буде виконано двічі, перш ніж умова `WhileValue < 2` стане хибною. Першого разу у вікні повідомлення буде виведено 0, другого — 1, а втретє значення змінної `WhileValue` дорівнюватиме двом. Отже, на третій ітерації циклу умова перетвориться на хибний булевий вираз 2 < 2. Виконання циклу завершиться, і повідомлення більше не виводитиметься.

/ Код між ключовими словами `Do While` та `Loop` слід виділяти відступами. Так ви можете легко побачити, які інструкції будуть виконуватися повторно. Виділений відступами код легко читати та налагоджувати.

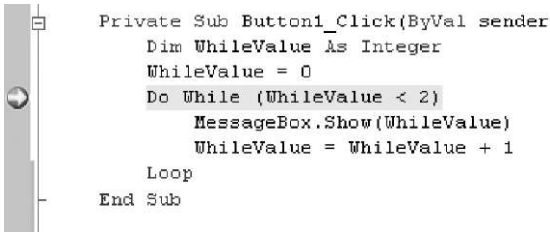
Покрокове виконання циклу Do While...Loop

А тепер, скориставшись інструментами налагодження середовища Visual Studio .NET, виконаємо цикл `Do While...Loop` із програми DoWhileLoop у покроковому режимі.

1. У вікні коду на рядку `Do While (WhileValue < 2)` установіть точку переривання. Цей рядок буде виділено червоним, а зліва від нього відобразиться червоне коло.



2. Запустіть проект. Код буде виконано до точки переривання. Рядок, помічений точкою переривання, буде виділено жовтим кольором.



3. Натисніть клавішу **F11** для виконання виділеного рядка коду:

```
Do While (WhileValue < 2)
```

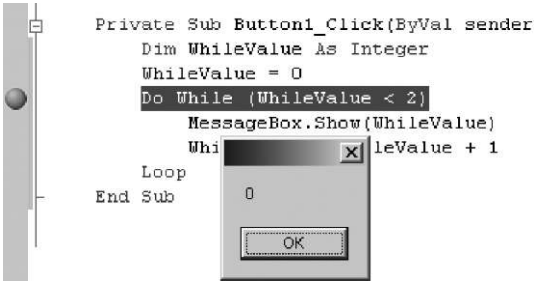
Наведіть покажчик миші на змінну `WhileValue`, яка має значення 0.

```
Private Sub Buttonl_Click(ByVal sender
    Dim WhileValue As Integer
    ShileVaiud = 0
          V WhileValue 0 line < 2)
        HessageBox.Shou(WhileValue)
        WhileValue = WhileValue + 1
    Loop
End Sub
```

4. Натисніть клавішу **F11** для виконання виділеного рядка коду:

```
MessageBox.Show(WhileValue)
```

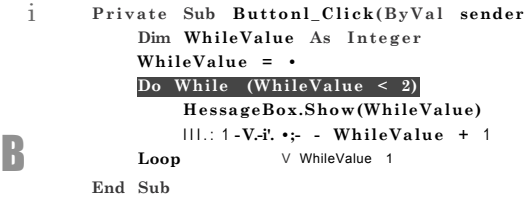
У вікні повідомлення буде виведено значення 0. Клацніть кнопку **OK.**



5. Натисніть клавішу **F11,** щоб виконати виділений рядок коду:

```
WhileValue = WhileValue + 1
```

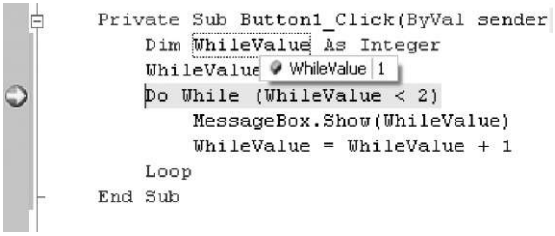
Значення змінної `WhileValue` буде збільшено на один.



11. Натисніть клавішу **F11**, щоб виконати виділений рядок коду:

```
Loop
```

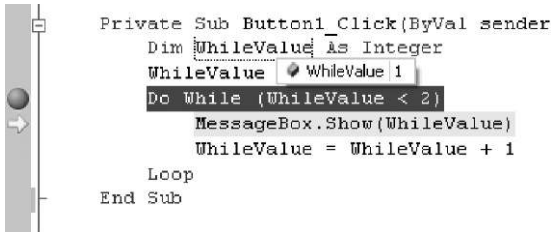
Першу ітерацію циклу завершено. Цикл буде виконано знову.



7. Натисніть клавішу **F11** для виконання виділеного рядка коду:

```
Do While (WhileValue < 2)
```

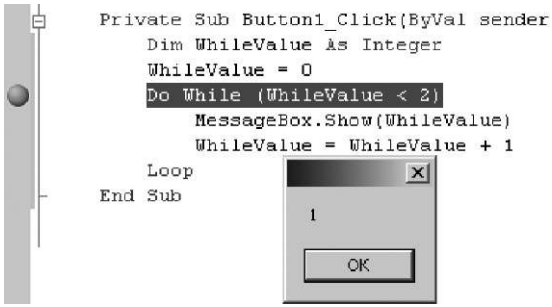
Наведіть покажчик миші на змінну `WhileValue`, яка має значення 1.



8. Натисніть клавішу **F11** для виконання виділеного рядка коду:

```
MessageBox.Show(WhileValue)
```

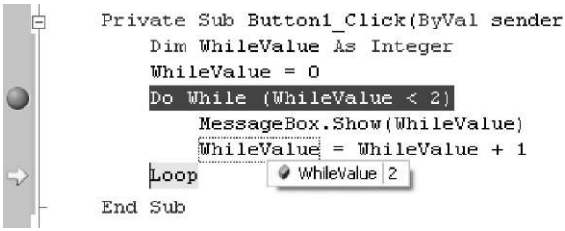
У вікні повідомлення буде виведено значення 1. Клацніть кнопку **OK**.



9. Натисніть клавішу **F11** та виконайте виділений рядок коду:

```
WhileValue = WhileValue + 1
```

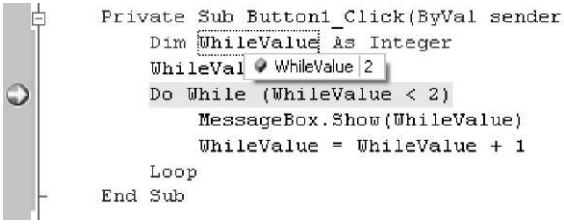
Значення змінної `WhileValue` буде збільшено на один.



10. Натисніть клавішу **F11** та виконайте виділений рядок коду:

```
Loop
```

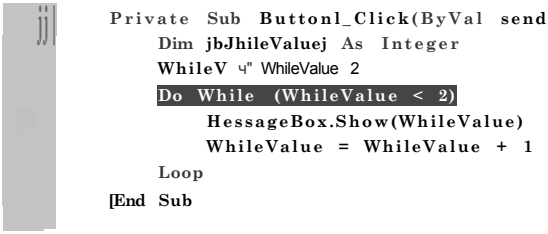
Другу ітерацію циклу завершено.



11. Натисніть клавішу **F11**, щоб виконати виділений рядок коду:

```
Do While (WhileValue < 2)
```

Наведіть покажчик миші на змінну `WhileValue`, яка має значення 2. Тепер умова `WhileValue < 2` є хибною, а отже, цикл завершено. Більше вікно повідомлення не буде виводитись, а значення змінної `WhileValue` не збільшуватиметься.



12. Натисніть клавішу **F11**, щоб виконати виділений рядок коду:

```
End Sub
```

Виконання коду обробника події клацання кнопки завершується, а форма програми DoWhileLoop знову стає активною.

Тепер, коли ви побачили, як працює цикл `Do While...Loop` — цикл типу «виконувати, поки», подивимося, як працює цикл `Do Until...Loop` — «виконувати, доки не».

Do Until...Loop — цикл «виконувати, доки не»

У циклі `Do Until...Loop` блок коду виконується доти, доки умова не стане істинною. Умовою може бути будь-який булевий вираз. Ось синтаксис циклу `Do Until...Loop` у Visual Basic .NET:

```
Do Until (умова)
   оператор 1
```

```
    оператор 2
    оператор N
Loop
```

Слова `Do`, `Until` та `Loop` є ключовими. У Visual Studio .Net вони виділяються синім кольором. Істинність умови перевіряється на початку кожної ітерації циклу. Якщо умова є хибною, код тіла циклу виконується. Коли умова стає істинною, цикл завершується.

Тепер я покажу вам приклад використання циклу `Do Until... Loop` у Visual Basic .NET. Створіть нову Windows-програму під назвою DoUntilLoop. Додайте командну кнопку до форми `Form1`, двічі клацніть її, а до обробника події клацання кнопки додайте такий код:

```
Dim UntilValue As Integer
UntilValue = 0
Do Until (UntilValue > 1)
   MessageBox.Show(UntilValue)
   UntilValue = UntilValue + 1
Loop
```

Побудуйте і запустіть проект. Клацніть кнопку **Button1.** У вікні повідомлення буде виведено значення 0. Клацніть кнопку **OK** — у вікні повідомлення відобразиться значення 1. Якщо ще раз клацнути цю кнопку, вікно повідомлення буде закрито.

Як працює цей код? Ми оголосили змінну `UntilValue` та присвоїли їй початкове значення 0. Кожної ітерації циклу значення змінної `UntilValue` збільшується на одиницю. На початку циклу перевіряється, чи `UntilValue > 1`. Якщо ця умова не виконується, то відкривається вікно повідомлення, а значення змінної `UntilValue` збільшується на одиницю. Якщо умова виконується, цикл завершує свою роботу. Перш ніж умова `UntilValue > 1` стане істинною, цикл буде виконано двічі. Першого разу у вікні повідомлення буде виведено 0, другого разу — 1, а втретє значення змінної `UntilValue` дорівнюватиме двом. Отже, на третій ітерації циклу умова перетворюється на істинний булевий вираз 2 > 1. Виконання циклу завершиться, а повідомлення більше не виводитиметься.

Програми з невизначеними циклами

Розглянемо кілька прикладів використання циклів Do While... Loop і Do Until...Loop. Умови обох циклів є булевими виразами, що безпосередньо не залежать від значення лічильника циклу. У першому прикладі наведено цикл Do While...Loop, в умові якого перевіряється, чи є значення певної змінної більшим або рівним нулю. Коли значення змінної стає меншим нуля, цикл завершується.

Створіть нову Windows-програму під назвою LoopslnAction (Цикли в дії). Додайте командну кнопку до форми Form1. Змініть значення атрибута Text кнопки Button1 на Зворотний відлік. До обробника події клацання кнопки Button1 додайте код:

```
Dim LoopIndex As Integer = 0
Dim WhileController As Integer = 167
Do While (WhileController >= 0)
    LoopIndex = LoopIndex + 1
    WhileController = WhileController - 23 * LoopIndex
Loop
MessageBox.Show(WhileController)
```

Побудуйте і запустіть проект. Клацніть кнопку **Зворотний відлік.** У вікні повідомлення буде виведено значення 63. Як працює цей код? Ми оголосили дві цілочислових змінних LoopIndex та WhileController і надали їм початкові значення. Значення змінної LoopIndex збільшується на одиницю кожної ітерації циклу. При цьому нове значення змінної WhileController обчислюється так: значення змінної LoopIndex множиться на 23 та віднімається від попереднього значення змінної WhileController. Кожного разу, коли виконується цикл, значення змінної WhileController стає все меншим і меншим. Важко передбачити, скільки разів

цикл буде виконано, перш ніж значення змінної WhileController стане меншим нуля. Зауважте, що булевий вираз, який керує циклом, безпосередньо залежить лише від значення змінної WhileController, а не від значення змінної-лічильника LoopIndex.

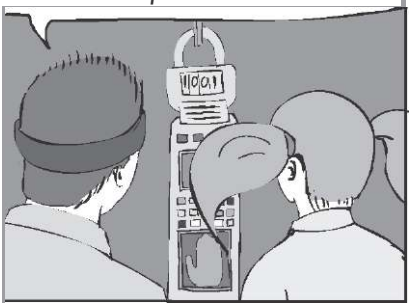
У другому прикладі продемонструємо використання циклу Do While...Loop із умовою, істинність якої залежить від того, чи встановлено перемикач RadioButton3. На кожній ітерації циклу встановлюватиметься по одному перемикачу, доки не буде встановлено перемикач RadioButton3.

У проекті LoopslnAction до форми Form1 додайте другу командну кнопку. Змініть значення атрибута Text кнопки на Поки. Додайте до форми три перемикачі, а до обробника події клацання кнопки Button2 — такий код:

```
Dim LoopIndex As Integer = 0
Do While (RadioButton3.Checked = False)
    LoopIndex = LoopIndex + 1
    If LoopIndex = 1 Then
        RadioButton1.Checked = True
    ElseIf LoopIndex = 2 Then
        RadioButton2.Checked = True
    ElseIf LoopIndex = 3 Then
        RadioButton3.Checked = True
    End If
Loop
MessageBox.Show(LoopIndex)
```

Побудуйте та запустіть проект. Зауважте, що жоден із перемикачів не встановлено. Клацніть кнопку **Поки.** Програма встановить усі три перемикачі, а у вікні повідомлення відобразиться кількість ітерацій циклу. Якщо ви достатньо спостережливі, можете побачити процес встановлення перемикачів RadioButton1 і RadioButton2, що відбувається майже миттєво.

Ми тут будемо сидіти, поки не відповимо правильно на всі запитання



Розглянемо код детальніше. Ми оголошуємо змінну LoopIndex, щоб відстежувати кількість виконаних ітерацій циклу. Проте змінна LoopIndex не використовується у булевому виразі, що керує циклом Do While...Loop. Вона застосовується, щоб кожного разу встановлювати новий перемикач. В умові циклу Do While...Loop пе-ревіряється, чи було встановлено перемикач RadioButton3. Як тільки виконуватиметься рівність RadioButton3.Checked=True (перемикач RadioButton3 встановлено), цикл буде завершено.

Зауважте, що для аналізу значення змінної LoopIndex ми використали ключове слово ElseIf, яке є скороченням пари ключових слів

```
End If
```

У третьому прикладі використаємо цикл Do Until...Loop. У його умові здійснюватиметься порівняння значення атрибута Text текстового поля зі значенням рядкової змінної, яке змінюватиметься кожної ітерації циклу. Додайте текстове поле до форми Form1 проекту LoopslnAction і залиште для його атрибута Text значення TextBox1. Додайте третю кнопку до форми Form1 та змініть значення її атрибута Text на Відповідність тексту. До обробника події клацання кнопки Button3 додайте такий код:

```
Dim MatchText As String = ""
Dim LoopIndex As Integer = 0
Do Until (MatchText = TextBox1.Text)
   LoopIndex = LoopIndex + 1
   If LoopIndex = 2 Then
      MatchText = "Text"
   ElseIf LoopIndex = 3 Then
      MatchText = "Box1"
   ElseIf LoopIndex = 4 Then
      MatchText = "TextBox1"
   End If
Loop
MessageBox.Show("Текст відповідає значенню " & LoopIndex &
                " змінної LoopIndex.")
```

Побудуйте та запустіть проект. Клацніть кнопку **Відповідність тексту.** У вікні повідомлення буде виведено значення змінної `LoopIndex`, якому відповідає рядок у текстовому полі.

У цьому коді застосовується оператор `Do Until...Loop`. Змінна `LoopIndex` використовується лише для того, щоб надати значення змінній `MatchText`. В умові здійснюється порівняння значення змінної `MatchText` зі значенням атрибута `Text` поля `TextBox1`. Під час четвертого проходу циклу змінна `LoopIndex` набуває значення 4 і тому виконується присвоєння `MatchText="TextBox1"`. Булевий вираз стає істинним, оскільки `"TextBox1"="TextBox1"`, і цикл завершується.

А що, коли у поле `TextBox1` ми введемо рядок `The Purple Pit`? Ви зможете додати ще один оператор `If...Then`, щоб для цього значення виконувалося п'ять ітерацій циклу?

Вправа 11.1. Комп'ютерний тир

Створіть гру, яка полягатиме в тому, щоб поцілити в мішень. Напишіть програму, що імітуватиме стрільбу в тирі. Для цього використайте код та форму з папки **Вправа_11.1.** Форма має такий вигляд:

Весь код обробника події клацання кнопки **Нова гра** надано, оскільки в ньому використовуються деякі графічні властивості, які ми не розглядали. Ваше завдання — написати код для кнопки **Грати.**

Мета гри — влучити в малесеньку точку в центрі мішені. Розмір цієї точки дорівнює одному пікселю. Оскільки поле зображення мішені має розмір 200x200 пікселів, центр мішені розміщується в точці з координатами (100; 100). Програма здійснює «постріли» автоматично і обчислює кількість спроб, зроблених для того, щоб поцілити в мішень.

Цілочислові змінні `X` та `Y` зберігатимуть відповідно горизонтальну та вертикальну координати точки влучення. Присвойте цим змінним випадкові цілі числа від 0 до 200. Ось фрагмент коду, що виконує цю дію:

```
Dim MyRandomGenerator As System.Random
MyRandomGenerator = New System.Random
Dim RanNum As Integer

' Генеруємо випадкову величину від 0 до 200, виключаючи 200.
RanNum = MyRandomGenerator.Next(0, 200)
```

Оскільки ми вивчаємо зараз невизначені цикли, використайте один із них. Цикл буде завершуватися, коли `X` та `Y` дорівнюватимуть 100.

Підрахуйте кількість ітерацій циклу, а результат запишіть у текстове поле `TextBox1`.

Цей код малюватиме точки в тому місці, куди програма влучатиме:

```
g.DrawEllipse(MyPen, New Rectangle(X, Y, 1, 1))
```

Інші форми невизначених циклів

Цикли `Do While...Loop` і `Do Until...Loop` можуть бути записані в альтернативній формі — з умовою в кінці циклу. Умова записаного в такій формі циклу перевіряється наприкінці його ітерації, тому тіло циклу хоча б раз обов'язково виконається. У мові Visual Basic .NET альтернативні форми циклів пред-

ставлені операторами Do...Loop While та Do...Loop Until. Ось приклад циклу Do...Loop While:

```
Dim WhileValue As Integer
WhileValue = 0

   MessageBox.Show(WhileValue)
   WhileValue = WhileValue + 1
Loop While (WhileValue < 2)
```

Наведемо приклад циклу Do...Loop Until:

```
Dim UntilValue As Integer
UntilValue = 0
Do
   MessageBox.Show(UntilValue)
   UntilValue = UntilValue + 1
Loop Until (UntilValue > 1)
```

Зауважте, що в обох випадках умова є частиною оператора Loop, а не Do. Інструкції між словами Do та Loop виконуються принаймні один раз. Потім перевіряється умова, за якою визначається, чи слід повторювати цикл. Якби в коді, що наводився вище, початкове значення змінної UntilValue дорівнювало 5, вікно повідомлення зі значенням цієї змінної було б виведене один раз.

Переривання невизначеного циклу

Пригадуєте, вчора ми дізналися, як виходити з циклу For... Next за допомогою оператора Exit For? У подібний спосіб ви можете «втекти» із циклів Do While...Loop, Do Until...Loop, Do...Loop While та Do...Loop Until. Але замість оператора Exit For слід використати оператор Exit Do. Для того щоб вказати умову, у разі істинності якої буде виконано оператор Exit Do, застосовуйте оператор If...Then. У такий спосіб можна вийти з циклу, перш ніж умову його завершення буде виконано.

Давайте напишемо код для виходу з циклу Do Until... Loop. Створіть нову Windows-програму під назвою ExitDo. Додайте командну кнопку до форми Form1. Двічі клацніть

кнопку Button1, щоб змінити код обробника події клацання кнопки. Додайте до нього код, що подібний до коду проекту DoUntilLoop:

```
Dim UntilValue As Integer
UntilValue = 0
Do Until (UntilValue > 10)
   MessageBox.Show(UntilValue)
   UntilValue = UntilValue + 1
   If UntilValue = 5 Then
      Exit Do
   End If
Loop
MessageBox.Show("Кінцеве значення = " & UntilValue)
```

Побудуйте та запустіть проект. Клацніть кнопку **Button1.** На кожній ітерації циклу значення змінної UntilValue збільшуватиметься на одиницю, а у вікні повідомлення виводитиметься номер ітерації (0, 1, 2, 3 та 4). Кожного разу клацайте кнопку **OK.** Коли значення змінної UntilValue сягне 5, умова оператора If...Then стане істинною, а отже, буде виконано оператор Exit Do. Цикл завершиться, а виконання коду продовжиться з наступного рядка після оператора Loop. У цьому рядку розташовано оператор виведення вікна повідомлення з текстом "Кінцеве значення = 5".

— Ну що ж, — сказав ВВ, — на сьогодні досить. Ви навчилися писати код різноманітних невизначених циклів, таких, що виконуватимуться, поки певна умова є істинною або доки вона не стане істинною. Також ви знаєте, як створювати невизначені цикли, в яких умова перевіряється на початку або наприкінці ітерації циклу. Якщо умова перевіряється наприкінці, блок коду буде виконано принаймні один раз. Тепер вам відомо, що різні типи циклів доповнюють один одного. Вони виконують функції, які не може виконати цикл For...Next.

— Хіба це все, що ми повинні були сьогодні зробити? — запитав Михась.

— Ні, — відповів ВВ. — Давайте ще виконаємо кілька завдань щодо циклів, а потім підемо в цирк. Ви думаєте, що там все залишилося таким самим?

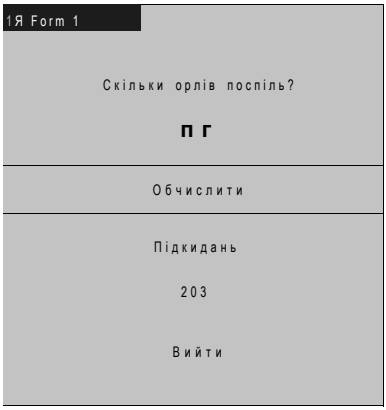
— Мабуть, що ні. Але про це ми дізнаємося, коли побуваємо там, — сказала Даринка.

Вправа 11.2. Коли неймовірне стає можливим

ВВ вирішив вдосконалити програму, за допомогою якої друзі могли вгадувати, як впаде монета.

— Ви коли-небудь цікавилися, скільки разів вам доведеться підкидати монету, щоб тричі поспіль випав орел? А чотири рази поспіль? А п'ять? — запитав ВВ. — Спробуємо це дізнатися за допомогою програми.

Створіть форму, яка мала б такий вигляд:



Для генерування випадкових чисел 0 або 1 ви можете використати такий код:

```
Dim MyRandomGenerator As System.Random
MyRandomGenerator = New System.Random
Dim RanNum As Integer

' Генеруємо випадкову величину від 0 до 2, виключаючи 2
RanNum = MyRandomGenerator.Next(0, 2)
```

Припустимо, що 1 — це орел, а 0 — решка. Продовжуйте генерувати випадкові числа, доки у вас не випаде стільки одиниць поспіль, скільки зазначено в текстовому полі TextBoxl.

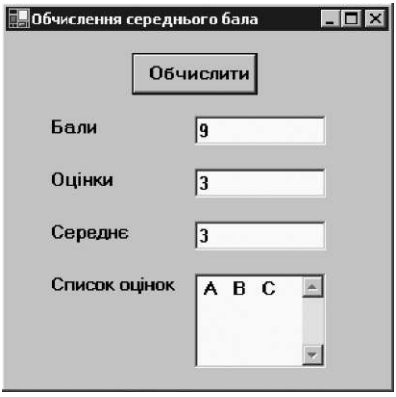
Якщо випадає нуль, починайте знову. Кількість підкидань монети зазначайте в полі `TextBox2`.

А тепер спробуємо розробити програму, яка за кількома оцінками обчислюватиме середній бал. Для виконання цього завдання також зручно буде скористатися невизначеними циклами.

Завдання 11. Оцінки у школі майбутнього

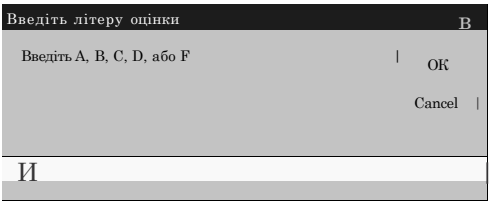
Коли ВВ розповів Михасю та Даринці про їхню шкільну систему оцінювання (вона ґрунтується не на цифрах, а на буквах: А аналогічне 12, В — 10, С — 8 тощо), Михась вирішив написати програму, яка б обчислювала середній бал за рядом оцінок. Спробуйте й ви створити таку програму.
Користувач вводитиме кілька літер, кожній з яких відповідає певний бал. Виходячи з цього, програма має обчислювати середній бал.

Створіть форму, як на рисунку.



Форма, що зображена нижче, використовується для введення даних. Вам не потрібно її створювати — Visual Basic .NET зробить це автоматично.

Оскільки щойно запропонований матеріал є новим, далі наведено код, що відображає таку форму.



Спочатку оголосіть змінну, в якій почергово зберігатимуться літери оцінок:

```
Dim GradeIn As String = ""
```

Потім, у циклі, запишіть такий код:

```
GradeIn = InputBox("Введіть A, B, C, D або F",
                   "Введіть літеру оцінки", "")
```

Цей код присвоює змінній `GradeIn` введену користувачем літеру оцінки. У тому ж циклі потрібно змінити сумарний бал відповідно до введеної користувачем літери. Знаючи сумарний бал, середній бал можна обчислити після завершення циклу. Користувач введе стільки оцінок, скільки забажає. Якщо він не введе жодної оцінки або клацне кнопку **Cancel**, рядок, куди вводиться значення змінної `GradeIn`, залишиться пустим (пустий рядок позначається двома подвійними лапками, між якими не розміщується жодного символу) — це критерій завершення введення.

A — 12 балів;

B — 10 балів;

C — 8 балів;

D — 6 балів;

F — 4 бали.

Будь-які інші літери або числа слід ігнорувати. Користувач може вводити малі чи великі літери.

Помістіть усі введені дані в текстове поле, що розташоване в нижній частині форми.

Після того як буде введено всі оцінки, обчислюватиметься середній бал.

Додаткове завдання

Запишіть за середнім числом балів відповідну літеру оцінки. Використайте оцінки з мінусами, наприклад:

3 F-;

4 F;

5 D-;

6 D;

7 C-;

8 C;

9 B-;

10 B;

11 A-;

12 A.

— Ось і все. Я думаю, що на цьому можна завершити наше сьогоднішнє заняття, — сказав ВВ.

— Не забувайте, що у нас ще заплановано похід до цирку, — нагадала Даринка.

— А ми про це пам'ятаємо, — відповів ВВ. — Але щоб туди потрапити, нам потрібно вийти з дому, а це ми можемо зробити лише тоді, коли дамо правильні відповіді на запитання кодового замка.

Кодовий замок

1. Цикл `Do...While` виконується, поки умова є:
   а) істинною;
   б) хибною;
   в) невизначеною.
2. Де в циклі `Do...While` записується умова?
   а) на початку;

б) в кінці;

в) на початку або в кінці, але не водночас.

3. Цикл `Do...Until` виконується, доки умова не стане:

а) істинною;

б) хибною;

в) невизначеною.

4. Яка команда завершує виконання циклу негайно?

а) `End Do;`

б) `Exit Do;`

в) `Stop Do.`

5. Чи можливо виразити оператор визначеного циклу через оператор невизначеного циклу?

а) ні, неможливо, тому й існують ці дві різні конструкції;

б) це залежить від вмісту визначеного циклу;

в) можливо завжди.

6. Що станеться, коли в умові циклу `Do While...Loop` записати завжди істинний логічний вираз?

а) виникне помилка;

б) цикл не виконається жодного разу;

в) програма «зависне».

7. Якщо в циклі `Do While...Loop` замінити `While` на `Until`, не змінюючи більше нічого:

а) виникне помилка;

б) цикл, що виконувався певну кількість разів, не виконається жодного разу, і навпаки;

в) нічого не зміниться.

8. Якщо є два вкладених цикли і всередині внутрішнього циклу — оператор виходу з циклу, то:

а) програма вийде з обох циклів;

б) програма вийде з внутрішнього циклу;

в) виникне помилка, оскільки програма не знатиме, з якого циклу виходити.

День 12
Підпрограми — зручна подільність

Цирк зазнав великих змін, які Михась із Даринкою помітили відразу, — не було звичних клоунів, акробатів, звірів — усі чомусь сміялися з роботів і з людей, які вдавали з себе роботів. Дітям із минулого це було цікаво, але зовсім не смішно. Коли зранку друзі зібралися на заняття, ВВ, ніби виправляючи свою помилку, замовив через тотальну мережу морозиво. Вже за хвилину до кімнати увійшов робот із трьома порціями холодних ласощів — улюбленої страви Даринки. В минулому часі, звідки вони прибули, такі порції назвали б маленькою, середньою і великою, але ВВ сказав, що то кіло-, мега- і гігапорції, й сам обрав найменшу, бо не зловживав солодким.

— Отже, сьогодні в нас одна з найважливіших тем — підпрограми й функції, — сказав ВВ.

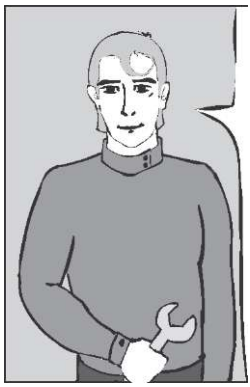
— А мені здавалося, що всі теми однаково важливі, — здивувалася Даринка, приділяючи більше уваги гігапорції морозива, ніж розповіді ВВ.

— Так, але сьогодні ви дізнаєтесь, де використовувати все те, що вивчили, перебуваючи в нашому часі, — промовив ВВ. — Я покажу вам, як писати власні підпрограми й функції та застосовувати ті, що вбудовані в середовище .NET Framework.

Поняття підпрограми

Підпрограми — це своєрідні міні-програми. Їх призначення — виконувати конкретні завдання в більшій програмі. Коли ви пишете програму, поділіть її на окремі функціональні части-

ни та створіть підпрограму для кожної. Обмежувати кількість підпрограм немає потреби, але намагайтеся розподіляти код за підпрограмами так, щоб кожна несла певне функціональне навантаження.



Навіщо використовують підпрограми? Вони полегшують написання та налагодження коду, оскільки дають змогу розмежовувати окремі функції програми. Крім того, набагато легше писати та налагоджувати кілька коротких підпрограм, ніж одну велику програму. Кожна з таких з підпрограм виконуватиметься тоді, коли ви вирішите її запустити.

Підпрограми можна використовувати неодноразово, отримуючи доступ до них із будь-якого місця програми. Тому підпрограми, так само як і цикли, дають можливість уникнути повторень коду: ви записуєте фрагмент коду один раз, а потім, коли це потрібно, звертаєтеся до нього.

У підпрограмах можуть навіть використовуватися інші підпрограми, уявляєте?! Підпрограми — це також засіб розподілення роботи зі створення великої прикладної програми. Окремі програмісти часто отримують завдання написати певний набір підпрограм для однієї програми. Вони відповідають за розробку й налагодження своїх підпрограм та за їх взаємодію з іншими частинами програми. Зазвичай підпрограми використовують для виконання обчислень, форматування та відображення інформації, настроювання інтерфейсу користувача і, нарешті, для введення та виведення даних.

Створення підпрограм

Написання підпрограм для великої програми — це ще одна з тих галузей програмування, які мені найбільше до вподоби. Створення підпрограм дещо схоже на розробку форм. Тут слід поєднувати творчий підхід зі знанням програмування. Оскільки підпрограми схожі на маленькі програмки, вони можуть містити будь-які оператори, що вивчалися нами.

Зараз ви напишете власну підпрограму, а згодом я покажу, як застосовувати її у програмі. Почнемо з вивчення синтаксису підпрограми, а потім я продемонструю, як наповнити цю синтаксичну конструкцію конкретним кодом. Ось синтаксис підпрограми:

```
Private Sub Ім'яПідпрограми()
   оператор 1
   оператор 2
   оператор N
End Sub
```

Зауважте, що слова `Private`, `Sub` та `End Sub` є ключовими. У середовищі Visual Studio вони виділяються синім кольором. Ключове слово `Private` означає, що до цієї підпрограми можна отримати доступ лише з коду форми `Form1`. `Ім' яПідпрограми` — це назва підпрограми. Ви можете використовувати будь-яке ім'я, але краще, щоб воно вказувало на призначення підпрограми.

Зверніть увагу на те, що після імені підпрограми записано дужки. Дані, які передаються до підпрограми, розміщуються у цих дужках. Ви бачили це в усіх підпрограмах обробки подій, якими користувалися раніше. Оператори коду підпрограми містяться між рядками `Sub` та `End Sub`. Оператори виконуються в порядку їх запису. В підпрограмі можна використовувати практично будь-які оператори.

Отже, тепер спробуйте написати власну підпрограму, а потім я покажу, як її викликати. Створіть нову Windows-програму, що називатиметься SimpleSub. Двічі клацніть форму `Form1`. Відкриється вікно коду, пов'язаного з подіями форми.

Після рядка

```
Public Class Form1
```

введіть такий код:

```
Private Sub MyMessage()
   MessageBox.Show("Ось повідомлення від підпрограми MyMessage.")
End Sub
```

Як бачите, під час виконання коду підпрограми `MyMessage` у вікні повідомлення буде виведено текст: «Ось повідомлення від підпрограми MyMessage». Але все ж таки, як нам використати такий код?

Виклик підпрограм

Щоб забезпечити виконання коду підпрограми, вам потрібно написати рядок коду в програмі, який буде «викликати» підпрограму. Коли викликається підпрограма, виконується код, який вона містить. Синтаксис виклику підпрограми простий: потрібно записати ім'я підпрограми, а за ним розмістити дужки:

```
Ім'яПідпрограми()
```

Спробуємо викликати підпрограму, яку ви щойно написали. У проекті SimpleSub додайте до форми `Form1` командну кнопку. Змініть значення атрибута `Text` кнопки `Button1` на `MyMessage`. Двічі клацніть кнопку `Button1`, щоб змінити код обробника події клацання кнопки. Додайте до нього такий рядок коду:

```
MyMessage()
```

Побудуйте та запустіть проект. Клацніть кнопку **MyMessage.** Відкриється вікно повідомлення з текстом: «Ось повідомлення від підпрограми MyMessage».

Звідки воно взялося? Його було виведено кодом, що міститься у підпрограмі `MyMessage`. Клацаючи кнопку, ви викликали підпрограму, і в такий спосіб було виконано цей код.

Тепер додайте ще одну підпрограму до проекту SimpleSub. Вона виводитиме на екран інше повідомлення. Відкрийте вікно

коду форми та введіть такий код після оператора End Sub підпрограми MyMessage:

```
Private Sub YourMessage()
  MessageBox.Show("Ось повідомлення від підпрограми YourMessage.")
End Sub
```

Викличте підпрограму YourMessage з підпрограми MyMessage, а не з обробника події клацання кнопки. Змініть підпрограму MyMessage, щоб вона набула такого вигляду:

```
Private Sub MyMessage()
  MessageBox.Show("Ось повідомлення від підпрограми MyMessage.")
  YourMessage()
End Sub
```

Ми додали виклик підпрограми YourMessage до підпрограми MyMessage.

Побудуйте та запустіть проект. Клацніть кнопку **MyMessage.** Послідовно відкриється два вікна повідомлення. У першому міститиметься текст: «Ось повідомлення від підпрограми MyMessage», а у другому: «Ось повідомлення від підпрограми YourMessage». Як спрацьовує такий код? Обробник події клацання кнопки викликає підпрограму MyMessage. Коли вона викликається, виконується код, що міститься в цій підпрограмі. У підпрограмі MyMessage розміщується два рядки коду: перший виводить на екран повідомлення «Ось повідомлення від підпрограми MyMessage», а другий викликає підпрограму YourMessage, яка, у свою чергу, забезпечує виведення на екран повідомлення «Ось повідомлення від підпрограми YourMessage».

Правда ж, зручно? Ви побудували дві власні підпрограми та викликали першу з обробника події клацання кнопки, а другу — з першої!

Як передають дані підпрограмам у Visual Basic .NET

У Visual Basic .NET, як і в більшості сучасних мов програмування, підпрограмі можна передавати дані. Ви можете розробити підпрограму, що обчислюватиме різноманітні результати або виконуватиме різні дії залежно від того, які дані ви їй

надасте. Змінні, через які дані вводяться до підпрограми, називають *параметрами,* а самі дані — *аргументами.* Коли ви пишете підпрограму, якій плануєте передавати аргументи, треба визначити, скільки буде параметрів і яким буде їх тип. Ось синтаксис підпрограми з параметрами:

```
Private Sub Ім'яПідпрограми(ByVal ім'яПараметра1 As
типПараметра1,ByVal ім'яПараметра2 As типПараметра2,
ByVal ім'яПараметраN As типПараметраN)
    оператор 1
    оператор 2
    оператор N
End Sub
```

Дужки після імені підпрограми не залишаються порожніми: у них міститься список параметрів, значення яких мають бути передані підпрограмі, та їх типів. Зверніть увагу на ключове слово `ByVal`. У середовищі Visual Studio воно виділяється синім кольором. Типи параметрів є звичайними типами змінних, такими як `Integer`, `String` тощо. Зауважте, що кожен параметр відокремлюється від інших комою.

Переконаний: приклад допоможе краще зрозуміти цей синтаксис. Відкрийте вікно коду в проекті SimpleSub. Під рядком `End Sub` підпрограми `YourMessage` додайте новий фрагмент коду:

```
Private Sub GeneralMessage(ByVal InMessage As String)
  MessageBox.Show(InMessage)
End Sub
```

Цій підпрограмі потрібно передати аргумент типу `String`. Коли її буде викликано, на екрані відобразиться вікно повідомлення зі значенням параметра (текстовий рядок).

Давайте викличемо підпрограму `GeneralMessage`, передавши їй аргумент типу `String`. Додайте другу кнопку до форми `Form1` проекту SimpleSub. Змініть значення атрибута `Text` кнопки `Button2` на `AnyMessage`. Двічі клацніть кнопку `Button2`, щоб змінити код обробника події клацання кнопки. Додайте три таких рядки коду:

```
GeneralMessage("Будь-яке повідомлення.")
GeneralMessage("Якесь інше повідомлення.")
GeneralMessage("Ще одне повідомлення.")
```

Побудуйте та запустіть проект. Клацніть кнопку **AnyMessage.** Буде виведено три різних повідомлення. Як таке можливо? У коді обробника події клацання кнопки підпрограма `GeneralMessage` викликається тричі. Кожного разу до підпрограми як аргумент передається новий рядок тексту. Першого разу підпрограма викликається з аргументом `"Будь-яке повідомлення"`, другого разу — з аргументом `"Якесь інше повідомлення"`, а третього — з аргументом `"Ще одне повідомлення"`. Коли викликається підпрограма, їй передається аргумент, який згодом використовується оператором `MessageBox.Show`.

Давайте розглянемо ще один приклад. Напишіть підпрограму, що отримує як аргументи два цілих числа, додає їх, а потім у вікні повідомлення виводить суму. Готові? Додайте нову підпрограму `Adder` (Суматор), до проекту SimpleSub. Підпрограма `Adder` матиме такий вигляд:

```
Private Sub Adder(ByVal AddOne As Integer,
ByVal AddTwo As Integer)
    Dim Total As Integer
    Total = AddOne + AddTwo
    MessageBox.Show(Total)
End Sub
```

Зауважте, що коли викликається підпрограма, їй потрібно передати два цілочислових аргументи. Тепер додайте третю командну кнопку до форми `Form1` проекту SimpleSub. Змініть значення атрибута `Text` кнопки `Button3` на `Adder`. Двічі клацніть кнопку `Button3`, щоб змінити код обробника події клацання кнопки. Додайте такий рядок коду:

```
Adder(34, 57)
```

Побудуйте та запустіть проект. Клацніть кнопку **Adder.** На екрані монітора відобразиться повідомлення з результатом 91. Обробник події клацання кнопки викликає підпрограму `Adder` та передає їй два цілочислових значення — 34 та 57. У підпрограмі `Adder` виконуються три рядки коду. У першому оголошується змінна `Total`, якій потім надається значення суми параметрів `AddOne` (34) та `AddTwo` (57). Змінні `AddOne` та `AddTwo` містять значення, що передаються підпрограмі,

коли вона викликається. Нарешті підпрограма виводить на екран значення змінної `Total` (91).

Тепер ви можете створювати власні підпрограми, викликати їх і передавати їм дані. Далі я покажу вам, як використовувати функції. Функціям ви можете передавати дані так само, як і підпрограмам, але, на відміну від підпрограм, із них можна також отримувати дані.

Створення функцій

Головна відмінність між підпрограмами та функціями полягає в тому, що з функцій можна отримувати дані. Їх називають значеннями, що повертаються. Інакше кажучи, функція повертає значення певного типу. Коли ви пишете функцію, то маєте вказати тип цього значення. Зараз я наведу синтаксис функції, що має параметри та повертає значення:

```
Private Function Ім'яФункції(ByVal ім'яПараметра1 As
типПараметра1, ByVal ім'яПараметра2 As типПараметра2,
ByVal ім'яПараметраN As типПараметраЮ
 As ТипЗначеннняЩоПовертається
   оператор 1
   оператор 2
   оператор N
   Ім'яФункції = ЗначенняЩоПовертається
End Function
```

Слова `Private`, `Function`, `ByVal`, `As` та `End Function` є ключовими. У середовищі Visual Studio вони виділяються синім кольором. Ім'я функції має бути описовим. Список параметрів та їх типів розміщується у круглих дужках після імені функції. Параметри відділяються комами. Кількість параметрів може бути довільною, але значення, що повертається, може бути лише одне. Ви маєте вказати тип цього значення, наприклад, `As Integer` або `As String`, після дужок у заголовку функції. Значення, яке функція повертатиме, присвоюється її імені в останньому рядку коду функції.

Синтаксис функції найкраще пояснювати на прикладі з реального життя. Створіть нову Windows-програму та назвіть її

FunctionJunction. Відкрийте вікно коду та знайдіть рядок Windows Forms Designer generated code. Починаючи з наступного рядка, запишіть такий код:

```
Private Function Multiplier(ByVal MultOne As Integer, ByVal
MultTwo As Integer) As Integer
   Multiplier = MultOne * MultTwo
End Function
```

Ви створили функцію Multiplier, яка отримує два цілочислових параметри, MultOne та MultTwo. Тип значення, що повертається, вказаний після дужок, в які взято список параметрів. У нашому випадку це буде значення цілочислового типу (As Integer). Функція містить один-єдиний рядок коду:

```
Multiplier = MultOne * MultTwo
```

У цьому рядку значення, що повертається, визначається як добуток двох параметрів функції. Ім'я функції Multiplier використовується як ім'я значення, що повертається. Якщо імені функції не присвоїти певного значення, виникне помилка. Давайте викличемо цю функцію та використаємо у програмі значення, що повертається нею.

Виклик функцій

Відмінність виклику функції від виклику підпрограми полягає в тому, що потрібно щось робити зі значенням, яке функція повертає. Наприклад, це значення можна присвоїти певній змінній, що має бути того ж типу, що й значення, яке повертається. Викличте функцію Multiplier, яку ви щойно створили, а потім використайте значення, яке вона повертає, у коді. Для цього до форми Form1 проекту FunctionJunction додайте командну кнопку. Змініть значення атрибута Text цієї кнопки на Помножити. Додайте такий код до обробника події клацання кнопки Button1:

```
Dim Product As Integer
Product = Multiplier(34, 57)
MessageBox.Show(Product)
```

Побудуйте та запустіть проект. Клацніть кнопку **Помножити.** Буде виведено повідомлення, в якому зазначатиметься результат множення 34 на 57, а саме 1938. Як працює такий код? У коді обробника події клацання кнопки оголошується цілочислова змінна `Product`. Після цього викликається функція `Multiplier`, якій передаються аргументи `MultOne` (34) та `MultTwo` (57). У коді функції `Multiplier` значенню, яке вона повертає, присвоюється добуток параметрів `MultOne` та `MultTwo`. Значення, що повертається, — це значення цілочислового типу. В обробнику події клацання кнопки цілочисловій змінній `Product` надається значення, яке повертає функція `Multiplier`, а також здійснюється його виведення у вікні повідомлення.

У мене виникла ідея! Оскільки ви створили функцію `Multiplier`, давайте дещо пофантазуємо! Додайте до форми `Form1` проекту FunctionJunction другу командну кнопку. Змініть значення атрибута `Text` кнопки `Button2` на `Помножити знову`. Додайте такий код до обробника події клацання кнопки `Button2`:

```
Dim Product As Integer
Product = Multiplier(Multiplier(2, 3),
Multiplier(5, 7))
MessageBox.Show(Product)
```

Побудуйте проект та запустіть його. Клацніть кнопку **Помножити знову** — на екрані відобразиться число 210. Цього разу функція `Multiplier` викликатиметься тричі. Спочатку ми викликаємо її з аргументами 2 та 3. Функція повертає значення 6, що використовується як перший аргумент, коли ми викликаємо функцію `Multiplier` втретє. Потім функція `Multiplier` викликається з аргументами 5 і 7 та повертає значення 35, що стає другим аргументом, коли ми викликаємо функцію `Multiplier` втретє. І третього разу ми викликаємо функцію `Multiplier` з аргументами 6 (значення, яке функція `Multiplier` повернула першого разу) і 35 (значення, яке функція `Multiplier` повернула другого разу).

Бачите, як функція може стати аргументом функції (навіть тієї самої)?

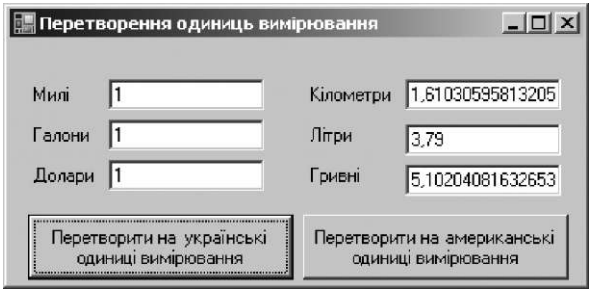
Вправа 12.1. Метри, кілометри, літри — чи може щось інше?

— Тато мені колись розповідав, що у вашому часі були різні одиниці вимірювання. Наприклад, метри та фути, кілометри та милі, літри та галони... Як ви в них розбираєтесь? — запитав ВВ.

— Та все дуже просто! — відповів Михась. — У нас усі знають, як переводити одні одиниці в інші, а тому проблем, пов'язаних із цим, майже не виникає.

— А давайте напишемо програму, яка буде перетворювати одиниці вимірювання! — запропонував ВВ.

Допоможіть у цьому ВВ. Використайте форму з папки **Вправа_12.1,** що має такий вигляд, як на рисунку.



Коли ви клацаєте кнопку **Перетворити на українські одиниці вимірювання,** перетворюються значення правого стовпчика. Коли ви клацаєте кнопку **Перетворити на американські одиниці вимірювання,** перетворюються значення лівого стовпчика.

Використовуйте такі коефіцієнти перетворення:

1 кілометр = 0,621 милі;

1 літр = 0,264 галона;

1 українська гривня = 19,6 центів США.

Ви вже навчилися створювати власні функції та підпрограми. У цій програмі вам потрібно застосувати функції.

Підпрограми та функції, вбудовані в .NET

Каркас .NET Framework містить багато вбудованих підпрограм і функцій, що призначені для виконання поширених завдань. Ці функції були ретельно перевірені та налагоджені. Їх можна використовувати в усіх мовах родини .NET. Якщо є така можливість, застосовуйте саме ці функції, замість того, щоб писати власні. В такий спосіб ви не лише економите свій час, але й стандартизуєте код, робите його легшим для сприйняття. Крім того, деякі з вбудованих функцій ви не зможете написати самостійно. Багато з них отримують доступ до базових класів середовища .NET Framework та інтерфейсу прикладного програмування Windows. Вам до цих засобів звертатися не потрібно!

А зараз я покажу вам кілька вбудованих підпрограм і функцій, які використовують найчастіше, та до яких можна отримати доступ із програм, записаних мовою Visual Basic .NET. Ви побачите, що деякі з них нам вже знайомі з вправ, які ми виконували раніше.

Функції обробки рядків

До каркаса .NET Framework вбудовано багато функцій, що призначені для обробки рядків. Ці функції дозволяють видаляти зайві символи, знаходити підрядки більших рядків, змінювати регістр символів тексту тощо. Призначення більшості з них легко зрозуміти на прикладах.

Синтаксис усіх рядкових функцій схожий. Їх можна застосовувати до одного чи кількох рядків. Деякі з них отримують аргументи, інші — ні. Більшість із них повертає рядкові значення, хоча функція `Length` повертає ціле число. Ось загальний синтаксис виклику вбудованих до Visual Basic .NET рядкових функцій, де словами `деякийРядок` позначено вираз рядкового типу:

```
Змінна = деякийРядок.Ім'яРядковоїФункції (аргумент1,
аргумент2, аргумент^
```

Тепер розглянемо кілька конкретних рядкових функцій. Корисною є функція `Length`. Вона повертає довжину рядка, тобто кількість символів у ньому. Наприклад:

```
MyText = "TextBox1"
myLength = MyText.Length 'повертає 8, довжину рядка "TextBox1"
```

Функції `ToLower` та `ToUpper` застосовують для заміни всіх символів рядка на символи нижнього або верхнього регістра. Погляньмо на ці приклади:

```
MyText = "TextBox1"

MyCaps = MyText.ToUpper 'повертає "TEXTBOX1"

MyText = "TextBox1"
mySmalls = MyText.ToLower 'повертає "textbox1"
```

Якщо функція не має параметрів, під час її виклику після імені функції можна не записувати дужки. Наприклад, вираз `MyText.Length` еквівалентний виразу `MyText.Length()`.

Коли потрібно переконатися, що на початку або в кінці рядка немає зайвих пробілів, тобто ви хочете видалити будь-які пробіли перед рядком тексту та після нього, використовуйте функцію `Trim`:

```
MyText = " TextBox1"

myClean = MyText.Trim    'повертає "TextBox1"

MyText = "TextBox1 "
myCleaner = MyText.Trim 'повертає "TextBox1"
```

Ще одна корисна функція — `Substring`, яка повертає частину рядка (підрядок). Коли ви її викликаєте, то їй слід передати два цілочислових аргументи. Перший аргумент — це початкова позиція, а другий — довжина підрядка. Зауважте, що перша літера в рядку розміщена в позиції 0, а не 1. Друга літера займає позицію 1, а не 2 тощо.

```
MyText = "TextBox1"
MySub = MyText.Substring(0, 4) 'повертає текст "Text",
 'починаючи з позиції 0 довжиною у 4 символи
```

```
MyText = "TextBox1"
MySub = MyText.Substring(1, 2)  'повертає текст "ex",
 'починаючи з позиції 1 довжиною у 2 символи
```

Я знаю, що ви не можете дочекатися, коли побачите ці рядкові функції в дії. Тож давайте створимо невелику програму, що покаже вам, як вони працюють! Створіть нову Windows-програму, що називатиметься BuiltIn. Додайте два текстових поля до форми `Form1`. Видаліть значення атрибута `Text` текстового поля `TextBox2`. Атрибуту `Multiline` цього текстового поля присвойте значення `Yes`, а атрибуту `Scrollbars` — значення `Vertical`. Висоту текстового поля `TextBox2` зробіть майже рівною висоті форми. Додайте до форми `Form1` командну кнопку. До обробника події клацання кнопки додайте такий код:

```
Dim MyText As String
Dim TempText As String = ""
MyText = TextBox1.Text
TempText = TempText & MyText
TempText = TempText & vbNewLine & MyText | ToLower
TempText = TempText & vbNewLine & MyText | ToUpper
TempText = TempText & vbNewLine & MyText | Trim
TempText = TempText & vbNewLine & MyText | Substring(0, 4
TempText = TempText & vbNewLine & MyText | Substring(1, 2
TempText = TempText & vbNewLine & MyText | Length
TextBox2.Text = TempText
```

Запустіть програму. Введіть довільний текст у поле **TextBox1** і клацніть кнопку **Button1.** У текстовому полі **TextBox2** відобразяться результати виконання всіх рядкових функцій, що застосовувалися до рядка, який ви ввели у текстове поле **TextBox1**. Змініть текст у полі **TextBox1** та клацніть кнопку **Button1** знову.

Генерування випадкових чисел

Багатьом програмам потрібно генерувати випадкові числа. Особливо це стосується комп'ютерних ігор, в яких багато випадкових подій, схожих на підкидання монети або кидання кісточок. У середовищі .NET Framework є клас `System.Random`, який містить функції, що використовують для генерування

випадкових чисел. Я допоможу вам написати код, що імітуватиме кидання шестигранних кісточок. До форми Form1 проекту BuiltIn додайте другу командну кнопку. Змініть значення атрибута Text цієї кнопки на Кинути кісточки, а до обробника події її клацання додайте такий код:

```
Dim myRandomGenerator As System.Random
Dim myRandomInteger As Integer
myRandomGenerator = New System.Random
myRandomInteger = myRandomGenerator.Next(1,7)
MessageBox.Show(myRandomInteger)
```

Побудуйте та запустіть програму. Клацніть кнопку **Кинути кісточки.** Відкриється вікно повідомлення з випадковим числом від 1 до 6. Знову клацніть кнопку. У вікні повідомлення буде виведено інше випадкове число від 1 до 6.

Як працює цей код? Спочатку ми оголошуємо змінну типу System.Random, яку називають myRandomGenerator. За допомогою ключового слова New створюємо об'єкт System.Random, і його значення надаємо змінній myRandomGenerator. А потім оголошуємо змінну цілочислового типу, яку називають myRandomInteger. Після цього викликаємо функцію Next(1,7) змінної myRandomGenerator, щоб згенерувати випадкове число від 1 до 6. Під час виклику функції Next(1,7)ми передаємо аргументи 1 та 7, що є нижніми та верхніми межами діапазону випадкових чисел, які ми хочемо генерувати. Функція Next () повертає випадкове ціле число від 1 до 6 (6 = 7 - 1), яке надається змінній myRandomInteger. Останній рядок коду забезпечує виведення випадкового числа на екран у вікні повідомлення.

Перетворення типів

У Visual Basic .NET є дві функції перетворення типів даних, якими можна скористатися, коли ви працюєте з числами. Це функції Val та Int. Розгляньмо спочатку функцію Val. Її використовують для перетворення рядка на число. Якщо в рядку міститься десяткова крапка, ця функція повертає число типу Double, якщо ні — число типу Integer. Функцію Val

часто використовують для перетворення значення параметра Text рядкового типу на число, яке можна використовувати в арифметичних виразах. Ось кілька прикладів застосування цієї функції:

```
Dim MyInt As Integer

MyInt = Val("123") + 123 'повертає 246

Dim MyDouble As Double
MyDouble = Val("123.22") + 123 'повертає 246.22
```

Функція Int повертає цілу частину числа, тобто частину числа ліворуч від десяткової крапки. Погляньте на такі приклади застосування функції Int:

```
Dim MyInt As Integer
MyInt = Int(123) 'повертає 123
MyInt = Int(123.45) 'повертає 123
```

— Я думаю, що теорії на сьогодні досить, — сказав ВВ.

— Сподіваюся, що ми все запам'ятаємо, — зауважив Михась, розбираючись у коді останньої програми.

— Не хвилюйся! — засміявся ВВ. — Для цього в мене є ще кілька вправ.

— Гаразд, — погодилася Даринка, — спробуємо їх виконати. А що потім?

— Ну, потім можемо подивитися кілька фільмів, — відповів ВВ. — Мені здається, що ви ще не бачили «Термінатор 10»?

— Звісно що ні! — здивувався Михась. — А такий вже є?

— Ні, то я пожартував, — розсміявся ВВ, — але цікавих фільмів вистачає.

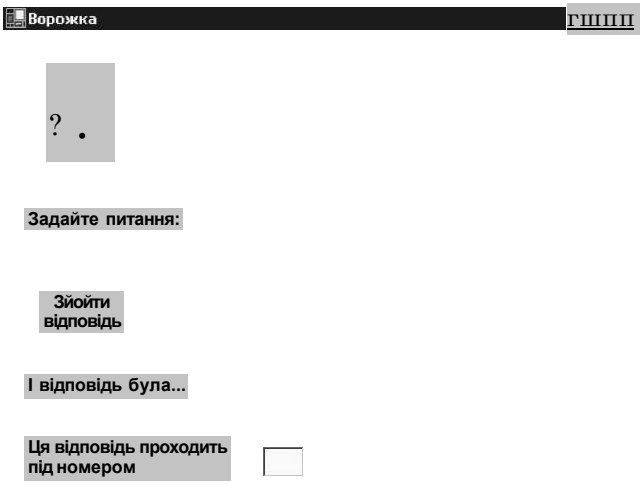
Вправа 12.2. Ворожка

— А що, як існують чарівниці, яким, щоб бачити майбутнє, не потрібна машина часу? — запитала якось Даринка.

— Щось я не вірю в такі байки! — відповів Михась.

— А давайте напишемо програму, що відповідатиме на питання так, як це робила б ворожка, — запропонував ВВ.

Форма має такий вигляд, як показано на рисунку.



Відповідь на питання повинна залежати від кількості голосних у тексті. Виділити окрему літеру з текстового рядка можна за допомогою функції `SubString`. Правильна відповідь має визначатися, коли ви клацаєте кнопку **Знайти відповідь.** Пам'ятайте про важливість регістра: літери «а» та «А» у Visual Basic вважаються різними. Перевірте, чи було введено питання. Подумайте також про цикли.

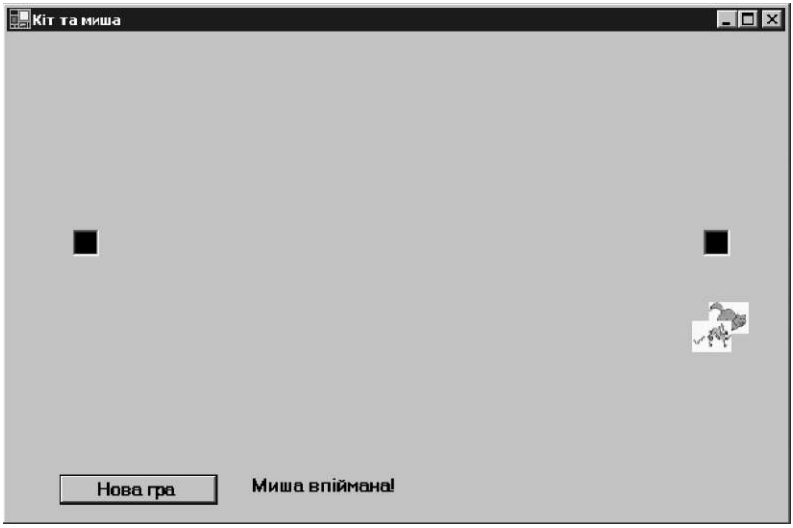
Не забувайте, що кількість символів у рядку можна визначити за допомогою функції `Length`. Символи нумеруються, починаючи від нуля, наприклад, десять перших символів займають позиції від 0 до 9. Пам'ятайте також, що за потреби замість сталого аргументу функції `Length` завжди можна використати змінну.

Пригадайте поширені відповіді ворожок, або вигадайте їх самі, наскільки це дозволяє ваша уява. Відповіді «так» і «ні» допустимі, але вони не цікаві. Вам потрібно ввести принаймні п'ять різних відповідей, хоча можна й більше.

А тепер виконайте завдання, щоб закріпити знання з використання підпрограм і функцій.

Завдання 12. Полювання

Створіть гру, в якій кіт наздоганятиме мишу. Для цього розробіть таку форму, як на рисунку.



Додайте до форми два поля зображень. Полю `PictureBox1` надайте ім'я `Cat`, а полю `PictureBox2` — ім'я `Mouse`. Переміщуйте кота та мишу за випадковими траєкторіями, змінюючи положення полів, що містять зображення кота та миші, шляхом додавання додатних або від'ємних чисел до поточних координат полів.

Якщо відстань між лівими верхніми кутами полів, де розташовано зображення кота й миші, не перевищує 40 пікселів, миша змінює напрям руху. Якщо миша розміщується не далі, ніж за 20 пікселів від кота, вважається, що її впіймано, а якщо не далі, ніж за 20 пікселів від будь-якої нори (чорного текстового поля), то вважається, що вона втекла.

Припустимо, що форма має розмір 600 х 300 пікселів. Тоді перша нора розміщуватиметься в точці з координатами (50; 150), друга — у точці з координатами (530; 150). Якщо мишу впіймано або вона втекла, гра завершується.

Кожну 1/4 секунди виникає подія Timer. До обробника цієї події додайте код, який забезпечуватиме пересування кота та миші. Щоб пересунути поле зображення, вам потрібно задати нові координати лівого верхнього кута поля (X та Y), а потім використати такий оператор:

```
Mouse.Location = New Point(X,Y)
```

Пересунувши кота та мишу, перевірте чи:

- миша втекла;
- кіт упіймав мишу;
- миші потрібно змінити напрям руху, бо вона опинилася поряд із котом;
- кіт або миша розташовані поблизу краю екрана.

У шаблоні є кілька функцій, що допоможуть вам написати програму. Функція HowFar повертає значення відстані між двома точками, заданими координатами X та Y. Наприклад,

```
Distance = HowFar(Cat.Location.X, Cat.Location.Y,
Mouse.Location.X, Mouse.Location.Y)
```

Функція NewChg повертає випадкове число, яке може бути як додатним, так і від'ємним. Воно дорівнює відстані, на яку пересувається миша або кіт. Наприклад,

```
CatXChg = NewChg()
```

В обробнику події клацання кнопки нової гри вкажіть вихідне випадкове розташування кота та миші. Задайте величину змінення координат зображень на кожному кроці гри. Запустіть таймер. Якщо об'єкт таймера має ім'я Timer1, то це можна зробити командою

```
Timer1.Start()
```

Програмісти вважають, що, написавши фрагмент коду, потрібно його перевірити. Розробляйте програму в такому порядку:

- вкажіть початкове місце розташування кота та миші;
- перемістіть кота;
- визначте координати зображення кота;
- перемістіть мишу;

- визначте координати зображення миші;
- перевірте, чи не втекла миша;
- перевірте, чи кіт не впіймав мишу;
- перевірте, чи не заблизько миша перебуває від кота.

Після введення кожного оператора тестуйте код.

Додаткове завдання

Відстежуйте, в який бік дивиться кіт або миша. Змінюйте орієнтацію зображення, якщо кіт чи миша змінюють напрямок руху. Використайте приблизно такий код:

```
Cat.Image.RotateFlip(RotateFlipType.Rotate180FlipX)
```

Обчисліть напрямок руху миші, потім змініть приріст координат й орієнтацію зображення кота так, щоб він рухався й дивився в напрямку миші. Наприклад, якщо координата `X` кота дорівнює 450, а миші — 250, то кіт буде рухатись у від'ємному напрямку.

Створіть поле із зображенням собаки, яка бігатиме за котом.

— Ну що ж, ви добре впоралися зі своїм завданням, — зауважив ВВ.

— А тепер ми можемо дивитися фільм? — запитав Михась.

— Так. Але не забудьте, що на всіх дверях встановлено кодові замки. І тому, щоб нам потрапити до кімнати, де стоїть телевізор, потрібно відкрити кодовий замок.

Кодовий замок

1. Що таке аргумент підпрограми?

а) змінна, що оголошується поза підпрограмою;

б) елемент даних, що передається підпрограмі;

в) кількість операторів у коді підпрограми.

2. Припустимо рядкова змінна `X` має значення `"ABCD"`. Значення якого виразу становитиме `"BC"`?

а) `X.Substring(1, 2)`;

б) `X.Substring(2, 3);`

в) `X.Substring(2, 2).`

3. Як викликати підпрограму?

а) `Ім'яПідпрограми();`

б) `Execute Ім'яПідпрограми();`

в) `GoTo Ім'яПідпрограми().`

4. Як перетворити значення змінної `X` рядкового типу на число?

а) `X.ToNumber;`

б) `Convert(X);`

в) `Val(X).`

5. Що станеться, коли викликати підпрограму, тіло якої містить один рядок — виклик цієї ж підпрограми?

а) такий код є помилковим;

б) програма «зациклиться»;

в) виконання підпрограми завершиться нормально.

6. Що станеться, якщо підпрограма потребує два аргументи, а їй передати три?

а) виникне помилка;

б) третій аргумент буде проігноровано;

в) підпрограма отримає додатковий параметр `New_param`.

7. Для чого використовують функцію `MyText.Trim()` ?

а) для форматування рядка;

б) для змінення регістра букв;

в) для видалення пробілів на початку чи в кінці рядка.

8. Припустімо, що змінна `X` має цілочисловий тип. Якого значення набуде ця змінна після виконання оператора `X = Int (2.7)`?

а) 2;

б) 3;

в) такий код містить помилку.

День 13
Масиви

— Сьогодні в нас одна з найважливіших і найцікавіших тем, — почав театральним голосом свою розповідь ВВ, — яка, безсумнівно, зацікавить вас. Ви здивуєтеся мудрості великих зодчих, які сховали таку сильну таємну зброю в глибини компілятора Visual Basic...

— ВВ, припини кривлятись і переходь до теми! — перебила його Даринка. — А то мені справді стане цікаво і я примушу тебе розповідати, аж доки ти не розкажеш все, що знаєш і не знаєш! — всі засміялися, і ВВ, котрий був у доброму гуморі, почав урок.

Коли потрібні масиви

— Часто трапляється так, що якась окремо взята річ нам не потрібна (наприклад, цеглина — важка, брудна, з якою нічого корисного не зробиш, хіба що в рюкзак покладеш, щоб у разі сильного вітру не злетіти), а якщо взяти цілу групу предметів (купу цегли), то вона вже набуває неабиякої цінності: з цегли, наприклад, можна побудувати гараж для машини, або, якщо дуже постаратися, то й цілий будинок. З однієї цеглини, навіть якщо сильно забажати, гараж не побудуєш. Те саме і з загорожею — одна штахетина — це не загорожа, загородити нею щось важко.

Ситуації, коли необхідна група подібних предметів, трапляються не тільки в повсякденному житті, а й в «електронному» світі. Припустимо, нам дали завдання з обробки великих обсягів одноманітної інформації, наприклад, відсортувати за алфавітом список учнів класу. Спробуємо написати програму, що робитиме це за нас.

Відкрийте Visual Studio, клацніть кнопку **New Project** і створіть проект Sort. Двічі клацніть форму `Form1`, щоб перейти до редагування коду, призначеного події `Form1_Load`. Усі подальші команди пишіть тут — вони виконуватимуться під час відкриття форми. Саму форму поки що залиште порожньою. Для виконання нашого завдання ім'я кожного учня можна зберігати в окремій змінній. Нехай у вашому класі п'ять учнів. Тоді потрібно п'ять змінних:

```
Dim name1 As String
Dim name2 As String
Dim name3 As String
Dim name4 As String
Dim name5 As String
```

Тепер введіть дані:

```
name1 = InputBox("Введіть перше ім'я:","Введіть поточне ім'я
name2 = InputBox("Введіть друге ім'я:","Введіть поточне ім'я
name3 = InputBox("Введіть третє ім'я:","Введіть поточне ім'я
name4 = InputBox('Введіть четверте ім'я:","Введіть поточне ім'я
name5 = InputBox("Введіть п'яте ім'я:","Введіть поточне ім'я
```

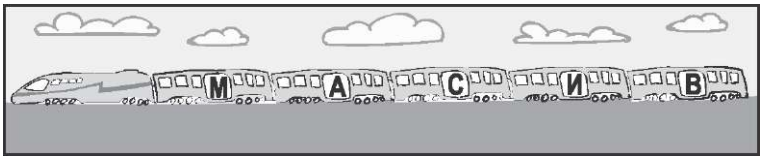
Отже, список учнів ви ввели. Як його відсортувати, — розберемося пізніше. Цього разу вам пощастило з класом, бо він маленький. А якщо необхідно буде ввести імена всіх учнів школи? Скільки тоді рядків буде у вашій програмі? І чи не здається вам, що ви пішли не в той бік? Розібратися у цих питаннях допоможуть *масиви*.

Оголошення масивів

Масив у програмуванні — це пронумерована множина елементів, які мають однаковий тип. Доступ до кожного елемента здійснюється за його порядковим номером, який називають

індексом. Зауважте також, що змінення одного елемента ніяк не впливає на інші.

Можна уявляти масив як потяг, у кожному вагоні якого зберігається певна інформація.



Для використання масиву, як і простої змінної, його спочатку слід оголосити. Оголошення масиву відрізняється від оголошення змінної тим, що після імені масиву потрібно в дужках вказати індекс останнього елемента:

```
Dim ім'яМасиву(ЗакінчуючиНомером) As ТипЕлементівМасиву
```

Можна також вказувати два індекси — першого та останнього елемнтів:

```
Dim ім'яМасиву(Починаючи номером, ЗакінчуючиНомером)
As ТипЕлементівМасиву
```

Коли ви зазначаєте тільки індекс останнього елемента, перший елемент матиме індекс 0, а не 1. У багатьох мовах програмування це звичний спосіб нумерування елементів, до якого з часом звикаєш. Якщо вам незручно постійно зменшувати номер на 1 (перший елемент розташований у комірці 0, другий — у комірці 1, третій — у комірці 2 і так далі), то просто ігноруйте нульовий елемент і починайте працювати з першого. Тільки не забудьте скрізь дотримуватися цього правила, бо потім довго шукатимете помилки у програмі.
Наприклад, щоб зберегти імена всіх моїх друзів, я створив такий масив.

```
Dim MyFriends(100) As String
```

Не знаю, може, в когось друзів і більше, а в мене їх нараховується приблизно стільки, тому думаю, що такого масиву мені вистачить.

У масиві MyMarks я зберігатиму інформацію про мої оцінки: кількість десяток, одинадцяток і, звичайно, одиниць (сподіваюся, що їх не найбільше). Нульовий елемент я ігноруватиму.

```
Dim MyMarks(12) As Integer
```

А тут можна буде знайти відомості про ціни на всі види шоколаду в нашому місті:

```
Dim ChocoPrices(50) As Single
```

— А звідки ти знатимеш, під яким номером зберігатиметься ціна на ту чи іншу шоколадку? — поцікавилася Даринка.

— Це справді проблема. Отже, вам необхідно зберігати назви всіх видів шоколаду й інформацію про те, скільки коштує кожен із них. Маєте якісь пропозиції? — запитав ВВ.

— Можна створити два масиви, — запропонував Михась.

```
Dim ChocoPrices(50) As Single 'тут ми збережемо ціни
Dim ChocoNames(50) As String  'а тут назви
```

— Гаразд, — відповів ВВ. — Цей спосіб використання масивів називають *паралельними масивами*. У паралельних масивах дані різних типів чітко розрізнені та не можуть сплутатись. Але, коли ви здійснюватиме якісь операції з одним із масивів, добре пам'ятайте, що те саме необхідно робити й з іншим. Кращим вирішенням проблеми є створення власного типу даних. У нашому випадку кожне значення цього типу складатиметься з двох елементів — назви шоколадки та її ціни. Такий тип даних називають *структурою*. Можна оголосити масив структур, наприклад масив пар {назва шоколаду, ціна шоколаду}. Зараз ми не вчитимемось працювати зі структурами — ви ще встигнете опанувати цю тему, якщо вирішите зайнятися програмуванням на професійному рівні.

Щодо оголошення масивів, то це, здається, все, що я хотів сказати. Тепер перейдімо до їх обробки.

Доступ до елементів масиву

Припустимо, ми створили масив. А навіщо він нам, якщо ми з ним нічого не можемо зробити? Ось зараз і розберемось, як

можна обробляти масиви. Відкрийте Visual Studio, клацніть кнопку **New Project** і створіть проект Array. Додайте до форми кнопку. Весь код, що наводиться нижче, записуйте в обробнику події клацання кнопки.

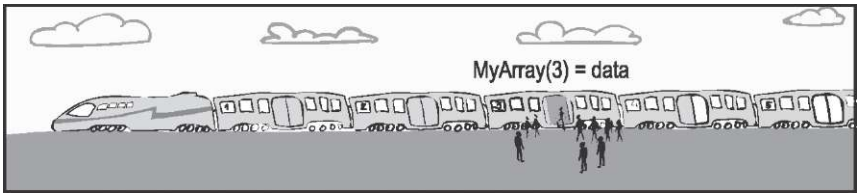
Отже, оголошуємо масив `MyArray`:

```
Dim MyArray(10) As Integer 'створюємо масив, з яким працюватимемо
```

— Що ж далі? — не стримався Михась. — Що необхідно зробити, аби присвоїти значення певному елементу масиву?

— Не поспішай, ми все розглянемо послідовно, — відповів ВВ. — У Visual Basic можна оперувати значенням певного елемента масиву («вагона» поїзда) за допомогою операції індексування: після назви масиву в круглих дужках вказуємо індекс елемента, до якого ми хочемо отримати доступ. Так, `MyArray(0)` — це перший елемент масиву `MyArray`. Фактично вираз `MyArray(0)` є іменем цілочислової змінної й записувати його можна всюди, де дозволено використовувати змінні цілочислового типу. Наприклад, присвойте третьому елементу масиву `MyArray` значення 5:

```
MyArray(3) = 5 'збережемо у третьому елементі масиву значення 5
```



Не забувайте, що в масиві перший елемент має індекс 0. Тепер заповніть значеннями кілька елементів:

```
MyArray(0) = 0
MyArray(1) = 1
MyArray(2) = 2
MyArray(3) = 3
MyArray(4) = 4
```

Ми знову зіткнулися з тим, що необхідно багато писати. Ви не маєте ніяких пропозицій щодо оптимізації цих дій? Правильно! Давайте використаємо цикл.

```
Dim counter As Integer
For counter = 0 To 4
    MyArray(counter) = counter
Next
```

Введіть і виконайте код — результат буде таким самим, як і після виконання п'яти операторів присвоєння. А змінивши 4 на 100, ви виконаєте 101 присвоєння! Неймовірно! Ми знайшли ще одне застосування циклів!

Подумайте, чому, коли в циклі змінну `counter` обмежити значенням 100, буде виконано 101 присвоєння?

Можна присвоювати також значення одних елементів масиву іншим. Наприклад, що, на вашу думку, станеться з масивом після виконання наведеного нижче коду?

```
Dim counter As Integer
For counter = 0 To 4
    MyArray(counter) = counter
Next
For counter = 0 To 3
    MyArray(counter) = MyArray(counter + 1)
Next
```

В режимі налагодження перегляньте, як змінюватимуться значення елементів масиву під час виконання цього коду. Не бійтеся після виконання кожної вправи поекспериментувати із завданням — змінюйте все, що захочете — Visual Studio не допустить серйозних помилок, а ви отримаєте можливість глибше зрозуміти тему.

Введення й виведення масивів

Михась із задоволенням експериментував.

— Я зрозумів, що у масивах можна зберігати різні числові послідовності, наприклад, послідовність квадратів цілих чисел, послідовність чисел, що діляться на 3 або 7... — повідомив він.

— Так, але я не розумію, як нам ввести в масив список друзів, перелік видів шоколаду або кількість різних оцінок, — нагадала Даринка приклади, які наводив ВВ. — Адже в цих

випадках дані елементам масивів має надати користувач, а не програма, як у прикладах, що ми розглядали.
— Зараз із цим розберемося, — відповів ВВ. — Повернімося до моїх прикладів. Як для кого, а для мене друзі важливіші за шоколадки та оцінки, тому зосередимося на прикладі зі списком друзів. Введемо цей список у масив і, після натискання певної кнопки, виведемо вміст масиву.
Як область для введення та виведення даних масиву найзручніше використовувати багаторядкове текстове поле. Створіть новий проект і розмістіть на формі два багаторядкових текстових поля: одне, `TextBox1`, для введення списку друзів, інше — `TextBox2` — для виведення.

Щоб зробити текстове поле багаторядковим, слід у режимі конструювання форми клацнути в полі правою кнопкою миші, вибрати з контекстного меню **Properties** і надати атрибуту `Multiline` значення `True`.

Додайте до форми кнопку, після клацання якої список виводитиметься в полі `TextBox2`. До обробника події клацання кнопки додайте код, що заповнюватиме масив значеннями:

```
Dim friends(100) As String
Dim sep arr() As Char = {}
friends = TextBox1.Text.Split(sep arr, StringSplitOptions.
RemoveEmptyEntries)
```

і код, що виводитиме значення елементів масиву:

```
Dim len As Integer = friends.Length()
For counter = 0 To len-1
    TextBox2.Text = TextBox2.Text + friends(counter) + vbNewLine
Next
```

Розберемося з цими фрагментами програми. Спочатку ви оголосили масив `friends`, в якому можна зберігати 100 рядків — 100 імен друзів. Масив `sep_arr` є допоміжним, у ньому зберігають символи роздільників рядків, але в нашому випадку він може бути порожнім. Подивіться уважніше на присвоєння

```
friends = TextBox1.Text.Split(sep arr, StringSplitOptions.
RemoveEmptyEntries)
```

`TextBox1.Text` — це список імен друзів, тобто один довгий рядок, що складається з окремих імен, розділених символами

кінця рядка. Функція Split перетворює цей довгий рядок на масив рядків, який і присвоюється змінній friends. Параметр StringSplitOptions.RemoveEmptyEntries дозволяє уникнути появи в масиві friends порожніх рядків, які за відсутності цього параметра з'являлися б між кожними двома іменами друзів.

Для виведення масиву в поле ми скористалися циклом, що «накопичує» у змінній TextBox2.Text імена друзів. Список значень масиву матиме кращий вигляд, коли кожен новий елемент буде виведено з нового рядка. Для цього ми використали константу vbNewLine.

— А для чого нам змінна len? — запитав Михась.

— Подивіться уважно, яке значення присвоюється змінній len, — почав пояснювати ВВ. — friends.Length () — це довжина масиву друзів, тобто кількість імен, яку користувач ввів і яку потрібно вивести. Вона й обмежує значення індексу counter у циклі, що формує вміст поля TextBox2.

— А чому ж індекс counter збільшується не до len, а до len-1? — поцікавилася Даринка.

— Згадай, як нумеруються елементи масивів. Якщо в масиві, наприклад, усього 5 елементів, останній матиме індекс 4.

Запустіть програму на виконання. Введіть у поле TextBox1 імена своїх найкращих друзів (кожне ім'я записуйте окремим рядком). Клацніть кнопку. Імена буде відображено в іншому текстовому полі. Якщо вам не зовсім зрозуміло, як працює цей метод, виконайте програму в покроковому режимі.

Написавши, налагодивши та запустивши програму, Михась із Даринкою стали пригадувати всіх своїх друзів.

— За моїми підрахунками, у мене 12 найкращих подруг, — з гордістю повідомила Даринка.

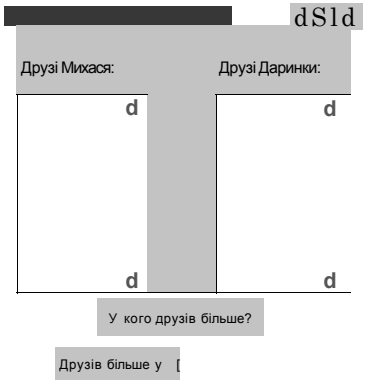
— Ну то й що, а в мене 20 друзів, — сказав Михась.

— Не може бути! Перелічи їх, — не стрималася Даринка.

— Не будемо сперечатися, — втрутився ВВ. — Краще виконаємо вправу, в якій змінимо нашу програму так, щоб вона автоматично визначала, у кого більше друзів.

Вправа 13.1. У кого друзів більше?

Використайте щойно створену форму з полями для введення й виведення двох списків друзів, або створіть таку саму нову форму. Над полем TextBoxl вставте напис Друзі Михася, а над полем TextBox2 — Друзі Даринки. Додайте до форми ще одне однорядкове текстове поле, а ліворуч від нього створіть напис Друзів більше у:. Змініть напис на кнопці Buttonl, щоб форма набула такого вигляду, як показано на рисунку.



Напишіть програму, яка б після натискання кнопки **У кого друзів більше?** у полі **Друзів більше у:** виводила ім'я Михася або Даринки залежно від того, в якому багаторядковому текстовому полі введено більше імен.

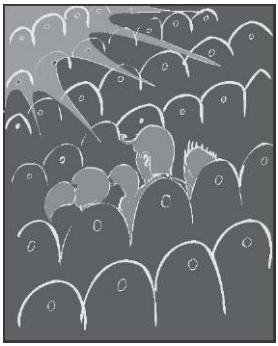
Ви можете використати функцію Length й умовний оператор, а от цикли для цієї вправи не потрібні.

Присвоєння масивів

Неможливо працювати зі змінними, не присвоюючи їх одна одній. Оскільки масиви — це ті самі змінні, постає питання: а чи можна один масив присвоїти іншому? Відповідь на це питання залежить від мови програмування. Якщо буде надана можливість виконувати таку операцію, то під час кожного присвоєння необхідно контролювати, щоб масиви були однакових

розмірів, або змінювати розміри масивів, коли вони різні, а це додасть певної ускладненості роботі програміста.

W Ми розглядаємо лише одновимірні масиви, елементи яких нумеруються за допомогою одного індексу. Але насправді в більшості мов програмування можна працювати з масивами довільної розмірності (з довільною кількістю вимірів). Елементи двовимірних масивів мають два індекси, тривимірних — три тощо. Двовимірний масив можна розглядати як масив масивів, тривимірний — масив двовимірних масивів тощо. Прикладом двовимірного масиву є зал кінотеатру — масив рядів, кожен з яких є масивом місць.



У мові програмування Visual Basic присвоєння масивів дозволене частково: можна присвоювати масиви елементів одного типу, які мають однакову розмірність, але, можливо, різну довжину. У цьому разі довжина масиву, якому присвоюється значення, може змінитися. Розглянемо процес присвоєння на прикладі. Припустимо, що в нас є дві змінні:

```
Dim x As Integer = 4
Dim y As Integer = 1
```

Що станеться, коли змінній `x` присвоїти змінну `y`? Звичайно, значення змінної `x` (це 4) буде видалено, і замість нього буде збережено значення змінної `y` (тобто 1), після чого обидві змінні стануть рівними одиниці. Присвоєння масивів здійснюється у схожий спосіб. Нехай оголошено два масиви:

```
Dim x(3) As Integer
Dim y(2) As Integer
```

Що станеться, коли ми виконаємо те саме присвоєння: `x = y`? Перші два значення, які зберігалися в `x`, будуть заміщені новими значеннями з масиву `y`, а довжина масиву `x` скоротиться на 1. Отже, після виконання операції присвоєння обидва масиви набудуть такого вигляду, який мав масив `y` перед присвоєнням.

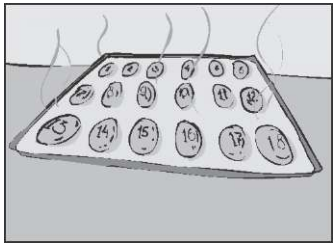
Масиви та їх елементи як параметри підпрограм і функцій

— Ви коли-небудь бачили, як бабуся пече пиріжки? — запитав ВВ.

— Ну, — Михась не знав, що сказати, але не згаяв із відповіддю, — нам більше подобається їх їсти.

— Зрозуміло, що головне для вас — смачний результат, але зараз нас цікавлять дії бабусі. Спочатку вона замішує тісто, потім розкачує його, кладе начинку, заліплює, посипає цукром або родзинками (залежно від того, що більше люблять онуки), а наостанок кладе в піч. Справді, їсти набагато смачніше, ніж описувати процес випікання! А тепер давайте розглянемо цей процес з іншого боку — з точки зору програміста, який має запрограмувати робота, що полегшить роботу бабусі, — вона і так за своє життя напрацювалася.

Відкрийте Visual Studio, клацніть кнопку **New Project** і створіть проект Patty. Додайте кнопку, натискання якої ініціюватиме процес випікання. Змініть назву кнопки на `Випекти пиріжки`. Оголошення змінних і код підпрограм вводьте в оголошенні класу форми, після рядка `Public Class Forml`. Як нам відомо, пиріжки необхідно десь зберігати, щоб вони не зачерствіли. Для цього найлогічніше використовувати масив, кожним елементом якого і буде пиріжок.



```
Dim patties(10) As String 'в кожному елементі масиву
'міститиметься назва начинки пиріжка
```

Оскільки ви виконуєте багато одноманітних дій, реалізуйте кожну з них в окремій підпрограмі.

Начинку пиріжка формуватиме функція `fill_patty`, зміст якої може бути довільним. Головне, щоб ця функція повертала рядок із назвою начинки. Зазначимо, що виокремлення таких

дій у функцію є логічним — завдяки цьому вам не доведеться переписувати код багато разів, якщо ви у процесі розробки програми зміните алгоритм визначення начинки пиріжка. Спочатку використайте найпростіший алгоритм: у випадковий спосіб вибиратиметься одна з чотирьох наперед відомих начинок. Запишіть код функції `fill_patty`, оголосивши перед цим змінну `rnd`, якій присвоюватиметься випадкове число.

```
Dim rnd As System.Random
Function fill patty() As String
   Dim names() As String = {"Капуста", "Сир", "Картопля",
                            "Варення"}
   fill patty = names(rnd.Next(4))
End Function
```

За допомогою функції `fill_patty` начинити 10 пиріжків можна, наприклад, так:

```
Dim counter As Integer
For counter = 0 To 10
   patties(counter) = fill patty()
Next
```

За випікання пиріжка відповідатиме підпрограма `bake_patty`. Вона виводитиме повідомлення про те, що пиріжок із певною начинкою випечено.

— А звідки підпрограма знатиме, з якою саме начинкою випікається пиріжок? — запитав Михась.

— Усе вирішується дуже просто, — відповів ВВ. — Для цього у підпрограми має бути параметр, скажімо `cur_patty`. Його значення й буде назвою начинки:

```
Sub bake patty(ByVal cur patty As String)
   MsgBox("Пиріжок із начинкою " + cur patty + " випечено")
End Sub
```

Під час виклику підпрограми `bake_patty` їй передаватиметься значення елемента масиву `patties`. Підпрограма, аргументом якої є певний елемент масиву, викликається так само, як і підпрограма з аргументом-змінною:

```
bake patty(patties(counter))
```

Весь процес начинення й випікання пиріжків у коді відображатиметься так:

```
Dim counter As Integer
'начинюємо пиріжки
For counter = 0 To 10
   patties(counter) = fill patty()
Next
'випікаємо пиріжки
For counter = 0 To 10
   bake patty(patties(counter))
Next
```

— А якщо ми захочемо зробити не один масив пиріжків? Що тоді? Повторювати один і той самий код кілька разів? — поцікавилася Даринка.

— Чудовим виходом із цієї ситуації є створення підпрограм начинення й випікання масиву пиріжків, — відповів ВВ.

— Але як нам передати до підпрограми цілий масив? — схвильовано запитав Михась.

— Не хвилюйся! І це можливо! — заспокоїв його ВВ, а після цього продовжив: — Ось перед вами код підпрограми начинення пиріжків:

```
Sub fill_patties(ByVal patties() As String)
   Dim counter As Integer
   For counter = 0 To 10
      patties(counter) = fill patty()
   Next
End Sub
```

Підпрограма `fill_patties` отримує цілий масив пиріжків і заповнює їх усі начинкою. Принцип дії підпрограми випікання пиріжків є таким самим, напишіть її самостійно.

Визначення довжини масиву

— Але як функція `fill_patties` може дізнатись, якого розміру масив їй передано? А якщо потрібно випекти не 10 пиріжків, а більше чи менше? Якщо в нас не всі дека для пиріжків одного розміру? — продовжував розпитувати Михась.

— Такі питання рано чи пізно виникають у кожного програміста, — відповів ВВ, — але не кожна мова програмування дає можливість легко вирішити цю проблему. У деяких мовах виникає необхідність передавати в підпрограму другий параметр, який визначає розмір вхідного масиву.

Visual Basic має кілька засобів для знаходження розміру масиву. Обчислюючи кількість друзів, ми з цією метою використали функцію `Length ()` . Але є й інші стандартні функції, що повертають максимальний і мінімальний індекс елемента масиву. Ось вони:

```
UBound(Ім'яМасиву)  'повертає найбільший індекс у масиві
LBound(Ім'яМасиву)  'повертає найменший індекс у масиві
```

Скориставшись цими функціями, перепишемо код підпрограми начинення пиріжків так, щоб вона стала універсальною:

```
Sub fill_patties(ByVal patties() As String)
   Dim counter As Integer
   Dim min As Integer = LBound(patties)
   Dim max As Integer = UBound(patties)
   For counter = min To max
      patties(counter) = fill patty()
   Next
End Sub
```

Тепер програма не видасть помилки, якщо раптом ви захочете спекти 20 чи навіть 100 пиріжків. Отже, ви досягли своєї мети!

Залишилося тільки записати код обробника події клацання кнопки **Випекти пиріжки.** Він може бути таким:

```
Rnd = New Random        ' ініціалізуємо змінну, необхідну для
                        ' визначення начинки пиріжка
                        ' у випадковий спосіб
```

```
fill patties(patties)   ' начинюємо пиріжки
bake patties(patties)   ' випікаємо пиріжки
```

Якщо весь попередній код було введено правильно, програму випікання пиріжків можна запускати на виконання. Смачного!

Повернення масиву функцією

— А якщо ми хочемо, щоб функція повертала масив? — запитала Даринка.

— Справді, резонне питання. Давайте розглянемо й цю проблему, — спокійно відповів ВВ.

Можна перетворити на функцію підпрограму `fill_patties`. Функція `fill_patties` буде повертати масив начинених пиріжків. Тоді відпадає необхідність передавати їй масив порожніх пиріжків як параметр, а також оголошувати його поза функцією. Масив пиріжків створюватиметься й начинятиметься всередині функції, а потім повертатиметься нею як результат:

```
Function fill patties() As String()
   Dim counter As Integer
   Dim patties(10) As String
   Dim min As Integer = LBound(patties)
   Dim max As Integer = UBound(patties)
   For counter = min To max
      patties(counter) = fill patty()
   Next
   fill_patties = patties
End Sub
```

Якщо є така функція, то процес начинення й випікання пиріжків можна реалізувати одним оператором:

```
bake patties(fill patties())
```

Цим рядком можна замінити два останні рядки коду з обробника події клацання кнопки **Випекти пиріжки,** в яких підпрограми `fill_patties` та `bake_patties` викликалися почергово.

— ВВ, будь ласка, давай будемо завершувати, адже тема була дуже важкою, — попросила Даринка. — Крім того, нам необхідно закріпити пройдений матеріал...

— Гаразд, — погодився ВВ, — але я вам дам серйозну вправу для закріплення, якщо вже ви це згадали!

— Добре, давай сюди її швидше! — сказав Михась. — Зараз ми її миттєво виконаємо.

— І куди ви так поспішаєте? — здивувався ВВ. — Ви ще навіть не знаєте, що я вам сьогодні підготував!

— Мабуть, політ в який-небудь повітряний ресторан, — пофантазував Михась.

— Чи заплив у підводний зоопарк, — пожартувала Даринка.

— Ні, я з вами хотів піти в кіберклуб! — заперечив ВВ.

— А що це? — запитав Михась. — Звучить знадливо!

— Та все дуже просто — у нашому часі, на відміну від вашого, було зроблено кілька кроків у напрямку створення віртуальної реальності, яка б не відрізнялася від того, що ми бачимо навколо. Звичайно, є певні проблеми, але є й успіхи. Наприклад, у деякі ігри справді набагато цікавіше грати, живучи там, а не натискаючи кнопки клавіатури. От я й хотів вам показати, як це жити у вигаданому світі.

— Як цікаво! — вигукнули разом Михась і Даринка.

— Ну, якщо так, то ось ваша вправа і завдання для закріплення вивченої сьогодні теми, — засміявся ВВ, а потім додав: — Як ви думаєте, звідки в комп'ютері береться випадковість?

— Може, випадкові значення обчислюються за формулою? — запитав Михась.

— Інколи й за формулою, — погодився ВВ. — Але сучасні програми намагаються спиратися на справді випадкові події.

— Наприклад? — поцікавилася Даринка.

— Наприклад, значення системного таймера, фізичні коливання кулера процесора чи миттєві зміни температури, — відповів ВВ. — Але повернімося до вправи: спробуйте згенерувати у випадковий спосіб масив, скажімо, з 20 елементів, а потім підрахувати їх середнє значення.

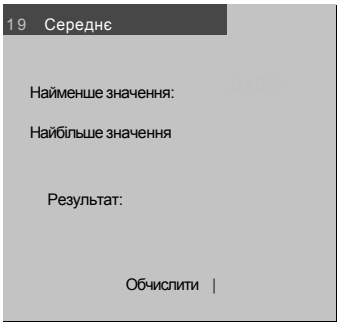
— І суму, — додала Даринка.

— Суму ми обчислимо в будь-якому випадку, — слушно зауважив Михась. — А як же інакше ти збираєшся знаходити середнє значення?

Середнє значення кількох чисел дорівнює їх сумі, поділеній на кількість цих чисел.

Вправа 13.2. Середнє значення

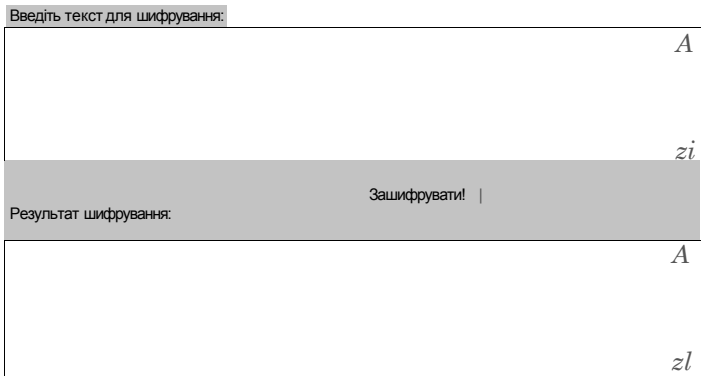
Згенеруйте масив із 20 випадкових чисел і обчисліть їх середнє значення. Створіть форму, на якій буде 3 текстові поля — у два можна ввести найменше та найбільше випадкові значення, а у третьому полі програма виведе результат. Обчислення здійснюватимуться після натискання кнопки.



— Отже, сьогодні залишилося тільки виконати завдання, — завершував урок ВВ. — Уявіть, що ви з друзями домовилися піти серед ночі покататися на циклах. Знаєте, як це класно — навколо темно, немає нікого і тільки ви, цикли та вітер у вухах свище! Так от, оскільки це страшенно небезпечно, мама заборонила вам таким чином розважатись. Але існує цікавий варіант вирішення цієї проблеми — можна тихенько вийти вночі з будинку, щоб вас ніхто не помітив. Інформацію про час та місце зустрічі друзі передадуть за допомогою шифру. А шифрувати й розшифровувати повідомлення буде програма, яку ми зараз напишемо.

Завдання 13. Таємний шифр

Зашифруйте рядок за таким принципом: кожній букві потрібно зіставити певне число (а = 1, б = 2, в = 3 тощо, символу пробілу зіставляється 0), потім це число множиться на 3 і до знайденого добутку додається 2. Отримана послідовність чисел (для її зберігання слід використати масив) записується у зворотному порядку. Створіть форму з двома текстовими полями: у перше користувач вводитиме вихідний текст, а у другому відображатиметься зашифрований результат. Шифрування здійснюватиметься за натисканням кнопки, яку також слід додати до форми.



Додаткове завдання

Напишіть функцію, яка здійснюватиме зворотні обчислення: отримуючи закодований масив як параметр, повертатиме рядок вихідного тексту.

— Дуже добре, що ви виконали завдання. Тепер ми зможемо продовжити нашу прогулянку містом, адже в нас залишився тільки один день, — сказав ВВ.

— А ми ще стільки цікавих місць не побачили, — зауважила Даринка.

— Тоді не будемо гаяти часу, — вигукнув ВВ, і всі троє попрямували до виходу, де на них чекав кодовий замок.

Кодовий замок

1. З чого складається масив?

а) з однакових за значенням елементів;

б) з однакових за типом елементів;

в) з однакових за значенням і типом елементів.

2. Яку кількість елементів містить масив?

а) фіксовану;

б) змінну;

в) невизначену.

3. Розмірність масиву дорівнює:

а) кількості елементів у ньому;

б) кількості індексів, за допомогою яких нумеруються елементи;

в) завжди дорівнює одиниці.

4. Як називають функції, що повертають масив?

а) масивними;

б) складеними;

в) вони не мають спеціальної назви.

5. Який з операторів може бути викликом підпрограми `f`, аргументом якої є масив `m` із п'яти елементів?

а) `f(m);`

б) `f(m());`

в) `f(m(5))`.

6. Якщо в масиві `x` 5 елементів, то оператор `a=x(6)` :

а) надасть змінній `a` значення 0;

б) призведе до помилки під час виконання програми;

в) спричинить помилку під час компіляції.

7. Від'ємне число:

а) у деяких випадках може бути індексом елемента масиву, в оголошенні якого вказано мінімальний та максимальний індекси;

б) у деяких випадках може бути індексом елемента масиву, в оголошенні якого вказано лише максимальний індекс;

в) не може бути індексом елемента масиву.

8. Для присвоєння одного масиву іншому необхідно, щоб:

а) масиви мали однакові довжини;

б) типи елементів масивів були однаковими;

в) масиви мали однакові довжини й типи їх елементів були однаковими.

День 14
Обробка масивів

— Сьогодні останній день вашого перебування в майбутньому, — із сумом промовив ВВ, коли Михась і Даринка зайшли до нього в кімнату.

— Нічого, спробуємо провести його так, щоб він приніс нам не тільки користь, а й задоволення! — підбадьорив його Михась.

— Добре, давайте працювати, а потім розважимося, — погодився ВВ. — Тато казав, що ваше повернення призначене на завтрашній ранок.

— Отже, в нас не так багато часу! Ми зараз швиденько розглянемо останню тему, а тоді спробуємо відпочити так, щоб подорож у майбутнє запам'яталася нам назавжди, — сказала Даринка.

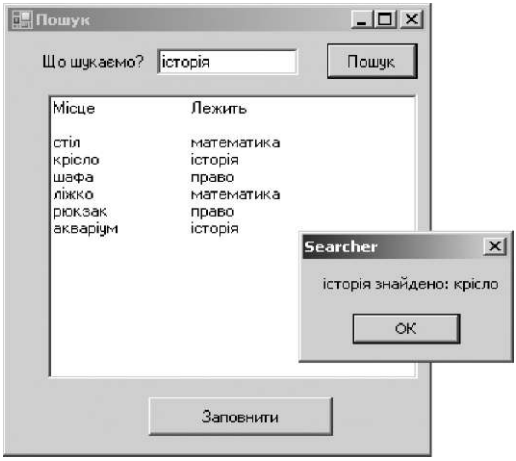
Пошук у масиві

— Я буду здивований, якщо ви мені скажете, що жодного разу нічого не шукали, — розпочав ВВ свій останній урок. — Пригадайте, як зранку, спросоння, запізнюючись до школи, ви не могли знайти зошит з математики, в якому вчора до ночі виконували домашнє завдання. Пригадали? Справді, неприємне відчуття, але мені й самому не раз доводилося його переживати.

А ви ніколи не мріяли, щоб зошит знайшовся сам? І, якщо повністю поринути у мрії, чи хотіли б ви, щоб зошити щоранку шукав робот? Справді, я теж не проти завантажити когось своєю роботою! Отже, давайте займемося цією проблемою — спробуємо створити програму, яка допомагатиме всім школярам світу!

Якщо серйозно замислитися над проблемою пошуку зошита зранку перед уроками (звичайно, краще про це думати звечора, бо зранку немає ні бажання, ні часу), то дійдемо висновку, що кількість місць, де може бути зошит — обмежена. Найімовірніше він буде на столі, під ліжком, за акваріумом, на кріслі, у шафі або в рюкзаку. Отже, зупинимось на цих шістьох варіантах і спробуємо написати програму пошуку зошита.

Відкрийте Visual Studio, клацніть кнопку **New Project** і створіть проект Searcher. Розмітьте форму так, як показано на рисунку. Праворуч від напису **Що шукаємо?** розміщуватиметься поле `TextBox2`, де користувач вводитиме назву шуканого зошита, а знизу — багаторядкове поле `TextBox1`, в якому відображатиметься спосіб розташування зошитів.



Було б логічно всі елементи керування, що стосуються самого процесу пошуку (напис **що шукаємо?,** текстове поле `TextBox2` і кнопку **Пошук),** залишати невидимими доти, доки поле `Text-`

Box1 не буде заповнено назвами предметів. Для цього надайте атрибутам Visible елементів керування, що стосуються пошуку, значення False.

Які масиви нам будуть потрібні? Насамперед масив назв предметів. Ще потрібно позначити місця та спосіб розташування предметів по місцях — усього виходить три масиви:

```
'у масиві places зберігатимуться назви місць
Dim places() As String = {"стіл", "крісло", "шафа", "ліжко",
"рюкзак", "акваріум"}
'у масиві units — назви зошитів, які ми шукатимемо
Dim units() As String = {"математика", "право", "історія"}
'масив placed units відображатиме розміщення зошитів
Dim placed units(UBound(places)) As String
```

Вміст масивів places і units буде задано під час оголошення, а заповнювати масив placed_units (розташовувати предмети по місцях) потрібно після натискання користувачем кнопки **Заповнити.** Отже, в обробнику події клацання кнопки **Заповнити** має виконуватися такий код:

```
'оголошуємо змінну-лічильник
Dim counter As Integer
'створюємо генератор випадкових чисел і заповнюємо
'масив placed units
Dim rnd As System.Random = New Random
For counter = 0 To UBound(placed units)
    'у випадковий спосіб визначаємо номер зошита і кладемо
    'його на місце з номером counter
   placed units(counter) = units(rnd.Next(0, UBound(units)))
Next
'розміщення зошитів відображуємо в полі TextBox1
TextBox1.Text="Місце"+vbTab+vbTab+"Лежить"+vbNewLine+vbNewLine
For counter = 0 To UBound(placed units)
    'місце
   TextBox1.Text=TextBox1.Text+places(counter)+vbTab+vbTab+
    'зошит
   placed units(counter)+vbNewLine
Next
'відображуємо елементи керування пошуком
Label1.Visible = True
TextBox2.Visible = True
Button2.Visible = True
```

Власне пошук виконується в обробнику події клацання кнопки **Пошук.** Значення кожного елемента масиву `placed_units` слід порівняти із вмістом поля `TextBox2`. Якщо вони рівні, відображатиметься повідомлення про те, що певний зошит знайдено на певному місці. Якщо вміст поля `TextBox2` не збігається зі значенням жодного елемента масиву `placed_units`, слід вивести повідомлення про те, що зошит не знайдено. Для відстеження успішності пошуку призначена змінна `found`: якщо зошит не знайдено, її значення становитиме `False`, інакше — `True`.

Наведемо відповідний код:

```
'оголошуємо змінну-лічильник
Dim counter As Integer
'змінна, яка визначає, чи був пошук успішним
Dim found As Boolean
found = False   'спочатку ще нічого не знайдено
'переглядаємо всі елементи масиву
For counter = 0 To UBound(places)
    'якщо знайдено шуканий зошит...
    If placed units(counter) = TextBox2.Text Then
        '... виводимо повідомлення про це
        MsgBox(TextBox2.Text + " знайдено: " + places(counter))
        found = True
    End If
Next
'якщо нічого не знайдено — повідомити про це користувача
If Not (found) Then MsgBox("Зошит не знайдено.")
```

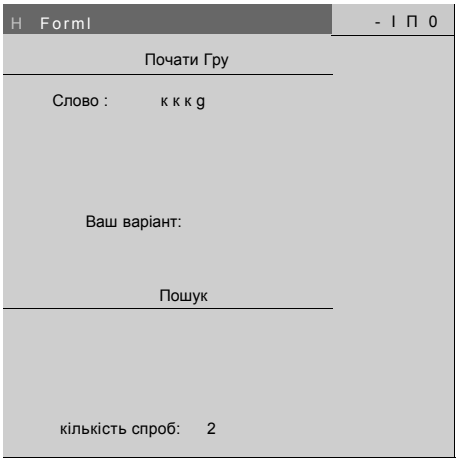
Тепер принцип пошуку в масиві має бути більш зрозумілим — переглядайте всі елементи, поки не знайдете той, що дорівнює певній еталонній змінній. Що ж, виконайте вправу на пошук. Спробуйте створити щось серйозне і цікаве, скажімо, гру «Поле чудес». Ви думаєте, що не справитесь? Справитесь і не з таким!

Вправа 14.1. Поле чудес

Розробіть спрощену гру «Поле чудес». Масив із 5 елементів у випадковий спосіб заповнюватиметься числами від 0 до 9 й відображатиметься спочатку як *****. Користувач намагаєть-

ся вгадати число, вводячи його в текстове поле й натискаючи кнопку **Вгадати.** Якщо одне з чисел вгадано, воно відображатиметься у відповідному полі замість символу «*». Якщо вгадано кілька однакових чисел, усі вони мають відображатися. Після завершення гри буде виведено повідомлення «ви виграли з» і кількість спроб.

Форма гри матиме приблизно такий вигляд, як показано на рисунку. Поки гра не почалася, певні елементи можуть бути неактивними або невидимими. Як ви вважаєте, які саме?



Гра розпочинається натисканням кнопки **Почати Гру,** у результаті якого відображатимуться всі елементи керування та заповнюватиметься еталонний масив. Крім того, вам потрібно записати 0 у поле **кількість спроб.** Вдалим рішенням буде створення й другого масиву, наприклад `visible`, елементи якого матимуть тип `Boolean` і визначатимуть, чи вгадано вже те чи інше число. На початку гри всім елементам масиву `visible` слід надати значення `False`.

Обробка клацання кнопки **Пошук** буде дещо складнішою. Перш за все необхідно буде перевірити, чи ввів користувач щось у текстове поле. Якщо ввів, слід збільшити кількість спроб і порівняти введене користувачем значення з кожним елементом еталонного масиву чисел. За умови, що введене користувачем

значення збігатиметься зі значенням певного елемента еталонного масиву, відповідному елементу масиву visible слід надати значення True.

Також після кожної спроби потрібно оновлювати вміст текстового поля TextBox1. Це можна зробити, наприклад, таким чином (array — еталонний масив чисел):

```
TextBox1.Text = ""        'робимо поле TextBox1 пустим
For counter = 0 To 4      'переглядаємо масив visible
   If visible(counter) Then 'якщо черговий елемент видимий,
                            'виводимо в полі число
      TextBox1.Text = TextBox1.Text + Str(array(counter))
   Else 'інакше виводимо *
      TextBox1.Text = TextBox1.Text + "*"
Next
```

Нарешті потрібно перевірити, чи містяться ще значення False в масиві visible. Якщо ні, то час завершувати гру.
Така перевірка є теж задачею з пошуку (шукаємо значення False в масиві visible). Для її розв'язання можна не створювати окремий цикл, а модифікувати той, що наведений вище.

Для пошуку можна використовувати стандартні функції Visual Basic. Наприклад, можна скористатися функцією Array.IndexOf(), що повертає індекс елемента, який ви шукаєте:

```
Dim Arr(10) As Integer
Dim what As Integer = 50
MsgBox(Str(what)+" is on "+Str(Array.IndexOf(arr,what)))
```

Інші вбудовані функції можна переглянути у списку автовведення, який буде відображено засобом IntelliSense після введення слова Array. (з крапкою).

Сортування масиву

Ви коли-небудь замислювалися, чому список учнів класу відсортований за алфавітом? А чому мама так любить, коли всі зошити та книжки лежать на своїх місцях? На мою думку, це полегшує пошук. Наприклад, коли вчитель запитує вірш, який вам потрібно було вивчити напам'ять, йому легше рухатися за списком, що складений в алфавітному порядку (вам пощастить, якщо ваші прізвища міститимуться у другій його

половині). Те саме можна сказати і про робоче місце: зошит чи ручку знайти легко, якщо вони лежать на своїх місцях. Отже, давайте повернемося до ситуації зі зборами до школи. Якби у вас на столі був порядок, то кількість місць, де могли б лежати потрібні вам книжки і зошити, значно скоротилася — навряд чи хтось захоче добровільно і наполегливо зберігати зошит з математики за акваріумом чи під ліжком.

У подібній ситуації ви опиняєтесь, коли шукаєте певне слово у словнику. Для цього не потрібно переглядати кожну його сторінку, бо ви точно знаєте: якщо слова немає на відповідному місці (всі слова розташовано за алфавітним порядком), то в цьому словнику воно відсутнє взагалі. Уявіть собі словник, в якому інформація не відсортована. Щоб знайти потрібне слово, вам доведеться переглядати кожну його сторінку! Ну, просто жах, а не словник — від нього жодної користі.

Отже, сортування є одним із найважливіших методів обробки масивів. Розберімося, який алгоритм криється під цим словом, «сортування». Ви побачите, що алгоритм надзвичайно простий, багато з нас його застосовують у повсякденному житті, навіть не помічаючи цього.

Розглянемо масив:

```
6 4 12 10 1 8 3
```

Як бачите, він не відсортований. Масив, відсортований у порядку зростання, виглядатиме так:

```
1 3 4 6 8 10 12
```

«Відсортований у порядку зростання» означає, що кожен елемент масиву більший за попередній, а «у порядку спадання» — менший:

```
12 10 8 6 4 3 1
```

Як ми можемо надати вихідному масиву відсортованого вигляду? Перш за все зауважимо, що найменше число має стояти

першим (якщо сортувати в порядку зростання). Щоб забезпечити це, слід найменший елемент (у нашому масиві це 1) поміняти місцями з першим елементом (число 6):

```
1  4  12  10  6  8  3
```

Проаналізуймо масив, який у нас вийшов: число 1 уже відсортоване — воно стоїть на своєму місці, а тому тепер на нього можна не звертати уваги і проводити сортування інших чисел так, ніби масив починається з другого елемента (число 4). Тому повторіть ті самі дії, які ви робили, але вже з меншим масивом. У ньому найменшим числом є 3, а першим елементом — 4. Поміняйте їх місцями:

```
1  3  12  10  6  8  4
```

Аналогічні дії необхідно повторювати, доки не дійдемо до кінця масиву:

```
1  3  4  10  6  8  12
1  3  4  6  10  8  12
1  3  4  6  8  Ц  12
1  3  4  6  8  10  12
```

Так само просто впорядковувати масиви символів, рядків, дат і багатьох інших об'єктів. Для закріплення вивченого методу виконайте вправу, а тоді ми перейдемо до ще цікавіших речей.

Вправа 14.2. Впорядковуємо набір чисел

Відсортуйте наведений нижче набір чисел у зростаючому та спадному порядку, занотовуючи кожен свій крок.

3, 12, 56, -1, 55, 0, 17, -66, 10, 17

Для сортування у Visual Basic є вбудована підпрограма `Array.Sort()`, яка сортує масив у зростаючому порядку. Наприклад:

```
Dim Arr(10) As Integer
Arr(3) = 6
Array.Sort(Arr)
```

Пошук у відсортованому масиві

Отже, масив ми відсортували. Що ж тепер? Спробуємо розробити алгоритм пошуку у відсортованому масиві. Для того щоб знайти певне значення в невідсортованому масиві, доводиться переглядати всі його елементи. У відсортованому масиві це можна зробити значно швидше. Ви запитаєте, як? Спробуємо розібратися з цим.

Пригадайте приклад зі словником. Ви погодитесь із тим, що слово «школа» недоречно шукати на початку словника, бо в будь-якому словнику слова розташовані в алфавітному порядку (слова, що починаються на букву «ш», містяться ближче до кінця). Те саме можна сказати і про числові відсортовані масиви. Згадаємо масив, який ми відсортували:

```
1 3 4 6 8 10 12
```

Знайдемо в цьому масиві число 4, прагнучи виконати якомога менше кроків. Розділимо масив навпіл. Посередині масиву розміщено число 6. Де може розміщуватися число 4 відносно 6, праворуч чи ліворуч? Зрозуміло, що тільки ліворуч! На цьому і ґрунтується пошук.

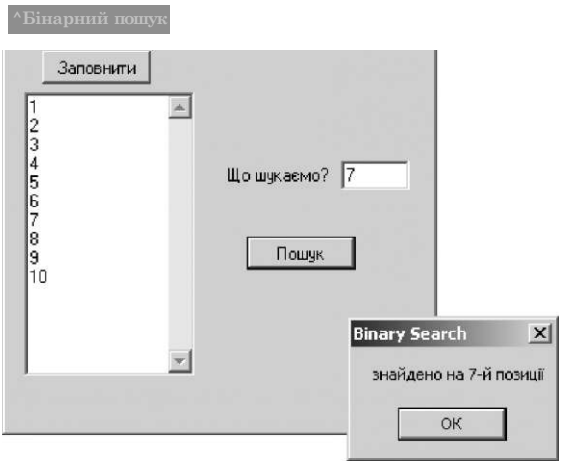
Отже, з'ясувавши, що число 4 розміщене в лівій частині масиву, обмежуємо пошук цією частиною:

```
1 3 4 6
```

Середнім індексом елементів цього масиву буде 1,5 (не забувайте, що нумерацію необхідно починати з 0). Заокруглюємо індекс до меншого значення, оскільки дробових індексів не буває. Отже, серединою масиву є число 3. Оскільки 4 > 3, то обмежуємо пошук правою частиною масиву:

```
3 4 6
```

Середнім у цьому масиві буде елемент 4. Результат отримано! Сподіваюся, що принцип пошуку зрозумілий, і ми можемо перейти до кодування. Створіть новий проект і розробіть форму, яку показано далі на рисунку.



Під час клацання кнопки **Заповнити** створюватиметься масив, який заповнюватиметься відсортованими значеннями. Зробимо це в такий спосіб:

```
Dim c As Integer
For c = 1 To 10
    array(c) = c
Next
```

Після створення масиву, а також після кожного кроку пошуку масив варто відображувати в текстовому полі. Нижче наведено процедуру пошуку:

```
'верхня межа
Dim up As Integer = UBound(array)
'нижня межа
Dim down As Integer = LBound(array)
'тимчасова змінна для середнього індексу
Dim tmp As Integer
'шукане значення
Dim what As Integer = Val(search.Text)
'змінна, що визначатиме, чи завершено пошук
Dim found As Integer = -1
'поки found = -1, здійснюємо пошук
While found = -1
```

```
'визначаємо середину
   tmp = (up + down) / 2
'вибираємо область, де має бути шуканий елемент.
   If array(tmp) > what Then
      up = tmp - 1
   Else
      down = tmp + 1
'якщо права межа «заскочила» за ліву, то...
   If up < down Then
      If array(up) = what Then
         found = tmp   '... або значення знайдене,
      Else
         found = -2    '... або його в масиві немає
End While
'обробка результату
If found <> -2 Then
   MsgBox("знайдено на" + Str(found) + "-й позиції")
Else
   MsgBox("К6 знайдено")
```

Бінарний пошук (саме його ви щойно запрограмували) реалізований у вбудованій функції `Array.BinarySearch()`. Не забувайте, що бінарний пошук відбувається на вже відсортованому масиві, тому перед його застосуванням масив необхідно відсортувати. Наприклад:

```
Dim Arr(10) As Integer
Dim what As Integer = 10  'що будемо шукати
Array.Sort(arr)  'не забувайте, що масив має бути
                 'відсортований
MsgBox(str(what)+ " is on " +
                      Str(Array.BinarySearch(arr,what)))
```

— Як цікаво! — із захопленням вимовив Михась. — Виявляється, що алгоритм бінарного пошуку люди вигадали задовго до появи комп'ютерів, тоді, коли винайшли перший словник.

— До речі, у давньому світі було винайдено й багато інших алгоритмів, — зауважив ВВ.

— Але це було так давно! Тепер, мабуть, якісь розумники вже здогадалися, як шукати інформацію ще швидше? — з цікавістю продовжував розпитувати Михась у ВВ про алгоритм бінарного пошуку.

— А от і ні. Пошуку, швидшого за бінарний, не існує. Та цього не скажеш про алгоритм сортування, який ми вивчали (його називають сортуванням шляхом пошуку мінімального елемента). На зорі комп'ютерної ери люди вигадали численні алгоритми сортування масивів і деякі з них залишають далеко позаду розглянутий нами метод. На мою думку, ви не пошкодуєте, якщо витратите час, щоб ознайомитися зі швидкими методами сортування та запрограмувати їх. Але спочатку ми маємо запрограмувати хоча б найпростіший метод — у цьому й полягає сьогоднішнє завдання.

Завдання 14. Власна програма сортування

Напишіть програму, яка сортуватиме в алфавітному порядку набір слів, уведений користувачем у багаторядкове текстове поле. Для цього скористуйтеся методом сортування, який ми сьогодні розглядали. Програма, що його реалізує, буде містити вкладені цикли: лічильник зовнішнього циклу вказуватиме на кінець відсортованої ділянки масиву, а у внутрішньому циклі здійснюватиметься пошук мінімального елемента з числа ще невідсортованих.

Не забувайте також, як вводити та виводити масиви за допомогою текстових полів — це ми вивчали на попередньому уроці. Крім того, вам потрібно буде поміняти місцями значення елементів масиву. Ця операція здійснюється так само, як і обмін значеннями двох змінних.

Додаткове завдання

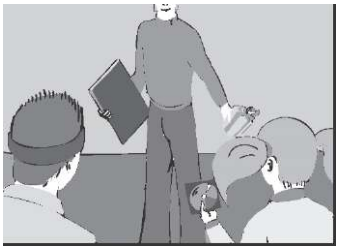
Для того щоб оцінити, наскільки швидким є той чи інший алгоритм обробки масивів, його потрібно застосувати до масиву, що містить n елементів (n — деяке ціле число, але невідомо, яке саме). Кількість операцій, які виконує алгоритм, можна

оцінити формулою, що залежить від n, наприклад, n/2, n^2, n^3 тощо. Знайдіть формули, що визначають такі величини:

1. Найбільшу кількість порівнянь, необхідну для пошуку заданого значення в невідсортованому n-елементному масиві.
2. Найбільшу кількість обмінів елементів, необхідну для сортування n-елементного масиву методом пошуку мінімального елемента.
3. Найбільшу кількість порівнянь, необхідну для знаходження заданого значення у відсортованому n-елементному масиві методом бінарного пошуку.

— Ну, от і все, — сказав ВВ, закриваючи середовище Visual Studio. — Сподіваюся, ми не марно витратили стільки часу.

— Звісно, ні! — підтвердив Михась.



— Знання з майбутнього зроблять нас геніями минулого! — пожартувала Даринка.— До речі, ми тобі подаруночок приготували. У Михася на комп'ютері було кілька наших родинних фотографій, от ми і створили програму для їх перегляду. — Коли згадаєш про нас, то запускай її.

— Дякую, друзі! — зрадів ВВ. — А я вам також дещо хочу подарувати! Тобі, Михасю, дарую ось цей новенький ноутбук. Гарантую, що ні в кого такого не буде! Даринці я підібрав парфуми, які сподобаються всім, а подружки будуть заздрити і шукатимуть по крамницях такі самі, але не знайдуть, бо їх виробляють лише в нашому часі.

— Ну, а тепер, коли з подарунками розібралися, давайте проведемо решту часу так, щоб ми цей день запам'ятали на все життя! — вигукнув Михась.

— Ну, що ж, нехай буде так, то чим ми займемося? — запитав, як завжди, ВВ.

Останній день свого перебування в майбутньому Михась і Даринка провели просто незабутньо. Вони гралися в кіберігри, літали на аероциклах, гуляли містом, пригадуючи ці 12 днів, які вони

ніколи не забудуть. А ввечері прийшов батько ВВ, ВВ/548, і разом із ним друзі за кілька годин здійснили навколосвітню подорож, зробивши купу різних фотографій, які забрали додому. Вранці, після прощання, всі вирушили до машини часу. Але на виході з дому на них чекав, як завжди, невеличкий сюрприз — кодовий замок.

— Як мені не вистачатиме цих тестів удома, — з іронією сказав Михась.

— Так, — погодилася з ним Даринка, і всі весело засміялися, а потім разом почали відповідати на питання цього незвичайного замка.

Кодовий замок

1. Під час пошуку в невідсортованому масиві перегляд елементів:
   а) має здійснюватися зліва направо;
   б) має здійснюватися справа наліво;
   в) може здійснюватись у довільному порядку.
2. Як сортування в порядку зростання впливає на індекси елементів масиву?
   а) індекси можуть тільки збільшуватися;
   б) індекси можуть тільки зменшуватися;
   в) індекси можуть як збільшуватися, так і зменшуватися.
3. Що не можна зробити за допомогою сортування?
   а) змінити розмір масиву;
   б) дізнатися значення мінімального елемента масиву;
   в) спростити пошук у масиві.
4. За наявності в масиві двох однакових елементів програма сортування:
   а) скоротить масив на один елемент;
   б) обробить цей масив, як і будь-який інший;
   в) не спрацює через помилку.

5. У програмній реалізації якого алгоритму застосовуються вкладені цикли?

а) пошук у невідсортованому масиві;

б) сортування масиву методом пошуку мінімального елемента;

в) бінарний пошук у відсортованому масиві.

6. Задано два рядки коду:

```
X = Array.BinarySearch(MyArray,what)
Y = Array.IndexOf(MyArray,what)
```

Що можна стверджувати щодо значень змінних `X` та `Y`?

а) вони завжди рівні;

б) вони завжди різні;

в) вони можуть бути як рівними, так і відмінними.

7. Необхідно знайти задане значення в невідсортованому масиві. Який із вказаних способів пошуку швидший?

а) пошук прямим перебором;

б) сортування, а потім — бінарний пошук;

в) швидкість вказаних способів однакова.

8. Яку кількість порівнянь слід виконати під час бінарного пошуку в 100-елементному впорядкованому масиві?

а) 100;

б) не більше 50;

в) не більше 7.

Попрощавшись, друзі врешті-решт зайшли до просторої кімнати. В ній стояв прилад, який «знав», куди та в яку саме мить їх потрібно повернути. Подорожі в часі вже були дослідженим явищем, тому це не було чимось неймовірним. Коли механізм запустився, все навколо замиготіло, і друзям здалося, що по щоках ВВ/550 потекли сльози, але це, мабуть, їм привиділося.

— Ми ще зустрінемося! — вигукнула Даринка, але не знала точно, чи почув ці слова ВВ, бо наступної миті діти вже

стояли в кімнаті Михася. Відтоді, як вони вирушили до майбутнього, минуло лише 15 хвилин і 34 секунди.

— Мені їх не вистачатиме, — промовив Михась.

— Кого? — здивувалася Даринка.

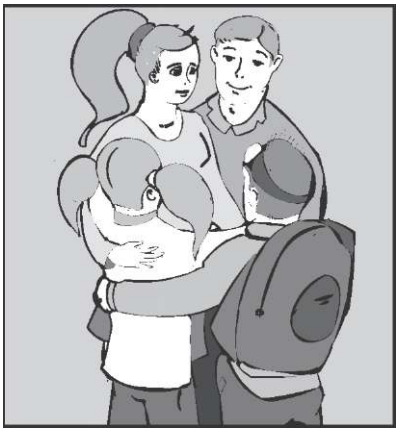
— Аероциклів, комп'ютерів, роботів... — з ностальгічними нотками в голосі відповів Михась.

— А ВВ? — підвищила голос Даринка. — Ти про нього навіть не згадав!

— Його, звичайно, також! — відповів їй Михась. — Хіба ти не будеш за ним сумувати?

— Діти, діти, чому ви не спите!? — запитала заспана мама, яка ввійшла до них у кімнату.

— Мамусю!!! — вигукнули обоє, і кинулися до неї на шию, миттєво забувши всі суперечки.



— Що з вами? Що сталося? Ви щось зламали? — злякано спитала мама. — З якого приводу така радість?

— Мамусю, мамусю, ми просто дуже раді тебе бачити, — відповіла Даринка.

Наступного дня, під час сніданку, батько розповів їм про те, що до лабораторії вдерлись якісь злодії, але нічого не вкрали. Михась із Даринкою лише таємниче переглянулись і посміхнулися.

Навчальне видання

Ігор Олександрович Завадський
Ростислав Ігорович Заболотний

ОСНОВИ ВІЗУАЛЬНОГО ПРОГРАМУВАННЯ

Керівник проекту І.О. Завадський
Редактор В.М. Бабійчук
Коректор В.М. Бабійчук
Комп'ютерна верстка З.В. Лобач

ТОВ «Видавнича група BHV»
Свідоцтво про внесення до Державного реєстру
суб'єктів видавничої справи України
серія ДК №175 від 13.09.2000 р.
Підписано до друку 06.06.07. Формат 60x84 1/16. Папір офсетний.
Гарнітура ShoolBook, Pragmatica. Друк офсетний.
Ум. друк. арк. 15,81. Обл.-вид. арк. 14,72.
Наклад прим. Зам. № 567.

Виготовлено в ТОВ «Освітня книга»,
м. Київ, вул. Орловська, 2/7, оф. 6.
Свідоцтво про внесення до Державного реєстру
суб'єктів видавничої справи України
серія ДК №2245 від 26.07.2005 р.