ВІД PASCAL ДО C#

Шевчук П.Г.

Анотація. У статті розглянуто один із можливих підходів до навчання програмування мовою C#, орієнтований на вчителів і учнів, які вже програмували мовою Pascal. Вказано подібні й відмінні риси цих мов. Ілюструється використання мови C# для написання найпростіших програм для консольного виконання. Наводяться приклади розв'язування задач мовами програмування Pascal і C#. Вказано суттєві особливості написання програм мовою C#. Розглядаються деякі принципи об'єктно-орієнтованого програмування. Наведено застереження щодо можливих помилок, до яких може призвести навчання програмування мовою C# з огляду на набуті знання й уміння програмування мовою Pascal.

Ключові слова: програмування, мова C#, мова Pascal, оператори, консоль, присвоєння, розгалуження, цикли, масиви.

* * *

Мова програмування C# набуває все більшої популярності. Її освоюють не лише початківці, а й ті, хто вже має певні знання й навички в галузі програмування. Нові мови програмування частіше всього створюються на основі вже наявних і широко розповсюджених. C# не лише створювалась з урахуванням особливостей надзвичайно популярних і широко розповсюджених мов C++ і Java, а й призначалася для вивчення фахівцями, які вже знають мови, схожі до «C». Поряд із цим у навчанні програмування початківців найбільше застосовується мова програмування Pascal. Перехід від мови Pascal до мови C# вимагає звикання до дещо іншого синтаксису побудови програм. Серед мов програмування, що мають так званий «C-подібний» синтаксис, мова C# одна з найбільш схожих на Pascal. Насамперед це пояснюється тим, що творцями мови програмування C# є люди, які свого часу були причетні до становлення й розвитку середовищ програмування мовою Pascal [1].

Вивчення нового шляхом порівняння з наявним — це не завжди якісний метод пізнання. Такий підхід може зашкодити усвідомленню нових особливостей на користь недоліків, які нововведення намагається подолати. У цьому випадку знайомство з об'єктно-орієнтованою мовою програмування C# шляхом розв'язування завдань процедурного програмування не лише не розкриває можливостей мови C#, а й дещо суперечить принципам її застосування. Проте для фахівців, що вже володіють процедурними мовами програмування це, напевно, єдиний спосіб швидко розпочати використовувати об'єктно-орієнтоване середовище. Актуальність дослідження полягає в ознайомленні з мовою C# шляхом розв'язування задач, що традиційно використовуються під час навчання програмування мовою Pascal. Це здійснюється не лише для розкриття відмінностей між мовами, а й для запобігання появі можливих помилок під час вивчення мови C# на основі набутого раніше досвіду.

Проблеми навчання програмування в курсі шкільної інформатики, методика і зміст такого навчання активно досліджуються вітчизняними вченими (В.В. Бондаренко, Я.М. Глинський, М.І. Жалдак, Р.І. Заболотний, І.О. Завадський, Т.П. Караванова, Н.В. Морзе, Ю.С. Рамський, В.Д. Руденко, О.М. Спірін). Знайомству з мовою C# розробників, які вже мають досвід програмування мовами із C-подібним синтаксисом, присвячено роботи Вячеслава Понамарьова, Герберта Шилдта, Карли Ватсона, Миколи Секунова, Метью Мак-Дональда та Маріо Шпушта, а також інших, переважно зарубіжних авторів. Поряд із цим вітчизняними науковцями недостатньо висвітлено проблему освоєння нових мов програмування на основі раніше вивчених, практично відсутні роботи щодо навчання програмування мовою C# в загальноосвітніх навчальних закладах.

Мета статті — розробити педагогічно виважений підхід до навчання програмування мовою C# на основі відомостей про мову Pascal.

Будь-яке програмне забезпечення, а також те, що використовується в навчанні, дуже швидко морально застаріває і потребує своєчасного оновлення. Перелік мов і середовищ програмування, що використовуються для навчання учнів загальноосвітніх шкіл, постійно розширюється. Однією з таких мов є C#. Ця мова дозволяє знайомити учнів з об'єктно-орієнтованим програмуванням, дозволяє створювати програми як для консольного, так і для віконного виконання. Є низка середовищ, що підтримують програмування мовою C# для різних операційних систем.

Опанування нових знань має передбачати використання набутих. Багато вчителів уже мають досвід навчання програмування мовами Basic, C++, Pascal, Delphi. Вони можуть використовувати нову мову програмування для розв'язування вже відомих задач. Оскільки мова Pascal — нині найпоширеніша в навчанні, то і випадки переходу від цієї мови до мови C# трапляються найчастіше.

Мова програмування C#

Мова C# є складовою програмної платформи Microsoft.NET, яку компанія Microsoft розробила в кінці 1990-х років. Автором мови є Андерс Хейльсберг, відомий раніше як творець компілятора Borland Turbo Pascal і провідний конструктор технології Borland Delphi. Створення мови не проходило шляхом розроблення нових наборів команд і правил побудови синтаксису. Як фундамент C# використано наявні мови, основна увага була зосереджена на покращеннях й інноваціях. C# має багато спільного з широко вживаними і популярними мовами програмування C, C ++, Java та PHP [1]. На цей час практично всі професійні програмісти використовують ці мови, тому перехід до C# відбувається без особливих труднощів. Незначні запозичення мова C# отримала і від мови Pascal.

Те, що мова C# має свої витоки в багатьох найрозповсюдженіших мовах програмування, дозволяє стверджувати, що компетентності, набуті під час оволодіння мовою C#, знадобляться для опанування інших мов програмування.

Середовища програмування

Для вивчення мови програмування Pascal нині використовують такі середовища, як Turbo Pascal версій 5.0 та 7.0, Free Pascal різних версій, Pascal ABC, Pascal ABC.Net, Delphi та інші. Деякі з них, наприклад Turbo Pascal, розроблені для морально застарілих операційних систем типу DOS і не підтримують багатьох можливостей використання сучасних програмно-апаратних платформ.

Поряд із цим є значна кількість середовищ розробки мовою C#, що працюють під управлінням сучасних операційних системи Windows XP, Windows Vista, Windows 7. Створювати програми мовою C# можна і для вільно розповсюджуваної операційної системи Linux і малопоширеної на Україні Mac OS. Є середовища програмування, що дозволяють розробляти такі програми для дещо застарілих операційних систем Windows 9x [2], [3].

До складу Microsoft Visual Studio, багатофункціонального пакета сучасних середовищ програмування, входить одна з найкращих платформ для розробки програм мовою C#. Професійна версія Microsoft Visual Studio Professional передбачає 90-денне безоплатне використання. Поряд із цим є повністю безкоштовні, так звані «експрес» версії середовищ програмування зі складу Microsoft Visual Studio. До них належить Microsoft Visual C# Express Edition, яка відрізняється від професійної версії лише незначним обмеженням функцій і неповними бібліотеками класів.

Цікавим, з огляду на навчальне використання, є середовище розробки мовою C# з відкритим кодом — Sharp Develop, інтерфейс якого багато в чому подібний до інтерфейсу Microsoft Visual Studio [4].

Для операційних систем, відмінних від Windows, зокрема Linux, створено аналог платформи .Net Framework, що отримав назву «Mono». Також для програмної платформи «Mono» розроблено Mono Develop — кросплатформене, вільно розповсюджуване середовище програмування. Нещодавно платформа Mono і середовище Mono Develop отримали власні реалізації і для операційної системи Mac OS.

За нетривалу історію існування мови C# було розроблено ще багато інших середовищ програмування, деякі з них уже встигли морально застаріти. Поряд із цим для всіх операційних систем сімейства Windows, починаючи від Windows 98, є можливість розробляти програми з використанням будь-якого, навіть найпростішого, текстового редактора. Для цього достатньо, щоб у системі було встановлено програмну платформу Microsoft .Net Framework [3].

Консоль і робота з нею

Мова C# дозволяє створювати програми як для віконного, так і для консольного виконання. Консольний інтерфейс найпростіший і підтримується практично всіма операційними системами. Для багатьох операційних систем він взагалі лежить в основі всіх інших способів організації діалогу з користувачем. Програми, що використовують консольний інтерфейс, зручні для навчання програмування. Оскільки традиційно саме в консольному варіанті створюються програми мовою Pascal, то розробка таких програм дозволяє одночасно розглядати розв'язування одних і тих же задач з використанням обох мов.

Найпростіша програма

Ознайомлення з мовою програмування часто розпочинається з написання програми, що носить умовну назву «Hello World!». Ця програма містить одну єдину команду виведення на екран повідомлення з вітанням до всього світу.

Проте для більшості мов програмування учням можна продемонструвати ще простіший код, що не містить жодних команд, не виконує жодних дій, але й під час компіляції не викликає помилок. Для мови Pascal найкоротший фрагмент коду, що можна запустити на виконання (відкомпілювати), містить всього два слова: Begin End і крапку в кінці. Жартома можна назвати такий код «програмою, що тільки й те робить, що нічого не робить». Вилучення з тексту такої «програми» будь-якого символу викличе помилку компіляції. Уже на такій «програмі» можна вчити зберігати і відкривати файли, запускати код на компіляцію й виконувати інші найпростіші дії, що потрібно знати учневі на перших етапах навчання програмування. Також випадає нагода наочно пояснити деякі теоретичні питання. Наприклад, учням можна поставити запитання: «Чи є код, що не виконує жодних дій програмою?». Відповідь навіть на таке просте запитання дозволяє краще з'ясувати означення понять алгоритм, програма, компіляція, виконання програми.

Принциповою відмінністю мов програмування Pascal і C# є те, що вони по-різному працюють з прописними і рядковими символами алфавіту. Якщо для Pascal великі й малі букви ідентичні, і зміна регістру використовується лише для надання тексту програми більшої зрозумілості, то в мові C# великі (прописні) і маленькі (рядкові) літери — це символи, що по-різному ідентифікуються компілятором. Тобто «A» та «a» в командах мови Pascal сприймаються як один і той же символ, а в мові C# — це два різних символи.

Ось можливий текст «найпростішої програми» мовою C#: *class FirstClass {static void Main( ) {}}* Цей набір символів також немає жодного практичного значення, окрім того, що код без проблем компілюється. На відміну від мови Pascal, написання навіть найпростіших програм мовою C# потребує створення хоча б одного класу і методу в ньому. Це обов'язково потрібно якось пояснювати учням. Така необхідність дещо ускладнює використання мови C# для навчання програмування початківців. Проте додаткові пояснення можуть бути досить загальними і не обов'язково точними. Наприклад, учням можна пояснити, як на сучасному комп'ютері виконується одночасно багато програм, що обробляють значний обсяг даних. Об'єкт — це поєднання даних і програм, що ці дані використовують. Кожен об'єкт належить до свого класу. Створюючи навіть

найпростішу програму, необхідно створити новий клас, у межах якого і буде виконуватися програма як самостійний об'єкт обчислювальної системи. Учням корисно розказати і продемонструвати те, що одночасно може існувати (виконуватися) багато об'єктів одного й того ж класу. Наприклад, можна на одному комп'ютері запустити на виконання кілька екземплярів найпростішої програми. Кожна з таких програм і буде незалежним об'єктом одного й того ж класу.

Оператор виведення, зазвичай, є першим оператором, що вивчається в різних мовах програмування. Доповнивши найпростіший код, що безпомилково компілюється, оператором виведення текстового повідомлення, отримаємо програму «Hello World!».

Приклад 1. Порівняння текстів найпростіших програм мовою Pascal та C# (табл. 1).

Таблиця 1

Pascal	C#
Program Hello_World; Begin **Write('Hello World! ');** End.	class FirstClass { static void Main() { **System.Console.Write("Hello World!");** } }

Синтаксис написання команд мовою C# потрібно пояснити на основі приналежності одних класів іншим. «Команда Write належить класу Console, який, у свою чергу, належить до простору імен — System». Уже на прикладі перших програм можна познайомити учнів із тим, як підключати додаткові бібліотеки класів оператором using, щоб уникнути постійного повторення імен класів у написанні команд і пришвидшити виконання програми. Для цього перед початком програми для кожного класу, бібліотеку якого ми хочемо підключити, після службового слова using пишуть ім'я класу. Наприклад using System;.

У мовах Pascal, C# та ще багатьох інших мовах програмування за допомогою найпростіших програм можна пояснити учням роль роздільників в алфавіті мови програмування, розтлумачити значення операторних дужок, продемонструвати можливість розділяти текст коду будь-якою кількістю проміжків, нових рядків, можливість додавати в певних місцях програми будь-яку кількість крапок з комою. Під час написання програм будь-якою мовою для зручності перегляду її код розбивають на кілька рядків так, щоб у кожному містилася окрема, відносно незалежна частина програми, іноді доречно сказати — команда або ж оператор. Крім поділу на рядки для команд створюють відступи від початку рядка. Відступи роблять різні — за ієрархічним принципом.

Програми мовою програмування Pascal зберігають з розширенням *.pas, а програми мовою C# з розширенням *.cs. Наприклад, ім'я файла, створеного мовою C#, може бути таким: «Hello_world.cs».

У деяких середовищах програмування мовою Pascal, наприклад Turbo Pascal 7.0, консольне вікно, де виводяться результати виконання програми, автоматично закривається після завершення її виконання. Отже, щоб переглянути результат виконання, потрібно виконати певні додаткові дії. Щось подібне зустрічається і в окремих середовищах розробки мовою програмування C#. Початківцям навіть нескладна маніпуляція створює певні труднощі. Уникнути передчасного закриття вікна з результатами роботи програми можна, додавши в кінці програми рядок, що містить оператор введення. Оператор введення зупинить програму до натискання клавіші «Enter» і на екрані можна буде бачити останні повідомлення середовища виконання. Застосовувати оператор введення лише для зупинки виконання програми можна без будь-яких параметрів.

Приклад 2. Порівняння текстів найпростіших програм мовою Pascal і C#, доповнених оператором введення для можливості перегляду результатів (табл. 2).

Таблиця 2

Pascal	C#
Program Hello_World; Begin Writeln('Hello World! '); **Read;** End.	using System; class FirstClass { static void Main() { Console.Write("Hello World!"); **Console.Read();** } }

Коментарі в програмі

Навіть програми для перших комп'ютерів уже містили досить багато команд. Щоб розібратися, чи просто знайти помилку в громіздкій програмі, доводиться докладати значних зусиль. Саме тому практично в усіх мовах програмування передбачено можливість створення так званих «коментарів». Коментар — це частина тексту програми, що ігнорується під час виконання і містить певні пояснення і підказки. Коментар завжди відокремлюється від основного тексту програми спеціальними вказівками. У мові Pascal коментарі розміщують у фігурних дужках {«коментар»}. Мова C# дозволяє створювати кілька видів коментарів. Знак «//» перетворює в коментар увесь текст після «//» і до кінця цього рядка. Якщо коментар потрібно реалізувати в кількох рядках, то можна користуватися вказівками «/*» і «*/». Усе, що знаходиться між цими знаками, незалежно від кількості рядків, під час виконання програми буде ігноруватися.

Доцільно додавати коментарі навіть до найпростішої програми.

Приклад 3. Порівняння текстів найпростіших програм мовою Pascal і C#, доповнених коментарями (табл. 3).

Коментування можна використовувати для того, щоб тимчасово зробити певну частину програми невиконуваною. Частину тексту програми, яку позначають як коментар, для того щоб вона тимчасово не виконувалась, називають «закоментованою». Процес перетворення частини програмного коду в коментар називають «закоментовуванням». Закоментовування використовують для налагодження програми: пошуку помилок, внесення тимчасових змін. Наприклад, у програмі, що не виконується через синтаксичні помилки, легко впевнитися, де вони знаходяться, закоментувавши вірогідно неправильні команди.

Таблиця 3

Pascal	C#
Program Hello_World; **{Hello_World – ім'я програми.** **Це ім'я задає програміст на свій розсуд. Існують певні правила, що визначають, як потрібно задавати імена змінних, процедур та функцій}** Begin Writeln('Hello World! '); ReadLn; End.	class FirstClass // **FirstClass – ім'я класу.** **// Це ім'я задає програміст на свій розсуд.** **/*Існують певні правила та домовленості, що визначають, як потрібно надавати імена класам та методам */** { static void Main() { System.Console.WriteLine("Hello World!"); System.Console.ReadLine(); } }

Виведення на екран даних різних типів

У найпростішій програмі «Hello World!» використовується оператор виведення Write, який виводить на екран дані **рядкового типу string**. Тип даних string практично однаково називається і записується в обох мовах програмування. От лише в Pascal для позначення рядка використовують одинарні лапки (апострофи), а для позначення рядка в C# звичайні подвійні. Під час нескладного експерименту можна перевірити максимально допустимий розмір рядкової величини. Для цього варто збільшувати в повідомленні оператора виведення кількість символів. Якщо вона перевищує максимально допустиму, то під час компіляції з'явиться повідомлення про помилку. Для мови програмування Pascal максимальний розмір даних типу string складає 256 символів. Для мови C# такий експеримент може бути дещо абсурдним, адже розмір даних типу string у цій мові не обмежується. Проте на практиці рядкові дані в программах, написаних мовою C#, можуть обмежуватися можливостями середовища розробки, точніше редактора коду, у якому така розробка здійснюється.

Якщо розмістити одну за одною дві команди виведення, то під час виконання програми повідомлення другої команди з'явиться в тому ж рядку, що й повідомлення першої. Щоб забезпечити після виведення на екран інформації переведення курсору консолі на новий рядок в обох мовах програмування існує додаткова версія оператора виведення. Вона записується WriteLn() — у Pascal і WriteLine() — у C#. Ln — скорочення від англійського словосполучення «line new» — новий рядок.

З допомогою оператора виведення також можна проілюструвати особливості основних типів даних мов Pascal чи C#. Для цього достатньо внести незначні зміни до найпростішої програми типу «Hello World!». Наприклад, не важко замінити в програмі текст, що виводиться на екран, на якесь невелике ціле число. На перший погляд, виведення на екран цифр відбувається так само, як і текстових даних. Проте оператор виведення багатьох мов програмування може виконувати над даними практично всі допустимі для цього типу даних операції. Над даними цілочисельного типу мов Pascal і C# можна виконувати додавання, віднімання, множення, ділення націло та визначення остачі від ділення. Тобто, якщо в операторі виведення вказати, наприклад, 2+2 — на екрані з'явиться результат його обчислення — 4. Відмінність між числовими і рядковим типами даних можна продемонструвати, подавши цей же вираз як текст, тобто в лапках. Тепер вираз обчислюватися не буде і на екрані з'явиться запис, аналогічний тому, що записаний як аргумент вказівки виведення «2+2».

Відмінність у записі операцій, допустимих над цілочисельними даними мов Pascal і C#, стосується лише ділення націло й визначення остачі від ділення. У Pascal це команди div і mod. Команда Writeln(5 div 2) на екран виведе число 2, а команда Writeln(5 mod 2) на екран виведе 1. Для мови C# ділення націло і визначення остачі від ділення позначаються відповідно «/» і «%». Ділення націло числа 5 на число 2 й остача від такого ділення мовою C# можна подати відповідно командами System.Console.WriteLine(5/2) і System.Console.WriteLine(5%2). Варто зазначити, що остача від ділення ще називають «діленням по модулю».

Подавши як аргумент оператора виведення числа у вигляді десяткових дробів, можна зумовити виконання дій над ними, як над дійсними числами. Результат таких обчислень мовою Pascal буде виводитися у вигляді чисел із плаваючою комою. Команда Writeln(5.0*2.0) виведе на екран число 10 у такому вигляді: 1.0000000000E+01. Мова C# автоматично не перетворює значення результатів обчислення дійсних чисел у форму з плаваючою комою. Тому вказівка System.Console.WriteLine(5.0*2.0) видасть на екран 10.0.

Пояснивши існування цілочисельних і дійсних числових типів, варто звернути увагу на відмінність у виконанні операції ділення над цими даними. Для дійсних числових даних в обох мовах програмування ділення позначається однаково — «/». Якщо потрібно отримати точний результат ділення цілих чисел у вигляді дійсного числа, тобто не як ділення чисел націло і не як остача від ділення, то виконати таку операцію можна, лише попередньо перетворивши типи даних аргументів з цілочисельного в дійсний. В обох мовах програмування можна застосовувати як явне, так і неявне перетворення типів.

Основні типи даних мов програмування Pascal і C#

Використання даних жодної мови програмування не обмежується їх розміщенням як параметра команди виведення повідомлень на екран. Для зберігання даних потрібно відводити в оперативній пам'яті комп'ютера поіменовану область відповідного обсягу. Такий процес чи операцію, називають оголошенням змінних. Щоб оголосити змінну, потрібно вказати її ім'я (ідентифікатор) і тип даних. У більшості мов програмування існує перелік стандартних типів даних, що дозволяє раціонально використовувати пам'ять для виконання операцій. Типи даних мають певні позначення, найменування. Дані відповідного типу не можуть виходити за межі певної області допустимих для даного типу значень. Дуже часто в різних мовах програмування типи даних з однією і тією ж назвою можуть набувати інших

діапазонів можливих значень. Мова програмування C# і класична версія мови Pascal також мають різні набори типів даних. Так, одна з найпоширеніших специфікацій мови, що використовується середовищем програмування Turbo Pascal, має одинадцять простих типів даних (табл. 4).

До цілочисельних типів даних належать: byte, word, shortint, integer, longint. До дійсних — real, single, double, extended. Логічний тип даних — boolean, символьний — char. Мова C# має тринадцять простих типів даних (табл. 5).

Таблиця 4

Типи даних мови програмування Паскаль

Тип	Область значень	Розмір
Byte	від 0 до 255	Беззнакове 8-біт ціле
Word	від 0 до 65535	Беззнакове 16-біт ціле
shortint	від -128 до 127	Знакове 8-біт ціле
Integer	від -32768 до 32767	Знакове 16-біт ціле
Longint	від -2147483648 до 2147483647	Знакове 32-біт ціле
Real	від $\pm 2{,}9 \times 10^{-39}$ до $\pm 1{,}7 \times 10^{38}$	6 байт, точність — розрядів
Single	від $\pm 1{,}5 * 10^{-45}$ до $\pm 3{,}4 * 10^{38}$	4 байти, точність — 7 розрядів
Double	від $\pm 5 * 10^{-324}$ до $\pm 1{,}7 * 10^{308}$	8 байт, точність —16 розрядів
extended	$3{,}4 \times 10^{-4932} - 1{,}1 \times 10^{4932}$	10 байт, точність — розрядів
boolean	true або false	1 байт
Char	усі символи коду ASCII	1 байт

Таблиця 5

Типи даних мови програмування C#

Тип	Область значень	Розмір
Sbyte	від -128 до 127	Знакове 8-біт ціле
Byte	від 0 до 255	Беззнакове 8-біт ціле
Short	від -32768 до 32767	Знакове 16-біт ціле
Ushort	від 0 до 65535	Беззнакове 16-біт ціле
Int	від -2147483648 до 2147483647	Знакове 32-біт ціле
Uint	від 0 до 4294967295	Беззнакове 32-біт ціле
Long	від –9223372036854775808 до 9223372036854775807	Знакове 32-біт ціле
Ulong	від 0 до 18446744073709551615	Беззнакове 32-біт ціле
Float	від ±1,5*10-45 до ±3,4*1033	4 байти, точність – 7 розрядів
Double	від ±5*10-324 до ±1,7*10306	8 байт, точність –16 розрядів
Char	від U+0000 до U+ffff	16-бітовий символ Unicode
Bool	true або false	1 байт

До цілочисельних типів даних мови C# належать: sbyte, byte, short, ushort, int, uint, long, ulong. До дійсних — float і double. Логічний тип — bool. Символьний — char.

Невідповідність типів даних різних мов програмування створює суттєву проблему трансляції програм з однієї мови на іншу. У переважній більшості така проблема вирішується використанням програмної платформи Microsoft .Net Framework, що підтримує багато мов програмування. Платформа .Net передбачає єдину систему типів даних для різних мов програмування. Мова програмування C# не лише базується на платформі .Net Framework, а є в рамках цієї платформи основною мовою програмування. Існують діалекти мови програмування Pascal, що також базуються на програмній платформі Microsoft .Net, наприклад ABC Pascal.Net [5].

Оголошення даних, присвоєння

Програма, написана мовою Pascal, зазвичай складається з двох частин: описової і виконуваної. В описовій частині програми здійснюється оголошення змінних і констант, підключення модулів, опис процедур і функцій. Для оголошення змінних у цій частині програми відводиться відповідний розділ, що позначається службовим словом «Var». Після цього слова розміщують перелік змінних і вказують їхній тип.

Приклад 4. Заголовок й описова частина програми, де оголошено цілочисельну величину a, дійсні величини b та c, рядкову величину d.

```
Program My_program;
Var a:integer; b,c:real;
d:string;
...
```

У мові програмування C# теж є щось подібне до описової частини Pascal-програми, де командою using можна підключати бібліотеки класів. Але специфікація мови C# [6] не передбачає відведення окремого розділу для оголошення даних. Хоча нові змінні можна оголошувати в межах певного класу, існують спеціальні вказівки, що визначають «зону видимості» змінної. Тобто можна вказати, чи буде змінна доступною лише в межах даного класу, чи можна використовувати оголошену змінну в інших класах програми. Фактично оголошувати змінні в мові C# можна майже в будь-якому місці програми.

Надання змінній певного значення називають присвоєнням. Присвоєння в обох мовах програмування має подібну структуру: <ім'я змінної> <знак присвоєння> <присвоюване значення змінної>. От лише в мові програмування Pascal знак присвоєння має вигляд «:=» а в мові C# «=».

Приклад 5. Порівняння текстів програм мовами Pascal і C#, що використовують вказівку присвоєння (табл. 6).

Таблиця 6

Pascal	C#
Program My_program; Var a:integer; Begin a:= 10; Writeln(a); ReadLn; End.	class MyClass { static void Main() { int a; a = 10; System.Console.WriteLine(a); System.Console.ReadLine(); } }

Поряд із цим у мові програмування C# присвоєння можна записувати відразу після оголошення змінної. Тобто запис команд int a; і a=10; можна спростити, об'єднавши в один — int a=10;

Оператори введення

Дуже часто користувач сам має вводити дані до програми під час її виконання. Для цього в мовах програмування використовуються **оператори введення.** Як правило, оператор уведення зупиняє програму, надає можливість користувачеві ввести дані з клавіатури і, після натискання клавіші **Enter**, переносить введені значення у відповідні області пам'яті і продовжує виконання програми.

Спільним у реалізації оператора введення обох мов є власне назва самого оператора — «Read». Проте, якщо в мові програмування Pascal цей оператор можна використовувати для введення даних будь-яких типів, то в мові C# передбачено введення лише символьних значень. Отримати з введених користувачем символів числові чи інші дані можна лише, використавши явне або неявне перетворення типів. Це, напевно, ще одна, після необхідності навіть в найпростішій програмі створювати класи, перешкода для оволодіння початківцями мовою C#. Проте така перешкода приносить певну користь для навчання програмування. Уже на початкових етапах учень має нагоду усвідомити особливості організації роботи комп'ютера. Адже комп'ютер завжди отримує з клавіатури саме символьні дані, які іноді потрібно перетворювати в числові, логічні чи якісь інші типи.

Оператори введення в мовах програмування Pascal і C#, аналогічно до оператора виведення, теж зустрічаються у двох модифікаціях. Оператори введення мови Pascal — це Read і ReadLn. Оператори введення мови C# — Read і ReadLine. У мові Pascal оператор Read, на відміну від оператора ReadLn, не переводить курсор консолі на новий рядок після завершення введення даних. Поряд із цим обидва різновиди оператора введення мови C# після введення даних обов'язково переводять курсор на новий рядок. Тобто мова C# немає оператора введення, який би залишав курсор консолі в тому ж рядку, що й уведені дані.

Відмінність між операторами Read і ReadLine мови C# полягає в тому, що, якщо перший дозволяє здійснювати введення лише даних символьного типу, то другий можна використовувати для введення рядкової величини. Тобто, якщо оператор Read дозволяє ввести з клавіатури лише один символ, то оператор ReadLine забезпечує введення практично необмежено довгого рядка символів.

Варто зазначити, що оператор введення в мові C# потрібно використовувати в поєднанні з оператором присвоєння. Якщо ми хочемо ввести з клавіатури дані в змінну «a», то потрібно писати a=System.Console.Read(). Записана за аналогією з мовою Pascal команда System.Console.Read(a) працювати не буде.

Приклад 6. Порівняння текстів програм мовами Pascal і C#, що використовують введення даних (табл. 7).

Таблиця 7

Pascal	C#
Program My_program; Var a:integer; Begin ReadLn(a); WriteLn(a); Read; End.	class MyClass { static void Main() { string a; a = System.Console.ReadLine(); System.Console.WriteLine(a); System.Console.Read(); } }

Хоча використання оператора введення в мові C# має суттєві обмеження порівняно з оператором введення мови Pascal, забезпечити введення з клавіатури даних типу, відмінного від рядкового (символьного), відносно не складно. Для цього доцільно використовувати перетворення типів командою Convert. Наприклад, щоб ввести до цілочисельної змінної дані з клавіатури, можна використати оператор введення як аргумент оператора перетворення типу даних:

```
int a=System.Convert.ToInt32(System.Console.ReadLine());
```

Програма обчислення суми двох цілих чисел, уведених з клавіатури, буде мати такий вигляд:

```
class MyClass
{
static void Main()
{
int a=System.Convert.ToInt32(System.Console.ReadLine());
int b=System.Convert.ToInt32(System.Console.ReadLine());
System.Console.WriteLine(a+b);
System.Console.Read();
}
}
```

Як уже зазначалось, для компіляції програм, написаних мовою Pascal, під час введення даних з консолі до нерядкових змінних теж відбувається конвертування. Але на відміну від мови C# у мові Pascal таке перетворення відбувається не явно. Це хоч і спрощує написання учнями простих програм, але й позбавляє їх можливості краще зрозуміти принципи організації роботи оператора введення.

Якщо командою using підключити перед текстом програми бібліотеку класу System, то текст програми стане помітно коротшим:

```
using System;
class MyClass
{
static void Main()
{
int a = Convert.ToInt32(Console.ReadLine());
int b = Convert.ToInt32(Console.ReadLine());
Console.WriteLine(a+b);
Console.Read();
}
}
```

(Далі буде)