ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І ЗАСОБІВ НАВЧАННЯ НАПН УКРАЇНИ

П. Г. Шевчук

НАВЧАННЯ ПРОГРАМУВАННЯ В КЛАСАХ ТЕХНОЛОГІЧНОГО ПРОФІЛЮ ЗАГАЛЬНООСВІТНІХ НАВЧАЛЬНИХ ЗАКЛАДІВ НА ОСНОВІ ВИКОРИСТАННЯ МОВИ C#

Методичні рекомендації для вчителів інформатики

Київ
2012

УДК 371.3:372.8:004:681.3.062
ББК 73м:32.97
ШЗ7
Затверджено на засіданні вченої ради Інституту інформаційних технологій і засобів навчання НАПН України, протокол № 10 від 29.11.2012 р.

Рецензенти:
Спірін О. М. – доктор педагогічних наук, доцент, головний науковий співробітник відділу комп'ютерно орієнтованих систем навчання та досліджень Інституту інформаційних технологій і засобів навчання НАПН України;
Яцишин А. В. – кандидат педагогічних наук, старший науковий співробітник відділу комп'ютерно орієнтованих систем навчання та досліджень Інституту інформаційних технологій і засобів навчання НАПН.
Матвійчук С. В. – вчитель інформатики вищої категорії, вчитель-методист Ружинської гімназії Житомирської області.

Шевчук П. Г.

ШЗ7 Навчання програмування в класах технологічного профілю загальноосвітніх навчальних закладів на основі використання мови C# : Методичні рекомендації для вчителів інформатики. – К., 2012. – 32 с.

Завдяки стрімкому розвитку інформаційних технологій актуальним є навчання новітніх засобів розробки комп'ютерний програм. Розглянуто проблеми ознайомлення учнів з основними особливостями об'єктно-орієнтованої парадигми програмування. Наведено приклади доступного пояснення принципів взаємодії об'єктів у комп'ютерній програмі. Описано деякі ресурси Інтернету для підтримки навчання програмування на основі C#. Детально розглянуто особливості навчання мови C# в консолі.

Дані методичні рекомендації адресовані вчителям інформатики загальноосвітніх навчальних закладів, що здійснюють навчання програмування на основі використання мови C#.

УДК 371.3:372.8:004:681.3.062
ББК 73м:32.97

Зміст

Вступ

Інформаційно-комунікаційні технології (ІКТ) лежать в основі становлення сучасного інформаційного суспільства, а галузь розробки комп'ютерних програм є важливою складовою науково-технологічного прогресу. Проте, незважаючи на бурхливий розвиток ІКТ, навчання програмування у загальноосвітніх закладах все ще здійснюється практично на тих же засадах, що й у перші роки впровадження цієї дисципліни до шкільних навчальних програм. Програмування – новітня, актуальна і на даний час досить різноманітна галузь людської діяльності, виробництва, сучасної культури. В залежності від обсягів, змісту навчального матеріалу, використовуваних форм, методів та засобів навчання, програмування може забезпечити досить широку профільну диференціацію. Навчання програмування може стосуватися як необхідних гуманітарію алгоритмів роботи з текстами, так і потрібного у інженерії моделювання технологічних процесів. Проте організація профільного навчання інформатики все ще не реалізовує весь потенціал цієї дисципліни. Новітні підходи до створення комп'ютерного програмного забезпечення, в якійсь мірі, залишилися поза увагою педагогів. Навчальний матеріал з курсу шкільної інформатики недостатньо торкається сучасних парадигм написання комп'ютерних програм. Зокрема недостатня увага приділяється найбільш поширеному та дуже актуальному об'єктно-орієнтованому програмуванню (скорочено ООП).

Загальна комп'ютеризація та впровадження Інтернету торкнулося усіх сфер економіки та управління. Важливість ефективного навчання програмування в класах технологічного профілю продиктована визначальним значенням цієї галузі у сфері найрізноманітніших виробничих технологій.

Використання багатьох сучасних мов та середовищ розробки комп'ютерних програм вимагає глибокого розуміння об'єктної моделі комп'ютерних програм. В той же час для повноцінного навчання об'єктно-орієнтованого програмування

доцільно використовувати сучасні мови та середовища розробки. Серед цілої низки таких мов найбільш корисно впроваджувати до навчання саме ті, що спеціально розроблялися для якомога повнішої реалізації об'єктно-орієнтованого підходу до написання програмного забезпечення. Однією з таких є мова програмування C# (читається Сі шарп).

Мова програмування C# має так званий «сі-подібний» синтаксис, багато її команд схожі на команди мов C та Pascal. Проте, щоб програмувати мовою C#, замало вивчити набори команд та синтаксис. Якщо ви звикли навчати дітей програмувати мовою Pascal чи будь-якою іншою, використовуючи традиційний структурний підхід, то, щоб реалізувати всі переваги об'єктного підходу до написання програм мовою C#, необхідно мати чіткі уявлення про те, що ж таке ООП та розуміти основні його принципи. Напевно, маючи значний багаж традиційних програмістських знань, умінь та навичок, звикнути до ООП досить не просто. Отже, перш ніж програмувати на C#, розглянемо основні особливості та принципи ООП, скориставшись не лише науковими визначеннями, а й аналогіями з конкретними життєвими прикладами.

1. Особливості об'єктно-орієнтованої парадигми програмування

Традиційно ми звикли, що в основі програмування лежить поняття алгоритму. Втім, так само як комп'ютер «мертвий» без програм, що ним виконуються, алгоритм не має жодної користі без даних, які він обробляє. Сукупність алгоритмів та даних і являє собою те, що у програмуванні називають об'єктом. ООП (об'єктно-орієнтоване програмування) – це технологія розробки програмного забезпечення. Структура даних у програмі, розробленій за принципами ООП, дуже нагадує те, як людина сприймає відомості про оточуючий світ. Адже людина із нерозривно пов'язаної в одне ціле природи виокремлює у своїй

уяві певні об’єкти, поіменовує їх, досліджує та класифікує. Навіть можна сказати, що об’єктно-орієнтовані програми нагадують за своєю будовою організацію світу, який нас оточує. Є окрема наукова теорія ООП, але початківцям найкраще пояснювати загальні поняття на конкретних життєвих прикладах.

1.1 Приклади, що дозволяють пояснити на аналогіях взаємодії об’єктів у комп’ютерній програмі

Існує аналогія між особливостями взаємодії об’єктів у комп’ютерній програмі та порядком виконання доручень, що люди дають один одному в повсякденному житті. Розглянемо конкретний приклад.

1.1.1 Передача завдання

Одного разу у мене виникло бажання подарувати своїй матусі пральну машину-автомат. Нині це не складно. Я дуже швидко вибрав у магазині найкращу, на мій погляд, модель. Але це ще не все. Подарунок потрібно привезти у квартиру моєї мами, змонтувати у належному місці та запустити в дію. Але ж я такого ніколи не робив! За аналогією з традиційним підходом до написання комп’ютерних програм мені доведеться не лише навчитися слюсарній та автомобільній справі, а й придбати необхідне обладнання, включаючи вантажний транспортний засіб.

Об’єктний підхід дозволяє передоручити певне завдання іншому об’єкту. Отже, я телефоную знайомому слюсарю, аби він відвіз і встановив пральну машину у квартирі моєї мами. Я можу бути впевнений, що машина буде належним чином відтранспортована і введена в дію. Механізм, що я використав, – це пошук відповідного агента (слюсаря) і передача йому завдання. Обов’язком агента є необхідність виконати моє завдання. Слюсар реалізує певний алгоритм – метод, щоб виконати моє завдання. Мені необов’язково знати, якими методами він виконає завдання. Методи агента навіть можуть

бути мені невідомі. Агент вільний у виборі методу вирішення завдання.

Можливо, якби я поцікавився методами агента-слюсаря, то виявив би, що слюсар не буде виконувати таке просте завдання сам, а доручить його помічнику. Передача завдання помічнику буде обумовлена ще й тим, що у помічника є власний автомобіль. Помічник залучить ще когось, щоб той допоміг йому завантажити покупку. Моє завдання все рівно буде виконане через послідовність завдань, які агенти надішлють один одному.

1.1.2 Деякі принципи ООП

Першим принципом об’єктно-орієнтованого програмування є спосіб **задавання завдання**. Хоча це і звучить дещо тавтологічно, але завдання мало сформулювати, його важливо комусь задати.

Завдання в об'єктно-орієнтованому програмуванні ініціюється за допомогою передачі повідомлень агенту (об'єкту), що є відповідальним за завдання. Повідомлення містить вказівку на виконання завдання і супроводжується додатковими даними (аргументами), необхідними для його виконання. Одержувач – це агент, якому надсилається повідомлення. Якщо він приймає повідомлення, то на нього автоматично покладається відповідальність за виконання зазначеної дії. Як реакція на повідомлення, одержувач запустить деякий метод, щоб задовольнити отриману вказівку.

Незалежність агента забезпечує принцип маскування даних (ще цей принцип називають **інкапсуляцією**) щодо пересилання повідомлень, згідно з якими клієнтові не потрібно знати про засоби, за допомогою яких його запит буде виконано.

Спочатку клієнт, якщо в нього є завдання, повинен знайти, кому можна було б його доручити. Це також певний принцип. Якщо програміст довгий час розробляє код традиційними методами, то цілком нормальне вміння «перекласти свої завдання на чужі плечі» в нього майже повністю атрофувалося. Практично всяка мова програмування дозволяє йому самому

розробити кожен окремий елемент програми від самого початку, не звертаючись до послуг ще когось.

У традиційних мовах програмування приховування даних також є важливим принципом. У підпрограмах (процедурах чи функціях) внутрішня структура відділена від зовнішньої певними рівнями доступу. Але є відмінності між викликом процедури і передачею «доручення». Перша полягає в тому, що в «доручення» є цілком конкретний одержувач – агент, якому надіслано повідомлення. Цей агент повністю самостійний. При виклику процедури немає настільки явно виділеного одержувача. Точніше процедура не володіє повною незалежністю, а виконується наче під наглядом основної програми. Ще одна відмінність полягає в тому, що інтерпретація повідомлення (тобто метод, що викликається після прийому повідомлення) залежить від одержувача і є різною для різних одержувачів. Це так званий **поліморфізм**. Я можу передати моє повідомлення, наприклад, моїй дружині, і вона його зрозуміє, і як результат завдання буде виконано (а саме пральна машина буде встановлена у квартирі моєї матусі). Однак метод, який використовує моя дружина для виконання запиту, буде іншим, ніж той, який застосує слюсар у відповідь на той же самий запит. Я більш ніж впевнений, що моя дружина просто переадресує завдання встановити пральну машину своїй подрузі – дружині слюсаря. Можливо, в такому випадку завдання буде виконано найкраще, але такі нюанси вже далеко виходять за межі ООП.

З наведених прикладів не слід робити висновок, що одне і теж завдання можна доручати кому завгодно. Якщо я попрошу потурбуватися про встановлення пральної машини мого сина, в нього може не виявитися відповідного методу для розв'язання поставленого завдання. Але якщо мій син не сприйме цей запит, то він видасть певне діагностичне повідомлення про помилку.

1.1.3 Відповідальність та пізнє зв'язування

У традиційному (процедурному) програмуванні використовують так зване раннє зв'язування. Раннє зв'язування

між повідомленням (ім'ям процедури або функції) і фрагментом коду (методом) відбувається при традиційному виклику процедури. Лише дана підпрограма і лише вказаним методом має виконати завдання. В ООП має місце пізнє зв’язування. Інтерпретація (вибір відповідного методу, який запускається у відповідь на повідомлення) може бути різною для різних одержувачів. Хоча у повідомлення існує певний одержувач, цей конкретний одержувач фактично лишається невідомим аж до виконання програми. Отже, визначити, який метод буде викликаний, заздалегідь неможливо.

Важливою особливістю ООП є **відповідальність** за виконання завдання. Слюсар вільний у виборі способів виконання завдання, втім саме завдання для нього обов’язкове. Це і задає ,більш вищий рівень абстракції, незалежність агентів – критичний фактор у складних програмах.

Об'єкти не можуть постійно реагувати на завдання тільки таким способом, що звертаються до інших з вказівкою виконати деяку дію. Це призведе до нескінченного циклу запитів. Пригадуєте, як Собакевич з Чичиковим пропонували один одному пройти в двері першим? Так-от, у випадку складної комп’ютерної програми ланцюжок (кількість «Собакевичів-Чичикових», що надсилають одному повідомлення пройти першим) може виявитись значно довшим, але суть залишиться та ж сама: програма не зможе виконуватись. Якщо не всі, то хоча б деякі об'єкти повинні виконувати певну роботу, перш ніж звертатися з вказівкою до інших.

1.1.4 Наслідування

Ще варто розглянути таке поняття, як наслідування. Сфера моїх життєвих потреб не обмежується вирішенням однієї проблеми – як подбати про побут рідної мами. Я реалізовую багато методів у різних сферах. У мене підростає син, і він у своєму житті, можливо, скористається тими ж методами, що й я. Але я не можу бути впевненим у цьому. Майже повсякчас діти не зважають на досвід своїх батьків. У ООП все багато простіше: методи, що наслідують об’єкти від своїх «батьків»

завжди чітко визначені. Як і в реальному житті наслідування допомагає зберегти зроблене «старшими поколіннями», тобто багаторазово використовувати один і той же код.

2. Мова програмування C#

Щоб оволодіти об'єктно-орієнтованим програмуванням на практиці, корисно знати та вміти використовувати конкретну мову програмування. Серед когорти мов програмування чільне місце займає мова, що унаслідувала практично усе найкраще від своїх попередників, молода і красива мова програмування C#. Ця мова набуває все більшої популярності. Її освоюють не лише початківці, а й ті програмісти, що мають навики використання інших мов програмування. Ця мова програмування повністю підтримує три основних принципи, що лежать в основі об'єктно-орієнтованого програмування: інкапсуляцію, поліморфізм та успадкування – і широко використовується для навчання об'єктно-орієнтованого програмування.

Поряд із цим, мова програмування C# має цілий ряд інших переваг, що роблять її зручною у використанні для навчання програмування:

– мова програмування C# – мультипарадигмальна (така, що підтримує декілька парадигм). Окрім об'єктно-орієнтованої вона підтримує імперативну та функціональну парадигми;

– мова C# надає можливість створювати програми як для консольного, так і для віконного виконання;

– «C-подібний» синтаксис мови C#, як найбільш поширений, дозволяє фахівцю відносно легко перейти до використання багатьох інших мов програмування;

– існує багато доступних для використання візуальних середовищ програмування мовою C#, (Microsoft Visual Studio Professional, Microsoft Visual C# Express Edition, Sharp Develop, Mono Develop, Borland C# Editor).

– програми, написані мовою C#, можуть виконуватися під управлінням багатьох операційних систем, що підтримують програмні платформи Microsoft .Net Framework чи Mono;

–фірма Microsoft (розробник мови програмування C# та середовища програмування Visual Studio Professional), а також інші розробники середовищ програмування мовою C# надають потужну і різносторонню підтримку своїх впроваджень.

2.1 Ресурси Інтернету для підтримки навчання мовою програмування C#

Мова програмування C# все ще мало використовується у навчальних цілях. Існують значні труднощі не тільки щодо навчально-методичної, а й загальнотехнічної підтримки впровадження. Для вирішенням таких проблем зручно скористатися багаточисельними ресурсами глобальної мережі Інтернет. Існує ціла низка навчальних, комерційних, благодійних, аматорських та інших колективів, організацій, що надають потужну підтримку навчання програмування мовою C# в мережі Internet.

Найбільш повно мова програмування C# представлена в мережі Інтернет її розробником, фірмою Microsoft. Ця фірма фактично одноосібно встановлює та розвиває стандарти мови C#, розробляє та супроводжує власний продукт, середовище розробки Visual Studio. Офіційна сторінка продукту доступна за адресою http://www.microsoft.com/visualstudio/en-us/home. Від початку виходу програмної платформи .Net до складу Microsoft Visual Studio обов'язково входить мова програмування C#. Фірма Microsoft здійснює в Інтернеті потужну та різносторонню підтримку як мови програмування, так і середовища розробки. Головний сайт підтримки проекту "Visual C# Developer Center" доступний за адресою http://msdn.microsoft.com/en-us/vcsharp/. Також насиченим джерелом інформації про мову C# є російськомовний блог "Microsoft для преси".

Знайти останню версію та основні матеріали з написання програм мовою C# з використанням середовища розробки Sharp Develop можна на сайті цього продукту http://www.sharpdevelop.com/OpenSource/SD/ . Сайт розробника Mono Develop знаходиться за адресою http://mono-project.com.

В Internet публікуються приклади представлення мови C# загальноосвітніми навчальними закладами. Два сайти сільських шкіл з Хмельницької області містять практичні матеріали з навчання програмування мовою C#. На сайті Малиницького НВК вчитель В. А. Ребрина розмістив матеріали факультативного курс для 8 класу з програмування на C# (http://malnvk.ucoz.ua/load/1-1-0-1). Також активно розвивається сайт "Творча лабораторія вчителя інформатики Гаврилівської ЗОШ Хмельницької області Пилипчука О. П.", на якому представлені корисні в навчанні матеріали для ознайомлення з мовою програмування C#. Зокрема О. П. Пилипчук демонструє на своєму сайті приклади написання класів C# для виконання обчислень із звичайними дробами, які спеціально розроблені для демонстрації переваг об'єктно-орієнтованого програмування (http://teachlab.ucoz.ua/publ/6-1-0-35).

Ще одним прикладом розміщення матеріалів з навчання програмування мовою C# може слугувати "Автоматизована система навчання "Уроки в Інтернеті" на сайті Миропільської гімназії Романівського району Житомирської області (http://www.ms1.org.ua/workbooks/). Доступ до матеріалів цієї системи мають лише зареєстровані користувачі: учні, вчителі та "гості" гімназії. В якості "гостя" може зареєструватись практично будь-який відвідувач сторінки. В старій версії системи "Уроки в Інтернеті" (http://www.ms1.org.ua/lessonsold/) без обмеження доступу можна проглянути уроки, серед яких є розробки з навчання програмування різними мовами та в різних середовищах. Для інформаційної підтримки навчання програмування мовою C# використовується форум цього сайту.

Окремої уваги заслуговують on-line виконавці (компілятори, інтерпретатори), автоматизовані системи перевірки виконання завдань, зокрема проект «E-olimp» (http://www.e-olimp.com.ua/). Такі системи є ефективними Інтернет-ресурсами для навчання програмування.

3. Методичні особливості навчання C# в консолі

3.1 Найпростіші програми мовою C#

В мові C# та ще багатьох інших мовах програмування за допомогою найпростішої програми можна пояснити учням призначення 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. При написанні 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.

В деяких середовищах програмування мовою C# 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.

3.2 Коментарі у програмі

Навіть програми для перших комп’ютерів вже 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 та підказки. Коментар завжди відокремлюється від основного тексту програми спеціальними вказівками. Мова C# надає можливість 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 «/*» та «*/». Все, що знаходиться між цими знаками, незалежно від кількості рядків, при виконанні програми буде ігноруватися. Доцільно додавати коментарі навіть до найпростішої програми.

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 інших тимчасових змін. Наприклад, у програмі, що не виконується через синтаксичні помилки, легко впевнитися, де вони знаходяться, закоментувавши вірогідно неправильні команди.

3.3 Виведення на екран даних різних типів

З допомогою оператора виведення також можна проілюструвати особливості основних типів даних. Для цього достатньо внести незначні зміни до найпростішої програми типу «Hello World!». Наприклад, не важко замінити у 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. Втім, оператор виведення багатьох мов програмування може виконувати над даними практично всі допустимі для цього типу даних операції. Над даними цілочисельного типу можна виконувати додавання, віднімання, множення, ділення націло та визначення залишку 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». Крім того, над цілочисельними даними можна виконувати ділення націло та визначення залишку від ділення. Для мови C# ділення націло та визначення залишку від ділення позначаються відповідно «/» та «%». Ділення націло числа 5 на число 2 та залишок від такого ділення на мові C# можна подати відповідно командами System.Console.WriteLine(5/2) та System.Console.WriteLine(5%2). Варто зазначити, що залишок 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. 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 та дійсних числових типів, варто звернути увагу на відмінність у виконанні операції ділення над цими даними. Для дійсних числових даних ділення позначається однаково – «/». Якщо потрібно отримати точний результат ділення цілих чисел у вигляді дійсного числа, тобто не як ділення чисел націло і не як залишок від ділення, то виконати таку операцію можна, лише попередньо перетворивши типи даних аргументів з цілочисельного в дійсний. Можна застосовувати як явне, так і неявне перетворення типів і доцільно це розглядати при більш поглибленому вивченні матеріалу.

Дуже часто в різних мовах програмування типи даних з однією і тією ж назвою можуть набувати інших діапазонів можливих значень. Невідповідність типів даних різних мов програмування створює суттєву проблему трансляції програм з

однієї мови на іншу. Значною мірою 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.

3.4 Оголошення даних, присвоєння

Нові змінні в мові програмування C# можна оголошувати в межах певного класу. Існують спеціальні вказівки, що визначають «зону видимості» змінної. 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.

3.5 Оператори введення

Дуже часто користувач сам має вводити дані до програми під час її виконання. Для цього в мовах програмування використовуються оператори введення. Зазвичай оператор введення зупиняє програму, надає можливість користувачу ввести дані з клавіатури і, після натискання клавіші «Enter», переносить введені значення у відповідні області пам’яті та продовжує виконання програми.

Оператором “Read” у мові C# передбачено введення лише символьних значень. Отримати з введених користувачем символів числові чи інші дані можна лише використавши явне або неявне перетворення типів.

Оператори введення, аналогічно до оператора виведення, теж буває в двох модифікаціях. Оператори введення мови C# – Read та ReadLine. Обидва різновиди оператора введення мови C# після введення даних обов’язково переводять курсор на новий рядок. Відмінність між операторами Read та 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 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().

Забезпечити введення з клавіатури даних типу відмінного від рядкового (символьного) відносно нескладно. 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());

3.6 Розгалуження

Лінійний хід виконання програми можна змінити, використавши оператор вибору, який ще називають «розгалуження». Розгалуження складається з трьох елементів:

1) умови;

2) дії (серії команд), що виконується, якщо умова істинна;

3) дії (серії команд), що виконується, якщо умова хибна.

Якщо розгалуження не містить третього елементу, то його називають неповним. Тобто відмінність повної і неповної форми розгалуження полягає в тому, що неповне розгалуження не містить ніяких дій (команд) для виконання в тому разі, коли умова хибна.

В мові програмування C# розгалуження записується: if (<умова>) <дії, що виконується, якщо умова істинна> else <дії, що виконується, якщо умова хибна>;. Неповна форма розгалуження записується без «else»: if (<умова>) <дії, що виконується, якщо умова істинна> ;.

3.7 Цикли

Цикл – це послідовність команд, що повторюються скінченну кількість разів. У більшості мов програмування

використовуються три основних типи циклів: з передумовою, з післяумовою та цикли з параметром. Мова програмування C# теж дозволяє створювати такі типи циклів. Втім вона має особливий тип циклів – «foreach». Цей тип циклів C# не унаслідувала від інших мов програмування [6]. Швидше за все розгляд циклів «foreach» варто здійснювати лише під час поглибленого вивчення програмування.

Цикли всіх типів обов'язково мають «тіло циклу». Тіло циклу – це, власне, і є ті команди, що повторюються. Одне окреме повторення тіла циклу називають ітерацією. Кількість повторень, відповідно, звуть кількістю ітерацій циклу. Цикли з після та передумовою, окрім «тіла». містять ще «умову». Умова – це логічна величина. Якщо в циклах з передумовою вона перевіряється до виконання тіла циклу, то в циклах з післяумовою умова перевіряється після кожного виконання тіла циклу. Тобто цикл з післяумовою передбачає хоча б одне обов'язкове виконання тіла циклу.

В мові програмування C# цикл з передумовою записується: while (<умова>) тіло циклу;.

В мові програмування C# цикл з післяумовою: do <тіло циклу> while(<умова>);.

Демонструючи відмінність у використанні циклів з перед- та післяумовою, доцільно вказати на можливість практично повної взаємозаміни цих алгоритмічних конструкцій. Наприклад, програму знаходження суми введених з клавіатури чисел, де використовувався цикл з передумовою, можна трохи змінити, щоб команда введення не з'являлась в ній двічі.

Цикли з параметром, окрім «тіла», містять вказівки надання додатковій змінній (параметру) початкового значення та зміни його під час ітерацій. За час виконання циклу з параметром додаткова змінна пробігає певний діапазон значень від свого початкового до кінцевого значення.

В мові програмування C# цикл з параметром характеризується, окрім початкового та кінцевого значень параметру, ще й «кроком». Крок вказує, на яку величину

змінюється параметр при кожному виконанні тіла циклу. Поряд із цим у мові C#, на відміну від Pascal, кінцеве значення параметру вказується у вигляді логічної умови. Для того, щоб цикл виконувався, умова повинна бути істинною. Ось так схематично записується цикл з параметром мовою програмування C#: for (<вираз визначення параметра та його початкового значення>, <умова набуття параметром кінцевого значення (припинення циклу)>, <вираз зміни параметра на величину кроку>) тіло циклу;.

3.8 Масиви

Окрім простих типів даних, мова програмування має цілу низку так званих структурованих типів. У шкільному курсі програмування детально вивчається лише структуровані типи даних «масиви». Їх ще звуть табличними величинами та матрицями. Масив – це скінченна пронумерована сукупність даних одного типу. Масив складається зі значень, що називаються елементами масиву. Тип даних елементів називають базовим типом даних масиву. Кожен елемент характеризується індексом. Індекс вказує на порядок розміщення елементу в масиві і набуває значень певного перелічуваного типу даних. Для кожного елементу масиву, крім першого і останнього, обов'язково є наступний та попередній. Індекс попереднього елементу масиву на одиницю менший, а індекс наступного елементу на одиницю більший.

Масиви можна оголошувати подібно до оголошення інших типів даних. В мові C# стосовно масивів діють ті ж самі правила видимості, що й для інших величин.

В мові програмування C# на те, що нова змінна буде масивом, вказують прямокутні дужки після зазначення її базового типу даних. Наприклад: фрагмент коду int[] m – вказує на те, що оголошується цілочисельний масив з іменем m. Специфікація мови C# передбачає два основних способи визначення розміру масиву. Найпростіше вказати розмір масиву, задавши на етапі оголошення усю множину значень його елементів. У такому разі кількість вказаних елементів масиву і

визначить його розмір. Наприклад, запис int[] m = {1,0,2,9,3,8,3,7,4,6,5}; вказує на те, що масив m складається з десяти елементів. Схематично оголошення масиву, в поєднанні із зазначенням усіх його елементів, записується так: <тип даних>[] <ім'я масиву> = {<значення першого елементу>,< значення другого елементу >,…,< значення останнього елементу >};. Якщо на етапі оголошення масиву не потрібно вказувати його значення, тобто доцільно залишити масив невизначеним, запис про оголошення масиву набуде наступного схематичного вигляду <тип даних>[] <ім'я масиву > = new <тип даних>[<кількість елементів>];. Наприклад, запис int[] m = new int[10] оголошує масив цілочисельних даних з іменем m, що має розмір 10 елементів. У наведених прикладах запису масиви мови C# індексуються цілими числами починаючи з нуля.

Розв'язання задач з використанням масивів спирається на деякі основні алгоритми роботи з табличними величинами: введення даних до масиву, виведення значень масиву, визначення суми елементів масиву, знаходження найбільшого (найменшого) елементу масиву, пошуку елементів або їх кількості за певними умовами, сортування елементів масивів.

Можна організувати введення даних до масиву з клавіатури та виведення їх на екран, використавши цикли. Найчастіше для роботи з масивами використовують цикли з параметром. Параметр циклу вказує на індекс елементу масиву, до якого відбувається звернення.

Для визначення суми елементів масиву потрібно оголосити змінну, що цю суму буде накопичувати.

Знаходження найбільшого елементу масиву потребує використання розгалуження:

Велику кількість задач можна запропонувати учням на пошук елементів або їх кількості за певними умовами. Наприклад, розглянемо програму, що визначає кількість від'ємних елементів масиву:

```
using System;
```

```
class My Class
{
  static void Main()
  {
  int[] m = new int[10]; //оголошення масиву з 10-ти значень
  int i; // «i» буде використовуватись як параметр циклу
  for (i = 0; i < 10; i=i+1)
          //  цикл в якому «i» змінюється від о до 9
           m[i] = Convert.ToInt32(Console.ReadLine());
  // тіло циклу – введення значень масиву з клавіатури
  int n = 0; // n – міститиме кількість від'ємних елементів
  for (i = 1; i < 10; i=i+1)
           // цикл організований як попередній
           if (m[i] <0) n=n+1; // тілом циклу є розгалуження
  Console.WriteLine(n); // результат виводиться на екран
  // програма зупиняється до натискання «Enter»
  Console.Read();
  }
}
```

Однією з класичних задач теорії алгоритмів є сортування (впорядкування) елементів масивів за їх значенням. У шкільному курсі найчастіше вивчаються два загальновідомі алгоритми сортування масивів – метод вибору та метод «бульбашки». Кожен із цих алгоритмів передбачає обмін місцями елементів масиву. Для цього зручно використовувати додаткову змінну базового типу.

Програма сортування масиву методом вибору полягає у визначенні найбільшого елементу масиву та обміну його з першим. Алгоритм повторюється щоразу з меншим числом даних. Елемент, що перед цим був визначений як найбільший, не бере участі в наступних обмінах:

```
using System;
class MyClass
{
```

```
        static void Main()
        {
        int[] m = new int[10]; // оголошення масиву
        int i, j, imax, r; // оголошення допоміжних величин
        for (i = 0; i < 10; i=i+1)
                    // цикл в якому і змінюється від о до 9
                    m[i] = Convert.ToInt32(Console.ReadLine());
                    // введення значень масиву
        for (i = 0; i < 9; i=i+1)//цикл в якому і змінюється від о до 8
              {
              imax = i;
              for (j = i+1; j < 10; j++)
                 // цикл в якому і змінюється від і+1 до 9
                    if (m[j] > m[imax])
                    // умова обміну  значень комірок масиву
                    {
                    imax=j;
                    r = m[i]; m[i] = m[imax];
                    m[imax] = r;
                    // обмін комірок масиву
                    }
          for (i = 0; i < 10; i=i+1)
                    Console.WriteLine(m[i]); //виведення результату
          Console.Read();
          // програма зупиняється до натискання «Enter»
        }
}
```

Сортування масиву методом «бульбашки» ще носить назву «обмінного сортування». Під час обмінного сортування відбувається обмін місцями сусідніх елементів масиву. Сортування припиняється тоді, коли не вдається знайти жодних двох сусідніх елементів, що потребують обміну. В такому разі виконується ітерація циклу, в якій не відбувається жодного

обміну елементів масиву, що й слугує сигналом для припинення сортування:

```
using System;
class MyClass
{
        static void Main()
        {
        int[] m = new int[10]; // оголошення масиву
        int i; int r; bool fleg; // оголошення допоміжних величин
        for (i = 0; i < 10; i=i+1)//цикл в якому і змінюється від о до9
         m[i] = Convert.ToInt32(Console.ReadLine());
        // введення масиву
        do // початок циклу з післяумовою
                {
                fleg = false; // покажчик припинення сортування
                for (i = 0; i < 9; i=i+1) // цикл проходження масиву
                if (m[i] > m[i+1])
                // порівнняння значень сусідніх комірок
                    {
                     r = m[i];
                     m[i] = m[i+1];
                     m[i+1] = r;
                     // обмін комірок масиву
                     fleg = true;
                     // вказує, що сортування не закінчено
                     }
                }
        while (fleg); // умова циклу з післяумовою
        for (i = 0; i < 10; i=i+1)
                    Console.WriteLine(m[i]);  // виведення результату
        Console.Read();
                    // програма зупиняється до натискання «Enter»
        }
}
```

3.9 Спрощення навчального матеріалу

Обсяг часу, що відводиться на уроках інформатики для навчання програмування постійно зменшується. Нова навчальна програма з інформатики для учнів 10-11 класів загальноосвітніх навчальних закладів, академічний рівень, передбачає всього-навсього 26 годин навчального часу на вивчення розділу «Основи програмування [7]». Програма з інформатики (рівень стандарту) для знайомства учнів з програмуванням взагалі відводить всього 5 годин теми «Базові поняття програмування [8]». За таких умов вчитель повинен подбати про підвищення доступності навчального матеріалу. Одним із способів підвищення доступності навчального матеріалу є його часткове спрощення та скорочення.

У наведених вище прикладах використання мови C# свідомо уникнуто використання оператора інкременту. Наприклад, у задачах обробки масивів замість безпосереднього інкременту змінної «i» використовується вираз, що реалізовує інкремент через присвоєння та додавання – “i <присвоїти> i+1”.

Таблиця 1. Реалізація інкременту мовою C#.

Операція	C#
Інкремент	i++
Заміна присвоєнням	i = i+1

Інкремент (з англійської increment *збільшення)* – операція збільшення аргументу на деяку фіксовану величину або ж, у деяких випадках, на змінну. Зворотну операцію називають декремент (зменшення). Збільшення певної величини на одиницю надзвичайно широко використовується при написанні комп’ютерних програм. Практично в усіх мовах програмування використання спеціальних функцій інкременту прискорює виконання алгоритму, а отже, є одним із засобів його оптимізації. Поряд із цим, саме позначення операції інкременту повпливало на появу назви мови C#. Одна з попередниць C# – мова програмування C++ отримала свою назву від позначення цієї операції. Значок # – став образним втіленням чотирьох

плюсів. Такий цікавий історичний факт можна повідомити учням, щоб пробудити інтерес до предмету.

Ще один аспект написання програм мовою C# – це використання додаткових модулів, бібліотек процедур, функцій, а для мови C# – бібліотек класів. Історично використання комп'ютера пов'язано з виконанням значних математичних обчислень. Дуже важливо продемонструвати учням обчислювальні можливості комп'ютера вже на етапі написання перших комп'ютерних програм. Проте в мові C# для виконання обчислень з використанням тригонометричних функцій, логарифмів та інших математичних засобів виникає необхідність використання додаткового класу. Платформа .NET Framework містить бібліотеку із звичайними математичними функціями, відому як бібліотека "Math". Розглянемо програму, що за відомими катетами обчислює гіпотенузу прямокутного трикутника.

```
using System;
class Example
{
 static void Main()
 {
 Console.WriteLine("довжину першого катета 'a'");
 double a = Convert.ToDouble(Console.ReadLine());
 Console.WriteLine("довжину другого катета 'b'");
 double b = Convert.ToDouble(Console.ReadLine());
 double c = (Math.Sqrt (a * a + b * b ));
 Console.Write("довжина гіпотенузи складає ");
 Console.Write(c);
 Console.Read();
 }
}
```

3.10 Умови деяких задач з системи E-olimp

Як уже зазначалось, проект «E-olimp» (http://www.e-olimp.com.ua/) доцільно активно використовувати в процесі навчання програмування на основі використання мови C#. Даний ресурс містить велику кількість задач. Їх список постійно поповнюється. Для усіх задач підібрані системи тестів, на основі яких можна оцінювати написані учнями-користувачами програми. Наводимо номери та умови задач з сайту E-olimp, які можна розв'язати на основі викладеного вище матеріалу.

Задача № 1000 «Задача A + B»

Загальна умова задачі: Знайти A + B

	Вхідні дані	Вихідні дані
Технічні умови	У кожному рядку задано два цілих числа A та B (|A|, |B| ≤ 30000). Дані зчитуйте до кінця файлу.	Для кожного наведеного прикладу виведіть суму A + B у окремому рядку.
Приклади даних	1 1	2

Задача № 937 «Добуток цифр трицифрового числа»

Загальна умова задачі: Знайти добуток цифр трицифрового числа.

	Вхідні дані	Вихідні дані
Технічні умови	У єдиному рядку задано трицифрове ціле число.	Шуканий добуток.
Приклади даних	198	72

Задача № 918 «Яка чверть?»

Загальна умова задачі: Задано точку з координатами x та y. Визначити, в якій координатній чверті вона розміщена.

	Вхідні дані	Вихідні дані
Технічні умови	У єдиному рядку через пропуск задано 2 дійсні числа – координати точки, значення координат по модулю не перевищують 100.	Єдине число – номер відповідної чверті, або 0, якщо однозначно визначити чверть неможливо.
Приклади даних	12 31	1

Задача № 1154 «Гурток хорового співу»

Загальна умова задачі: У деякому начальному закладі функціонує гурток хорового співу. Початок роботи гуртка завжди відбувається однаково: за сигналом керівника гуртка всі N учасників стають у круг і кожен М-й для розспівки співає гаму.

Керівник гуртка помітив, що розім’яти горло не завжди вдається всім учасникам гуртка. За заданими N і M допоможіть йому визначити, чи у черговий раз у розминці приймуть участь всі учасники хору.

	Вхідні дані	Вихідні дані
Технічні умови	Вхідні дані складаються з декількох тестових випадків. Кожен тестовий випадок розміщено у окремому рядку і містить два цілих числа N і M. ($1 \le N, M \le 10^3$).	Для кожного тестового випадку у окремому рядку виведіть "YES", якщо у розминці приймуть участь всі учасники хору, у протилежному випадку виведіть "NO".
Приклади даних	4 1 6 3	YES NO

Задача № 76 «Нова шафа»

Загальна умова задачі: Задано розміри прямокутних дверей a, b та розміри шафи, що має форму прямокутного паралелепіпеда x, y, z. Чи можна пронести шафу у двері, якщо проносити її дозволяється так, щоб кожне ребро шафи було паралельне або перпендикулярне кожній стороні дверей.

	Вхідні дані	Вихідні дані
Технічні умови	П'ять дійсних чисел a, b, x, y, z (0 < a, b, x, y, z < 10).	Вивести 1, якщо шафу можна вільно пронести у двері та 0 у протилежному випадку.
Приклади даних	5 7 4 6 8	1

Задача № 916 «Цікавий добуток»

Загальна умова задачі: Визначити всі можливі значення добутку i*j, якщо цілочислові значення змінних i та j змінюються відповідно і від a до b та j від c до d (1 ≤ a, b, c, d ≤ 10).

	Вхідні дані	Вихідні дані
Технічні умови	У єдиному рядку 4 числа через пропуск: a, b, c та d.	Єдине число – кількість можливих варіантів добутку.
Приклади даних	1 10 1 10	42

Задача № 927 «Кількість іграшок»

Загальна умова задачі: Задано кількість видів іграшок в магазині, кількість іграшок кожного виду та вартість іграшки кожного виду. Визначити кількість іграшок, вартість яких менша за 50 грн.

	Вхідні дані	Вихідні дані
Технічні умови	У першому рядку задано кількість наявних у прейскуранті видів іграшок n (0 ≤ n ≤ 1000). У наступних	Вивести кількість іграшок,

	n рядках задано по 2 числа через пропуск: спочатку кількість іграшок a ($0 \le a \le 1000$) чергового виду та їх ціна b ($0 < b \le 10000$) в грн.	вартість яких менша за 50 грн.
Приклади даних	3 2 100.00 5 23.00 10 22.50	15

Задача № 928 «Сума найбільшого та найменшого»

Загальна умова задачі: Задано одновимірний масив цілих чисел. Визначити суму найменшого та найбільшого елементів масиву.

	Вхідні дані	Вихідні дані
Технічні умови	У першому рядку задано кількість елементів масиву h ($h \le 100$). У другому рядку через пропуск задано самі елементи масиву, значення кожного з яких за модулем не перевищує 100.	Вивести суму найменшого та найбільшого елементів масиву.
Приклади даних	4 1 2 3 4	5

Задача № 932 «Висота трикутника».

Загальна умова задачі: Визначити висоту трикутника площею S, якщо його основа більша за висоту на величину a.

	Вхідні дані	Вихідні дані
Технічні умови	Два цілих числа: S ($0 < S \le 100$), та через пропуск a ($\|a\| \le 100$).	Шукана висота з точністю до сотих.
Приклади даних	15 1	5.00

Задача № 926 «Формула Герона»

Загальна умова задачі: Задано сторони a, b, c, d та діагональ f опуклого чотирикутника. Визначити площу чотирикутника, використовуючи допоміжну функцію обчислення площі трикутника за формулою Герона.

	Вхідні дані	Вихідні дані
Технічні умови	У єдиному рядку задано через пропуск 5 чисел: a, b, c, d, f ($0 < a, b, c, d, f \leq 100$), як це показано на рисунку (див. сайт).	Єдине число – площа чотирикутника, обчислена з точністю до 4-х знаків після десяткової крапки.
Приклади даних	3 4 4 2 5	9.7997

Задача № 948 «Площа та об`єм піраміди»

Загальна умова задачі: Сторона основи правильної чотирикутної піраміди d, бічне ребро p. Визначити площу повної поверхні та об'єм піраміди.

	Вхідні дані	Вихідні дані
Технічні умови	У єдиному рядку через пропуск основа та бічне ребро. Вхідні дані не перевищують 100.	Через пропуск шукані площа та об'єм, результат вивести з точністю до тисячних.
Приклади даних	20 15	847.214 666.667

Список використаної літератури:

1. Бадд Т. Объектно-ориентированное программирование в действии / Тимоти Бадд ; [пер. с англ.]. – СПб.: «Питер», 1997. – 464 с.
2. Концепція профільного навчання в старшій школі. // Трудове навчання. – 2010. – № 4(28). – С. 3-7.

3. Шевчук П. Г. Від Pascal до C# [Текст] / П. Г. Шевчук // Комп'ютер у школі та сім'ї : Науково-методичний журнал. – 2011 – № 4 – С. 47–52; № 5 – С. 40–45.
4. Шевчук П. Г. Значення стилю програмування в процесі навчання учнів та студентів / О. М. Кривонос, П. Г. Шевчук // Комп'ютерно-інтегровані технології: освіта, наука, виробництво : Науковий журнал; Відп. ред. В. Д. Рудь. – Луцьк, 2011. – № 5 – С. 148 – 150
5. Шевчук П. Г. Програмно-технологічні умови використання мови C# для навчання програмування в загальноосвітніх навчальних закладах / П. Г. Шевчук // Комп'ютерно-орієнтовані системи навчання: збірник наукових праць. М-во освіти і науки України, НПУ ім. М. П. Драгоманова; Відп. ред. М. І. Жалдак. – Київ, 2011. – Вип. 17 – С. 80 – 83.
6. C# – в школу. [Електронний ресурс] // Веб-сайт «Microsoft для прессы». – Режим доступу: http://www.ms4press.ru/post/2009/11/03/C-d0b2-d188d0bad0bed0bbd183.aspx – Назва з екрана.
7. C# у школі [Електронний ресурс] // Веб-сайт «Для тих хто хоче дізнатися яке ж місце за шкільною партою дістанеться новій мові програмування C#.» – Режим доступу: https://sites.google.com/site/c4plus – Назва з екрана.
8. C#. Language Specification. Version 3.0 / Copyright. Microsoft Corporation 1999-2007. All Rights Reserved. [Електронний ресурс] // Центр загрузки Microsoft. – режим доступу: http://download.microsoft.com/download/3/8/8/388e7205-bc10-4226-b2a8-75351c669b09/CSharp%20Language%20Specification.doc – Назва з екрана.
9. E-olimp. On-line контролююча система. [Електронний ресурс] // Веб-сайт «E-olimp». – режим доступу: URL : http://www.e-olimp.com.ua/. – Назва з екрана.

Навчальне видання

ШЕВЧУК Петро Георгійович

НАВЧАННЯ ПРОГРАМУВАННЯ В КЛАСАХ ТЕХНОЛОГІЧНОГО ПРОФІЛЮ ЗАГАЛЬНООСВІТНІХ НАВЧАЛЬНИХ ЗАКЛАДІВ НА ОСНОВІ ВИКОРИСТАННЯ МОВИ C#

Методичні рекомендації для вчителів інформатики

Формат 60x90/16. Папір офсетний.
Гарнітура Times New Roman. Друк різографічний.
Ум. друк. арк. 4.0. Обл. вид. арк. 3.5. Наклад 100.

32