І.О. Завадський

Програмування в курсі інформатики: сучасність та анахронізми

Протягом останніх 10, а може й більше років, шкільне програмування перебуває в стані тяжкої кризи, пов'язаної з категоричним несприйняттям цієї теми учнями, яким вчитися важко, нецікаво, нудно. Деякі вчителі знаходять винуватців цього становища дуже просто: кому нудно, той і винен, діти розучилися мислити, сучасне суспільство веде до дебілізації і т.д., і т.п. Так, зараз стрімко змінюється все: технології, люди, суспільство, час, але протягом 26 років гордо височіє одна скеля, непорушна константа — шкільний курс програмування під прапором «Turbo Pascal». Просто дивовижно, як значна частина інформатичної спільноти вперто й категорично відмовляється бачити зрушення в домінуючій парадигмі програмування, що відбулися не сьогодні і не вчора, а навіть ще ДО появи предмету інформатика, відриваючись все далі і далі від реальності, приходячи в тертя з технологіями, з учнями, з усім навколишнім світом.

На нашу думку, шлях подолання цієї кризи доволі простий, хоча для декого болісний. Потрібно лише визнати застарілими й відмовитися від кількох основоположних тез, повторити крок, зроблений програмістами в усьому світі у 80-х роках (краще пізно, ніж ніколи). Далі ми розглядаємо 3 найтиповіші хибні твердження та детально пояснюємо логіку й методику сучасного підходу до викладання основ програмування на матеріалі перших 3 уроків, оскільки саме в них визначається ідеологія всього курсу.

Типові концептуальні помилки

Типова помилка №1. *«Програма є послідовністю команд, реалізацією певного алгоритму у формі, придатній для виконання комп'ютером.»*

Загалом помилковість даного твердження очевидна, якщо його приміряти до будь-якої сучасної відомої програми. MS Word, Paint, Photoshop, комп'ютерні ігри, СКБД, геть усі програми, включно з ОС MS DOS і операційною оболонкою Norton Commander, не є реалізаціями певного одного алгоритму. 99,9% існуючого на сьогоднішній день програмного забезпечення є *подійно-орієнтованим*. Програма містить багато алгоритмів, кожен з яких виконується внаслідок настання тієї чи іншої події, як-от клацання кнопки, встановлення прапорця, натискання клавіші, надходження повідомлення від іншої програми тощо. Програма є мультиалгоритмічним середовищем (рис. 1,б).

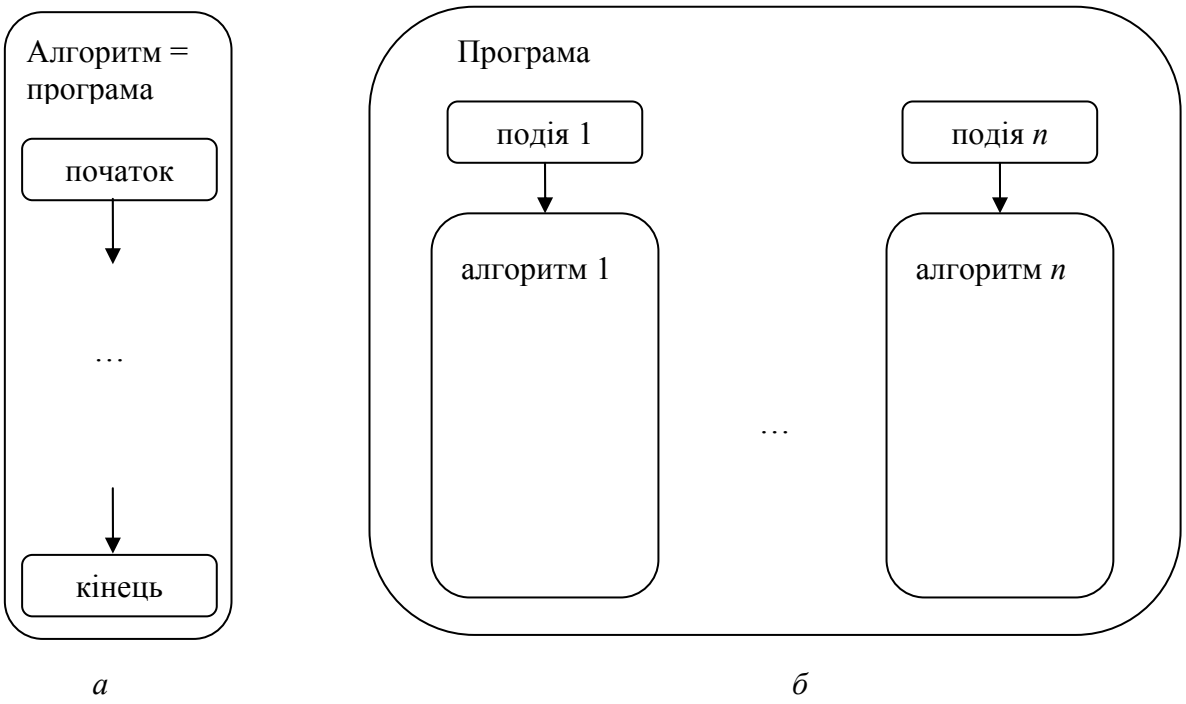

Рис. 1. Концепція прикладної програми: а — 60-і роки XX століття; б — сучасність

Підкреслимо, що йдеться не про об'єктно-орієнтовані програми і не про Windows-програми, а про 99,9% всіх наявних на сьогоднішній день програм взагалі. Навіть переважна більшість DOS-програм вже були подійно-орієнтованими (різниця лише в тому, що їх інтерфейс не був графічним). Тому твердження №1 не просто застаріле, а застаріле в квадраті; воно почало втрачати актуальність з появою такого засобу як командний рядок (а після революції «командного рядка» у галузі ПЗ відбулася ще одна революція — «графічного інтерфейсу»). Це твердження є точним для програмного забезпечення не 80-х і навіть не 70-х, а 60-х років XX століття. На превеликий жаль і сором, воно й досі (у 2011 році!!!) ретранслюється переважною більшістю українських вчителів інформатики на першому уроці з теми «Програмування».

Типова помилка №2. *«Програма є моделлю певного явища, процесу або об'єкта. Основою програмування є моделювання.»*

Погіршений варіант типової помилки №2. «*В основу програмування покладено математичне моделювання.»*

«Ноги» цієї помилки також ростуть з 60-х років XX століття, коли дане твердження дійсно відображало стан речей. Як відомо, ЕОМ I–III поколінь використовувалися переважно саме для моделювання явищ, процесів, математичних задач. Знов-таки, яку сучасну програму не взяти, вона, очевидно, не є моделлю чого-небудь. Моделлю чого є MS Word? Звичайного зошита для записів? Звичайно, ні, адже функції текстового процесора значно ширші. Зошит є лише віддаленим прототипом MS Word, а не об'єктом, який ця програма моделює. А про те, що в основу програм не покладено математичних моделей — годі й казати. Очевидно, що математичні моделі явищ і процесів використовуються в програмах фрагментарно (наприклад, під час перетворення геометричних фігур), а математичне моделювання — це доволі специфічна галузь математики і для її «обслуговування» призначені вузькоспеціалізовані програми.

Типова помилка №3. *«Візуальне програмування у середовищах на кшталт Visual Studio по суті є кнопкодавством, воно відволікає від змісту предмету, підмінює навчання розвагою.»*

По-перше, слід зазначити некоректність терміну «візуальне програмування». У середовищах, подібних Visual Studio, програмний код переважно пишеться вручну або ж генерується автоматизовано, але без використання «візуальних» засобів. Можна казати лише про автоматизовані засоби розробки графічного інтерфейсу програм. По-друге, використання подібних середовищ розробки жодним чином не скасовує, не підміняє і не знижує питомої ваги алгоритмізації у курсі програмування загалом. Єдина відмінність Visual Studio від Turbo Pascal з точки зору шкільного курсу інформатики полягає в тому, що перше середовище орієнтоване на розробку мультиалгоритмічних подійно-орієнтованих програм, (рис. 1,б), а друге — програм, що реалізують один алгоритм (рис. 1,а). Таким чином, Visual Studio орієнтоване на сучасну й узагальнену парадигму програмування, а Turbo Pascal — на застарілу й звужену. А про те, що у Visual Studio код створювати легше, алгоритмічні конструкції інтерпретуються наочніше, мотивація учнів щодо роботи з середовищем вища, розроблені програми цікавіші і, що головне, узгоджуються з принципами функціонування сучасного програмного забезпечення, можна писати дуже довго; це тема окремої статті.

Який вихід?

Шлях подолання кризи такий же очевидний, як і розглянуті вище типові помилки: 1) Трактувати поняття програми як подійно-орієнтованої автоматизованої системи (рис. 1, б), а не єдиного алгоритму (рис. 1, а).

2) Відмовитись від застарілого середовища розробки програм Turbo Pascal на користь Delphi, Visual Studio тощо.

Зазначимо, що ми не пропонуємо зрубати повністю дерево традиційної методики викладання програмування в школі, а лише прищепити його на новий корінь. 90% курсу, тобто вся алгоритміка: умовні оператори, цикли, обробка масивів і т.д. залишається тією ж, хоча й опрацьовується на дещо інакших задачах. Принципово змінюються лише перші 10%, тобто 2–3 уроки.

Далі описано методику проведення перших трьох уроків з теми «Основи програмування» згідно з чинною програмою з інформатики (авт. І.О. Завадський, Ю.О. Дорошенко, Ж.В. Потапова) та експериментальною програмою (авт. І.О. Завадський, Ю.О. Дорошенко, О.П. Пилипчук, Є. В. Шестопалов).

Методика проведення перших трьох уроків з теми «Основи програмування»

Урок 1

Зміст навчального матеріалу

Складові програми: дані, логіка, інтерфейс. Поняття об’єкта у програмуванні. Елементи інтерфейсу користувача як об’єкти. Параметри об’єкта. Поняття події та обробника події. Поняття про методи об’єкта.

Методичні вказівки

На початку уроку варто попросити назвати учнів якомога більше програм, а потім поставити кілька проблемних запитань.

Запитання №1: Що спільного мають всі названі програми і чим вони відрізняються від непрограмних об'єктів, скажімо книжки чи парти?

Відповідь: програми виконують дії.

Запитання №2: А над чим програми виконують дії?

Відповідь: на ***даними***.

Запитання №3: А звідки програма «знає», яку дію їй зараз виконувати?

Відповідь: Це їй вказує користувач.

Запитання №4: Яким чином, за допомогою яких засобів користувач може вказати програмі, які дії виконувати?

Відповідь: за допомогою ***інтерфейсу***.

Слід зауважити, що дії програм можуть бути різними залежно від обставин, у яких вони виконуються. Наприклад, якщо в MS Word не виділено фрагмента тексту і ми натискаємо клавішу **Del**, то буде видалено лише один символ, якщо фрагмент був виділений, то видалений буде він увесь. Таким чином, програми не просто виконують дії, а реалізують певну ***логіку дій***.

У такий спосіб ми підводимо учнів до розуміння того факту, що складовими програмного забезпечення є логіка, дані та інтерфейс.

Запитання №5: З чого складається інтерфейс?

Відповідь: кнопки, меню і т.п.

Запитання №6: А як це все назвати одним словом (згадуємо початок 9 класу)?

Відповідь: це ***об'єкти***.

Слід закцентувати на тому, що тут ми маємо справу не з якими-небудь об'єктами, а саме з об'єктами у програмах, або ***програмними об'єктами*** і зауважити, що такими об'єктами можуть бути не лише елементи інтерфейсу. Попросіть учнів назвати програмні об'єкти, що не є елементами інтерфейсу. Правильними відповідями можуть бути: абзац тексту, документ, піксель, коло тощо.

Запитання №7: Згадайте, чим характеризуються, що мають всі об'єкти?

Відповідь: ***параметри*** і поведінку (сукупність дій), або особливість, стан (сукупність значень параметрів) і поведінку.

Слід відобразити якийсь програмний об'єкт (скажімо, кнопку або абзац тексту) і попросити учнів назвати його параметри. Для кнопки це може бути розмір, колір, стиль шрифту, для абзацу тексту — величина відступів, міжрядковий інтервал.

Тепер запропонуйте розглянути уважніше взаємодію користувача з програмою.

Запитання №8: Як назвати одним словом всі акти взаємодії користувача з програмою: натискання клавіш, клацання кнопок і т.д.? Що це «з точки зору» програми?

Відповідь: це ***події***. У відповідь на настання кожної події програма виконує певну послідовність дій. Той фрагмент програми, який реагує на подію, називається ***обробником події***.

Варто повибирати різні команди меню, поклацати кнопки якоїсь програми й прокоментувати, які при цьому трапляються події та що виконують обробники подій.

Слід звернути увагу учнів на те, що всі події пов'язані з тими чи іншими програмними об'єктами: клацання *кнопки*, виділення *символу* тощо, і тому обробники подій теж було б логічно пов'язувати з певними об'єктами. Власне, так і є: фрагмент програми, що реалізує

логіку обробника події, як правило зв'язаний з якимось об'єктом і називається його ***методом***.

Підсумовуючи матеріал уроку, слід відобразити та прокоментувати спрощену схему роботи прикладної програми, зображену на рис. 1, б.

Урок 2

Зміст навчального матеріалу

Поняття алгоритму, виконавця алгоритму. Поняття мови програмування, програмного коду, компілятора, середовища розробки програм. Життєвий цикл програмного забезпечення. Графічне подання алгоритмів. Конструювання лінійних алгоритмів та їх графічне подання.

Методичні вказівки

Запитання №1: Давайте подивимося «всередину» методів програмних об'єктів. Як ви думаєте, що містить такий метод?

Відповіді учнів можуть бути найрізноманітнішими. Частково правильними відповідями є: команди, оператори, коди. Ви маєте дати правильну відповідь: кожен метод об'єкта реалізує певний ***алгоритм*** — і дати означення цього терміну.

Зазначте, що алгоритми використовуються не тільки у програмуванні і наведіть приклади словесного опису алгоритмів (наприклад, алгоритму переходу через вулицю). Зазначте, що кожен алгоритм має ***виконавця*** і наведіть приклади таких виконавців. Виконавцем алгоритмів, реалізованих у програмах, є комп'ютер.

Запитання №2: Яким чином «нерозумний» комп'ютер «розуміє» алгоритми?

Відповідь: алгоритми записують «зрозумілою» для комп'ютера мовою — ***мовою програмування***.

Потрібно провести невеличку демонстрацію, для якої знадобиться і готова програма, і програмний код. Спочатку слід показати виконання якоїсь дії у готовій програмі, а потім перейти у середовище програмування, двічі клацнути на об'єкті, з яким пов'язано цю дію, й прокоментувати відображений програмний код. Достатньо, щоб це була програма, яка у відповідь на клацання кнопки виводить повідомлення «Hello, World!», хоча незле, щоб там був і умовний оператор, який аналізує вміст текстового поля і відображує щось на кшталт повідомлення «від'ємне число» чи «додатне число». Слід наголосити, що програмний код в цілому читабельний, він фактично складається з англійських слів і, трохи потренувавшись, будь-яка людина навчиться його розуміти.

Запитання №3: Згадайте матеріал 9 класу: який саме компонент комп'ютера виконує програми?

Відповідь: процесор.

Запитання №4: У якому вигляді мають бути записані команди, щоб їх міг виконати процесор?

Відповідь: у вигляді машинного коду, тобто одиничок і нуликів, і в такому вигляді зрозуміти команди людині дуже важко.

Отже, ми маємо певне неузгодження: програмісти записують програми більш-менш зрозумілою англізованою мовою, а процесор може виконувати тільки зовсім незрозумілі для нас послідовності 0 і 1. Це неузгодження знімає ***компілятор*** — спеціальний

програмний засіб, який перетворює програму, записану мовою програмування, на машинний код.

Компілятор є складовою ***середовища програмування***. Потрібно назвати і продемонструвати середовище і його складові (засоби візуального конструювання інтерфейсу, вікна програмного коду і т.п.).

Запитання №5: Тепер ви знаєте, що таке середовище й мова програмування, програмний код, компілятор. Всі ці терміни стосуються процесу розробки програм. Попробуйте описати його в цілому. На які етапи він поділяється?

Відповідь:

1) формулювання задачі;

2) проектування інтерфейсу (слід зауважити, що в серйозних етапах цей етап полягає у проектуванні не лише інтерфейсу, а об'єктно-орієнтованої архітектури всієї програми);

3) розробка алгоритмів;

4) написання програмного коду;

5) компіляція;

6) виконання програми, тестування й налагодження;

Зауважте, що якщо до цього додати ще експлуатацію програми, то отримаємо повний ***життєвий цикл*** програмного забезпечення. Програма подібна до живої істоти: вона народжується, розвивається, живе і вмирає, коли її перестають використовувати та забувають.

Чи не найважливішим етапом створення програм є розробка алгоритмів.

Запитання №6: У якій формі можна подавати алгоритми?

Відповідь: у вигляді словесного опису, програмного коду та ін.

Ви маєте зазначити, що алгоритми можна зображувати і графічно, продемонструвати й обговорити блок-схему лінійного алгоритму та дати вправу на складання лінійного алгоритму. Це може бути алгоритм якоїсь відомої дії з попередніх тем курсу інформатики, скажімо відкриття документа в MS Word. Учні мають подати цей алгоритм у вигляді блок-схеми.

Урок 3

Зміст навчального матеріалу

Принципи роботи у середовищі розробки програм. Програмний проект, його відкриття, виконання та збереження. Поняття оператора та різновиди операторів. Покрокове виконання та аналіз роботи готових програм. Створення найпростішого програмного проекту.

Методичні вказівки

Ви маєте відобразити папку готового програмного проекту в середовищі ОС, зазначити, файл з яким розширенням є власне файлом проекту, відкрити проект, двічі клацнувши цей файл, та показати його у середовищі розробки. Зверніть увагу на такі елементи середовища розробки, які вікно конструктора форми, панель елементів керування (Toolbox) та вікно параметрів (Properties) вибраного елемента керування.

Запитання №1: Чим є всі елементи керування (згадайте перший урок)?

Відповідь: програмними об’єктами.

Запитання №2: Що мають всі програмні об’єкти (згадуємо перший урок)?

Відповідь: параметри та методи.

Продемонструйте, де записані параметри об’єкта «кнопка» та їхні значення (вікно Properties), змініть значення певного параметра елемента керування (скажімо, кольору кнопки) у вікні параметрів та зверніть увагу учнів на те, як змінився сам елемент керування.

Клацніть двічі кнопку — ви перейдете у вікно коду. Поясніть, що це код обробника події клацання кнопки. Покажіть, де в коді ***оператори*** (їх має бути 2–3). Поясніть, що оператор — це елемент програми, який виконує якусь одну дію. Поясніть призначення всіх операторів у вікні коду.

Запустіть програму й клацніть кнопку, яку ви розглядали. Зверніть увагу, як виконуються щойно розглянуті оператори.

Виконайте програму в покроковому режимі. Після виконання кожного оператора прокоментуйте його дію.

Нарешті учні мають виконати 3 вправи:

Вправа 1. Учні відкривають і переглядають у покроковому режимі програму, у якій є одна кнопка і 5–6 операторів в обробнику події її клацання, до яких вже додано коментарі. Завдання учнів — проаналізувати дію програми.

Вправа 2. Учні відкривають і переглядають у покроковому режимі іншу програму, у якій є одна кнопка і 5–6 операторів в обробнику події її клацання, до яких не додано коментарів. Завдання учнів — дописати коментарі, що пояснюють дію операторів, та зберегти проект.

Вправа 3. Учні створюють, зберігають і запускають програму, що виводить повідомлення «Hello, World!» у відповідь на клацання кнопки. Учитель має попередньо продемонструвати виконання вправи.