Завадський Ігор Олександрович
к. ф-м. н., доцент, факультет кібернетики КНУ ім. Шевченка, м. Київ
zava@ukr.net

Розвивальна інформатика

Якщо не враховувати інформатику, то всі класичні шкільні навчальні дисципліни можна поділити на дві категорії: знаннєво-компетентнісні й розвивальні. До другої категорії повністю можна віднести тільки математику й образотворче мистецтво, і частково – фізику й літературу. У решті дисциплін набуття знань, умінь, навичок, компетенцій є самоціллю, вони можуть розвивати мислення лише принагідно і фрагментарно. Сучасна інформатика, очевидно, належить до обох категорій водночас і така роздвоєність призвела до бурхливої дискусії в інформатичній спільноті, що виросла із протиставлення «алгоритміки, вона ж прикладна математика» і «так званих технологій, по суті різновиду трудового навчання», пропозицій поділити предмет на два інших, появи цілих партій прихильників того чи іншого підходу. У цій статті ми спробуємо показати, що згадане протиставлення є спекулятивним, а різноманітні прикладні програмні продукти можуть стати чудовим засобом розвитку мислення.

Що має розвивати інформатика

Насамперед слід дати відповідь на питання: що саме, які мисленнєві здібності може і повинна розвивати інформатика? Часто, особливо від тренерів програмістів-олімпіадників, можна почути відповідь на кшталт: «приблизно ті самі, що й математика». Дійсно, ідеальна олімпіадна задача з програмування – це на 90% математична задача, що вимагає від учня передусім вигадливості, винахідливості. Власне ці якості математика і розвиває, а може розвивати й інформатика. Однак якби інформатика розвивала тільки винахідливість, це була би лише «недоматематика», «напівматематика» чи, у кращому разі, «ще одна математика». Звичайно, така дисципліна була би просто недоцільною.

Насправді коло розумових здібностей, які може і повинна розвивати інформатика, значно ширше. Сам метод дослідження, сама постановка типової задачі в математиці та інформатиці кардинально різняться. Математична задача створює для учня прокрустове ложе жорстких, штучно «закручених» умов, з яких той має знайти вихід. І чим жорсткіші умови, чим неможливішим видається вихід, тим цікавішою і сильнішою вважається задача. В інформатиці такий підхід застосовується хіба що в олімпіадному програмуванні, а вже у програмуванні промисловому, не кажучи про інші інформаційні технології, постають задачі зовсім іншого типу. Вони не складні чи не дуже складні з логічної точки зору. Їхня складність, як правило, полягає у необхідності правильно структурувати величезні обсяги даних, внести порядок у хаос, не просто знайти вихід, а відшукати оптимальний шлях до нього. У фірмах, що займаються розробкою програмного забезпечення, цінують не стільки дипломи переможців олімпіад, скільки чіткість мислення, порядок у голові, здатність писати добре структурований і читабельний код, уміння працювати в команді і т.п.

Зауважимо також, що інформатика є значно ближчою, ніж математика, до творчості у широкому розумінні. Програміст, розробник бази даних, а тим більше комп'ютерний дизайнер – це творець, перед ним чистий аркуш і досить загальна задача без жорстких обмежень, що має, як правило, багато коректних способів розв'язання, і його завдання,

повторимося, полягає не в тому, щоб просто цю задачу розв'язати, а в тому, щоб знайти оптимальний чи принаймні ефективний спосіб розв'язання.

Тому ми вважаємо, що інформатику некоректно прив'язувати до математики чи як-небудь корелювати з нею і саме такого роду некоректність призвела до виникнення згаданої у преамбулі дискусії. Інформатика – це зовсім окрема дисципліна, що має свій шлях розвитку, свої цілі і завдання, зокрема розвивальні.

Отже, спробуємо такі цілі і завдання окреслити. З нашого погляду, навчання інформатики має в учнів розвинути:

А. Алгоритмічне мислення.

Б. Структурне мислення.

В. Творчі здібності.

Під алгоритмічним мисленням найчастіше мають на увазі такі вміння:

- визначати послідовність дій, які необхідно виконати для розв'язання певної задачі;
- подавати алгоритми в певному формальному вигляді і виконувати їх;
- використовувати в алгоритмах алгоритмічні структури, тобто розгалуження і повторення;
- опрацьовувати величини, сталі й змінні.

Завдання розвитку алгоритмічного мислення ставилося ще в першому курсі інформатики 1986 року і тому стосовно цього аспекту сказано й написано достатньо багато. Ми не будемо повторюватися, а натомість зосередимо увагу на другому розвивальному компоненті.

Бази даних – основа структурного мислення

Розвиток алгоритмічного мислення, безумовно, був і залишається одним із найважливіших завдань курсу інформатики. Проблема виникає тоді, коли він проголошується єдиним завданням, самоціллю, а прикладні програми розглядаються не як полігон для застосування алгоритмів, а як чужинці, що тільки заважають «правильному та інтелектуальному» навчанню програмуванню, їх використання вважається «кнопкодавством» і т.п. Найприкріше, коли у категорію «вигнанців» потрапляють такі продукти, як табличний процесор і система керування базами даних, хоча насправді робота з ними є не менш розвивальною, ніж програмування. Цього не помічають тому, що вони дають змогу розвинути інший, ніж алгоритмічний, тип мислення, який ми назвемо мисленням *структурним*. До цього типу мислення належать такі вміння:

- визначати параметри об'єктів та їх можливі значення;
- класифікувати явища та об'єкти;
- знаходити структурні та ієрархічні зв'язки між класами об'єктів, класифікувати зв'язки;
- розв'язувати задачі з обробки структур даних (передусім – формалізовувати вимоги щодо відбору даних за певними критеріями).

Образно кажучи, мова йде про вміння «розкладати дані по поличках», яке в сьогоднішньому переповненому інформацією світі є надзвичайно важливим. І в інформатиці є тема, яка, у разі застосування правильної методики, виховує саме це вміння. Йдеться про бази даних. Цю технологію категорично не можна перелічувати через кому в ряді «офісних» технологій. Вона незрівнянно фундаментальніша, ніж, скажімо, обробка текстів чи створення презентацій. СУБД по суті не є офісною програмою, MS Access

включено у пакет Office лише з певних маркетингових міркувань. Фундаментальне значення цієї технології в курсі інформатики пояснюється не стільки тим, що база даних є складовою ледь не кожної більш-менш великої програмної системи, скільки тим, що в основі будь-якої бази даних лежить модель певної предметної області, а конструювання таких моделей розвиває три перших з перелічених вище чотирьох компонентів структурного мислення.

Перші 3–6 уроків з теми «Бази даних» має бути присвячено побудові моделей «сутність-зв'язок» предметних областей. Це теоретичні уроки, які мають проходити або з вимкнутими комп'ютерами, або з використанням редактора моделей «сутність-зв'язок», наприклад http://www.gliffy.com. Вивчення баз даних не можна починати з роботи в середовищі СКБД, оскільки тоді технологічні аспекти замулюють сутність справи, учневі важко відрізнити, які відомості важливіші – про склад меню «Файл» у MS Access чи про головний принцип семантичного моделювання. А головне в цій темі – навчитися виявляти в предметних областях сутності (це все одно, що класифікувати об'єкти) та коректно і ненадлишково встановлювати між ними зв'язки. Таке уміння стане міцним фундаментом для подальшого вивчення і СКБД, і об'єктно-орієнтованого програмування, і загалом сформує в учня певні мисленнєві схеми. Це один з найважливіших розвивальних аспектів курсу інформатики.

Електронна таблиця – полотно для інформатичної творчості

Чудовим засобом пропедевтики «серйозних» розвивальних тем, засобом комп'ютерного моделювання, а також середовищем, де можна застосувати для розв'язання практичних задач навички алгоритмічного мислення, є табличний процесор (ТП). Справді, електронна таблиця – це полотно, величезне поле для творчості, тільки не художньої, а формульно-числової, тобто ідеальний навчальний інструмент з огляду на проголошені нами вище завдання курсу інформатики.

Усі задачі, що розв'язуються засобами ТП, можна поділити на 3 класи:

А. Робота з даними в окремих клітинках.

Б. Обробка рядів даних.

В. Обробка наборів однотипних об'єктів.

Задачі класу А розв'язуються за допомогою однієї або кількох формул і не потребують роботи з діапазонами клітинок. Хоча ці задачі найменш цікаві і більшість із них можна розв'язувати без табличного процесора, наприклад у калькуляторі, вони, однак, мають певне розвивальне значення, адже, як правило, в такій задачі проектується і застосовується модель економічного, фізичного чи біологічного процесу або математичної задачі.

Вкрай важливо, що, вводячи лише кілька формул, у табличному процесорі можна реалізувати лінійний алгоритм або алгоритм із розгалуженням і виконати його для різних наборів вхідних даних. Аналогом умовного оператора в MS Excel є функція IF, клітинки – це аналоги змінних, їхні адреси – імена змінних (а ще краще надавати клітинкам змістовні імена), вміст – значення змінних, типи даних в електронній таблиці майже нічим не відрізняються від типів даних у програмуванні. У ТП є також логічні функції, що дають змогу будувати складені логічні вирази, таблиці істинності булевих функцій і опановувати основи алгебри логіки загалом. «ТП-програміст» не обмежений синтаксичними умовностями мов програмування, він пише тільки той «код», який безпосередньо стосується алгоритму, але головним є те, що ТП процесор надає дуже зручне і просте середовище для застосування алгоритмів до розв'язання практичних задач.

З точки зору побудови навчальних програм з інформатики такий зв'язок між алгоритмами і табличним процесором процесором означає, що в тему, присвячену

формулам і логічним функціям у ТП, слід вводити такий матеріал, як «моделювання алгоритмів в ЕТ», а розміщувати цю тему варто відразу після чи безпосередньо перед темою, присвяченою вивченню лінійних алгоритмів і алгоритмів з розгалуженнями.

Багато задач класу Б тісно пов'язані з алгоритмічною конструкцією повторення, яка моделюється копіюванням в електронній таблиці деякої формули в діапазон. Але найважливіший випадок – коли копіюється рекурентна формула, наприклад, формула для обчислення ряду чисел Фібоначчі або факторіалу. У цьому разі формула, яку вводить учень у клітинку електронної таблиці, є нічим іншим, як тілом циклу. Звичайно, розвивальний момент подібної обчислювальної задачі полягає саме у винайденні цієї формули, а не в написанні слів FOR, PROGRAM, оголошенні змінних і т.п. Тобто у ТП учень відразу опрацьовує саму суть задачі, а в середовищі програмування виконує багато зайвих дій. Таким чином, навчання моделюванню простих ітеративних обчислень за допомогою ТП є більш ефективним.

Розвивальний аспект реалізується і в іншому типі задач класу Б – задачах на інтерпретацію, вибір типу й побудову діаграм. Як показало міжнародне порівняльне дослідження рівня природничо-математичної освіти TIMSS, характерною рисою українських школярів є абсолютна безпорадність у розв'язанні задач на графічну інтерпретацію даних. Дійсно, цього матеріалу немає у програмі жодної зі шкільних дисциплін, а в інформатиці він часто зводиться до навчання користуванню майстром діаграм, хоча насправді питання про те, які кнопки натискати в цьому засобі, абсолютно другорядне, а першочерговим завданням є набуття таких умінь:

- уміння інтерпретувати діаграми;
- уміння добирати тип діаграми залежно від природи вхідних даних.

Для вибору типу діаграми слід застосовувати певний алгоритм, знайти який можна запропонувати самим учням. Чудова розвивальна задача і ще один зв'язок ТП з алгоритмізацією!

Задачі класу В (сортування, фільтрація, обчислення проміжних підсумків, функції для роботи з базами даних) – це пропедевтика реляційних баз даних. У них електронна таблиця використовується як однотаблична реляційна база даних. Рядки таблиці – це записи, кожен із яких містить інформацію про певний об'єкт, а стовпці – поля, що містять значення параметрів об'єктів. Структури даних такого типу у сучасному світі надзвичайно поширені і вміння працювати з ними – одна з найзначущих інформатичних компетенцій. Крім компетентнісного аспекту, у цих задачах можна знайти й численні розвивальні, пов'язані із останнім пунктом у наведеному вище переліку компонентів структурного мислення. Наприклад, запис умов відбору рядків у розширеному фільтрі потребує застосування математичної логіки.

Описані нами змістові зв'язки теми «Електронні таблиці» з двома іншими темами зображено на рис. 1.

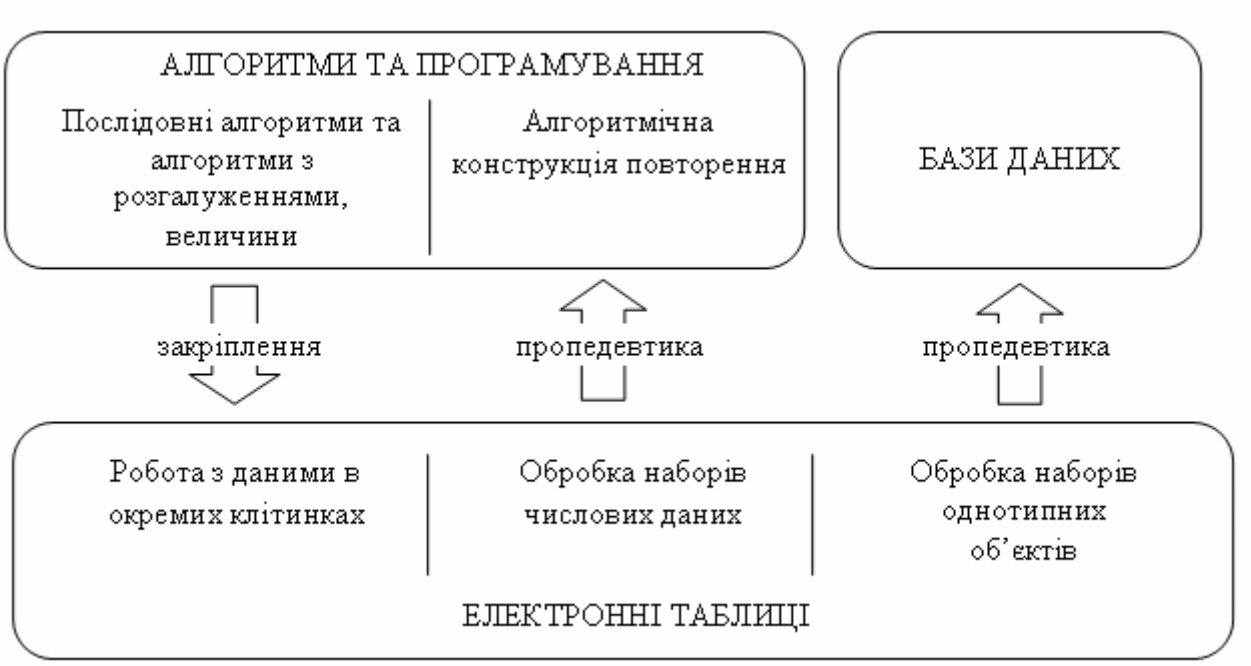

Рис. 1. Змістові зв'язки розвивальних аспектів у різних темах курсу інформатики

Програмування – не лише кодування

Повертаючись до питання про зв'язок ТП із алгоритмізацією та програмуванням, зазначимо, що ТП, звичайно, не є засобом кодування (хоча «код» ТП-програм і можна побачити у режимі відображення формул), однак це не означає, що він не є засобом програмування взагалі. ТП, так само, як і Скретч, – це середовища *нетекстового* програмування. Програмування у Скретчі є *візуальним*, а в ТП – *табличним*.

Загалом слід зазначити, що парадигма програмування як написання коду поступово відходить у минуле. У 70-х роках XX століття програміст уподібнювався письменнику, що сідав за комп'ютер і писав програму приблизно так, як пишуть роман. Зараз створення програм більше нагадує будівництво, а «програміст» – це така сама збірна назва багатьох спеціальностей, як і «будівельник». Важливо не тільки, як кладуть цеглу (пишуть код), але й якою є архітектура споруди (програмної системи). Для проектування архітектури програмних систем призначено суто візуальну мову UML, яка, строго кажучи, є не мовою, а набором правил опису різнотипних діаграм. Немає жодних причин, щоб не вивчати окремі аспекти UML у середній школі. Це не менш корисно і цікаво, ніж писати програмний код.

Варто також згадати про численні непроцедурні мови, наприклад, HTML чи SQL. Їх синтаксис дає змогу описувати вимоги до інформаційних систем. Тобто людина за певними формальними правилами описує, ЩО вона хоче отримати від інформаційної системи, не описуючи, ЯК цього досягти, оскільки «як» генерується зі «що» автоматично. Не слід думати, що непроцедурні мови менш розвивальні за мови програмування. Можна сформулювати безліч запитів, запис яких мовою SQL потребує нетривіального застосування логіки і є не менш цікавою задачею, ніж задачі на олімпіадах з програмування всеукраїнського рівня. В умовах відбору записів у мові SQL використовуються не тільки логічні функції «НЕ», «І» та «АБО», але й оператор «належить», квантори існування та загальності та інші засоби. Інакше кажучи, у програмуванні застосовується апарат логіки висловлювань, а в SQL – числення предикатів. Тому мова SQL є значно потужнішим, ніж програмування, засобом розвитку

логічного мислення (ще одного розвивального компоненту курсу інформатики, який можна віднести як до структурного мислення, так і до алгоритмічного).

Висновки

Ця стаття не претендує на вичерпність розкриття піднятої в ній теми. І найменш вичерпними є переліки розвивальних напрямів курсу інформатики, компонентів алгоритмічного та структурного мислення. Фактично ми лише навели кілька прикладів того, як можна орієнтувати курс інформатики у розвивальному напрямі завдяки двом темам: «Робота з електронними таблицями» та «Бази даних». Ми не вказали розвивальних можливостей інших тем, зовсім не розкрили зміст третього розвивального компонента – розвитку творчих і комунікативних здібностей, ми постійно повторювали слова «моделювання», «об'єкт», «оптимізація», не зробивши з цього ніяких висновків (а це не що інше, як наскрізні змістові лінії курсу, і його розвивальні аспекти має бути розглянуто в їхньому розрізі також). Все це може стати предметом наступних досліджень і статей, у яких незмінним залишатиметься один постулат: інформатика може і повинна бути розвивальною. Кожен урок має спонукати чи навіть змушувати учня зробити якісь рухи головним мозком. А інакше для чого взагалі навчатися?