Медведєва М.О.,
к.пед.н., доцент кафедри вищої математики
Уманського державного педагогічного університету
імені Павла Тичини

ДОСВІД НАВЧАННЯ ДИСКРЕТНОЇ МАТЕМАТИКИ У ЗАРУБІЖНИХ УНІВЕРСИТЕТАХ

Анотація: У статті аналізуються підходи до навчання дискретної математики у зарубіжних університетах. Виділено шість підходів до вступних курсів з програмування. Імперативний, об'єктний та функціональний підходи ґрунтуються на тому чи іншому виді програмування. Альтернативні підходи – алгоритмічний, апаратний та з максимальним охопленням матеріалу.

Аннотация: В статье анализируются подходы к обучению дискретной математике в зарубежных университетах. Выделено шесть подходов к вступительным курсам по программированию. Императивный, объектный и функциональный подходы основаны на том или ином виде программирования. Альтернативные подходы – алгоритмический, аппаратный и с максимальным охватом материала.

Summary: The article analyzes the approaches to teaching discrete mathematics in foreign universities. Identified six approaches to the introductory programming course. Imperative, object and functional approaches are based on some form of programming. Alternative approaches – algorithmic, and hardware with maximum coverage of the material.

Постановка проблеми. Теоретичний розділ будь-якої науки базується на математичних методах дослідження. Це стосується й інформатики. Вона використовує методи математики для побудови і вивчення моделей обробки,

передачі і використання даних, створює той теоретичний фундамент, на якому будується вся інформатика. Саме таким фундаментом і є математичний апарат дискретної математики.

Вивчення основних розділів курсу дискретної математики, які необхідні для оволодіння сучасним математичним апаратом із метою подальшого його застосування при вивченні інформатичних та математичних дисциплін, а також при проведенні самостійних наукових досліджень займає важливе місце в системі підготовки майбутнього вчителя інформатики та математики, оскільки сприяє як формуванню наукового світогляду в цілому, так і математичної культури зокрема.

Було встановлено, що дискретна математика розглядається багатьма науковцями як невід'ємна складова навчання сучасних фахівців галузі знань «Системні науки і кібернетика». Для інтенсифікації навчального процесу, потрібно провести теоретичний аналіз друкованих та електронних джерел, повідомлень для виявлення стану проблеми – обрання оптимального підходу до навчання дискретної математики – та засобів і форм її вирішення, що використовуються у ВНЗ України та світу.

Постановка завдання. Дана стаття присвячена аналізу підходів до навчання дискретної математики у зарубіжних університетах.

Виклад основного матеріалу. Дослідження підходів до навчання інформатики в університетах США, Канади, Ізраїлю, Великобританії, Японії, Австралії, надало можливість класифікувати їх за принципом побудови навчальних планів курсів і місцем в них дискретної математики. Основними джерелами даних стали звіти таких організацій, як Комп'ютерне співтовариство Інституту Інженерів по Електротехніці і Електроніці (IEEE Computer Society), Асоціація по обчислювальній техніці (Association of Computing Machinery), Австралійське комп'ютерне співтовариство (Australian Computer Society), Британське комп'ютерне співтовариство (British Computer Society), Японське співтовариство з опрацювання даних (Information Processing Society of Japan).

Кожна із названих організацій має в своїй структурі Раду з питань освіти. До обов'язків зазначених рад входить розробка навчальних планів і рекомендацій щодо побудови навчальних планів з дисциплін комп'ютерного циклу. Звіти і рекомендації складаються за участю викладачів провідних ВНЗ кожної країни, загалом у складанні звітів беруть участь 150-400 чоловік.

Математичні методи та формальні міркування є складовою частиною більшості галузей інформатики. Теорія визначається як один з трьох основних базисів інформатики, і можна припустити, що цей принцип не застарів і на сьогодні. Інформатика залежить від математики та її фундаментальних визначень, аксіом, теорем і методів доведення. Крім того, математика надає можливість працювати з поняттями, що належать до інформатики, конкретними засобами аналізу та верифікаціями, а також є теоретичною основою для розуміння важливих ідей інформатики. Наприклад, функціональне програмування та вирішення проблем засноване на математичних концепціях і нотаціях для функцій; аналіз алгоритмів безпосередньо залежить від таких розділів математики, як комбінаторика та теорія ймовірностей; аналіз паралелізму та запобігання блокуванням вимагає використання теорії графів; нарешті, верифікація програм і аналіз обчислень базуються на формальній логіці і дедукції. Таким чином, для ефективнішого засвоєння студентами теоретичних основ дисципліни, до програм навчання інформатики необхідно включати достатній обсяг математики. Такий підхід до вивчення теоретичних основ інформатики реалізується через вивчення дискретної математики.

Враховуючи провідну роль математики в інформатиці, фахівці П. Деннінг та Д. Сомер рекомендують до програми навчання включати математичні концепції якомога раніше і як можна частіше. Основні математичні концепції мають бути представлені студентам ще на перших курсах навчання у ВНЗ, а на старших курсах їх необхідно регулярно використовувати [1].

Хоча коледжам і університетам доводиться адаптувати свої вимоги до попередніх знань абітурієнтів для того, щоб представити локальні потреби та можливості, пояснює Л. Блестер, досить важливо використовувати на старших

курсах математичні навички, розвинені на молодших курсах. Крім того, відомості про попередні вимоги до рівня знань, умінь та навичок студентів необхідно зафіксувати у формальних документах факультету [2].

В результаті проведеного аналізу було визначено, що програми з інформатики містять необхідні розділи математики, зокрема, дискретної математики. Для цього виокремлено галузь знань, яка складається з дискретної математики, обов'язкової для університетської програми. Ця галузь визначає теми і розділи, що, на нашу думку, є найбільш важливими для основного курсу. **Дискретна математика** може бути введена як окремий курс або інтегрована в процес навчання інформатики. В будь-якому випадку реалізації, особливе значення впродовж усього навчання віддається застосуванню методів дискретної математики.

Спеціальною комісією CC2008 розроблено рекомендації щодо навчання інформатики у ВНЗ, в яких подано певні поради щодо математичного змісту в курсі інформатики [3]:

Дискретна математика. Всі студенти мають бути ознайомлені з прийомами дискретної математики. Рекомендується дати більше одного курсу в цій галузі; програми з інформатики повинні висвітлювати обов'язкові теми дискретних структур.

Додатковий курс математики. Студенти повинні отримувати додаткові знання з математики для розвитку своєї майстерності в цій галузі. Додатковий курс математики може складатися з різних курсів, включаючи статистику, математичний аналіз, лінійну алгебру, чисельні методи, теорію чисел, геометрію або логіку. Вибір повинен залежати від цілей програми навчання, потреб навчального закладу і потреб самих студентів.

Як уже було зазначено, студентам, які вивчають інформатику, пропонують вивчати дискретну математику якомога раніше, переважно в перший рік навчання. Для досягнення зазначеної мети в американських і європейських університетах відзначено дві стратегії:

1. Вимагати від студентів, які вивчають інформатику, прослухати курси з дискретної математики одночасно з вступним циклом. Описи курсів включають дві реалізації курсу з дискретної математики: семестровий курс, який охоплює основний обсяг матеріалу з області дискретних структур в сукупності знань, і більш повніший, але повільніший варіант, розділений на два курси, в яких викладається весь необхідний матеріал, а також деякі корисні додаткові теми.

2. Інтегрувати, як мінімум, частину матеріалу з дискретної математики безпосередньо до вступного циклу інформатики так, щоб студенти могли краще зрозуміти, як математичні інструменти застосовуються в практичному контексті. Використання такого підходу надає певні переваги. Однак, в процесі вивчення деяких тем, необхідно переконатися, що студенти в достатній мірі володіють знаннями з дискретної математики, необхідними для опанування навчального матеріалу. Враховуючи обсяг знань в області дискретних структур, вбачається неможливим об'єднання всіх необхідних тем у вступному циклі з інформатики без приєднання додаткового математичного курсу. Тому типові реалізації мають включати деякий матеріал безпосередньо до циклу інформатики, але при цьому зберігати окремий курс дискретних структур з метою забезпечення необхідного обсягу матеріалу. Реалізація трьох курсів з застосуванням підходу "з максимальним охопленням матеріалу" використовує модель впровадження математичних модулів безпосередньо у вступних курсах з інформатики [4].

Дослідження основних підходів до вступних курсів з програмування, що практикуються в зарубіжних університетах показало, що їх існує шість (рисунок). З яких: три є підходами, що ґрунтуються на тому чи іншому виді програмування (ліва частина на рисунку), інші три – альтернативні.

Самим традиційним із усіх зазначених підходів вважається підхід «з орієнтацією на імперативне програмування» (надалі – «імперативний підхід»). Практика показує, що відомо дві реалізації моделі «з орієнтацією на імперативне програмування», однією з яких пропонується вивчення матеріалу,

який показано нижче, протягом трьох семестрів, а інша – протягом двох семестрів:

- основи програмування;
- об'єктно-орієнтована парадигма;
- структури даних і алгоритми;
- вступ в програмування;
- абстракція даних.

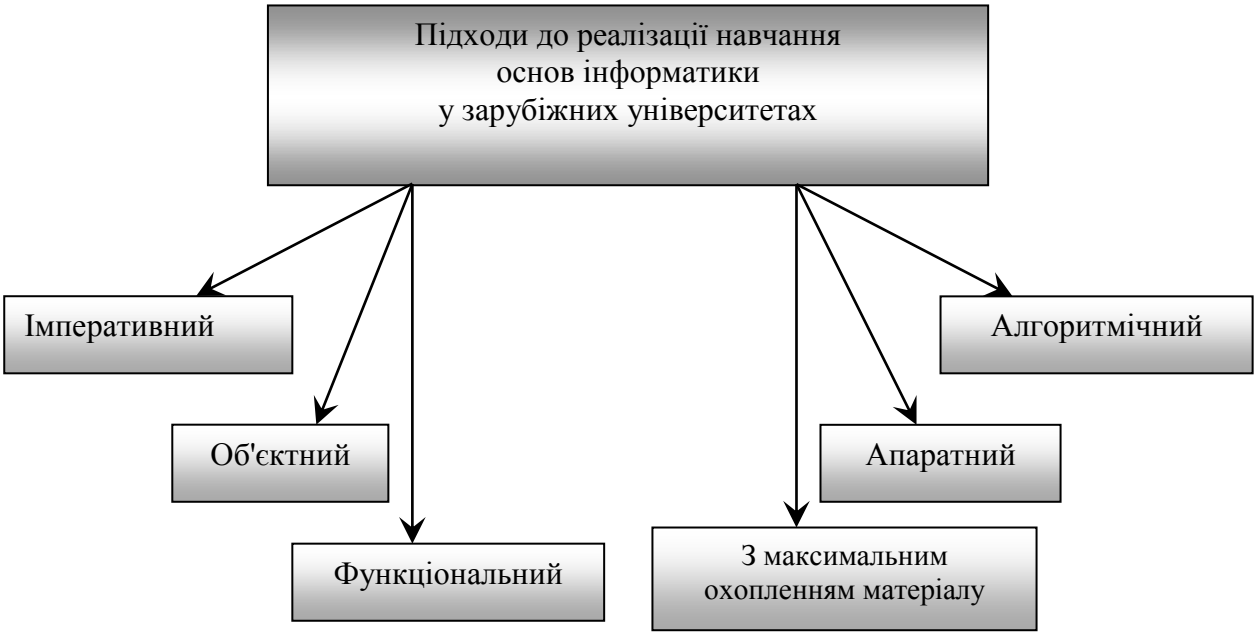

Рис. Підходи до навчання основ інформатики у зарубіжних університетах

В освіті двосеместрова модель відома як більш традиційна реалізація. Вона пропонує вступ в програмування при застосуванні імперативного підходу. Надалі, отримані знання розширюються шляхом ознайомлення студентів з великою частиною матеріалу традиційного курсу, але з концентрацією на програмуванні в об'єктно-орієнтованій парадигмі. Трьохсеместрова реалізація пропонує ширше охоплення більшості тем, що дозволяє студентам краще опанувати фундаментальні концепції. Крім того, в цьому випадку, навчальною програмою передбачено виокремлення певного навчального часу для вивчення додаткових тем. Зазначене означає, що студенти отримують можливість ширшого бачення дисципліни.

Виокремлюючи основну відмінність імперативного підходу від об'єктно-орієнтованої моделі, необхідно відзначити, що вона полягає в акценті та дотриманні порядку тем, що вивчаються першими. Навіть якщо навчання ведеться з використанням об'єктно-орієнтованої мови програмування, перший курс фокусується на імперативних аспектах цієї мови: виразах, управляючих структурах, процедурах і функціях, а також інших центральних елементах традиційної процедурної моделі. Ознайомлення з технологіями об'єктно-орієнтованого проектування перенесено на наступний курс.

Прийняття «імперативного підходу» означає, що студенти отримують менше досвіду в технологіях об'єктно-орієнтованого програмування, ніж при застосуванні моделі «з орієнтацією на об'єктно-орієнтоване програмування». Враховуючи ключову роль об'єктно-орієнтованого програмування у вимогах до навчальних планів, а також труднощі, які виникають під час написання своїх перших об'єктно-орієнтованих програм, стає зрозуміло, що перенесення вивчення цього матеріалу на другий рік навчання є негативною стороною описуваного підходу.

В той же час, студентам, в дійсності, вбачається необхідним розуміння традиційного імперативного стилю програмування, який все ще широко використовується та, при цьому, є невід'ємною частиною будь-якої об'єктно-орієнтованої мови. Тому існують різні судження науковців та практиків щодо визначення того, яка модель має бути представлена першою. Деякі з них стверджують, що студенти-початківці навчання за впровадження імперативного підходу, отримують більше проблем при переході до об'єктно-орієнтованого підходу. Інші, навпаки, свідчать, що студенти, які починали навчання з об'єктно-орієнтованої мови, також будуть мати певні проблеми тоді, коли вони будуть вимушені не використовувати засоби мови, що роблять об'єктно-орієнтоване програмування настільки потужним.

Підхід «з орієнтацією на об'єктно-орієнтоване програмування» (далі – «об'єктний підхід») також фокусується на програмуванні, але, при цьому, на самому початку акцент ставиться на принципи об'єктно-орієнтованого

проектування та програмування. Аналогічно до імперативного підходу, в об'єктному підході відзначаються як двох-, так і трьох семестрові реалізації, що передбачають вивчення тем:

- введення в об'єктно-орієнтоване програмування;
- об'єкти і абстракція даних;
- алгоритми і структури даних;
- об'єктно-орієнтоване програмування;
- об'єктно-орієнтоване проектування і методологія.

В кожному циклі перший курс розпочинається безпосередньо з вивчення поняття об'єктів та наслідування. Далі навчання переходить до представлення більш традиційних структур управляючої логіки, але завжди в контексті загальних понять об'єктного підходу. На подальших курсах передбачається детальніше вивчення матеріалу щодо алгоритмів, основних структур даних і питань програмної інженерії.

Головною перевагою об'єктного підходу до навчання вступних курсів є раннє ознайомлення з об'єктно-орієнтованим програмуванням, яке в останні роки стало надзвичайно важливим як для академічного середовища, так і для промисловості. Наприклад, в грудні 2000 р. Рада коледжів (College Board) анонсувала свої плани з посилення об'єктного підходу в програмі спеціалізованих курсів (Advanced Placement [AP2000]).

В той же час об'єктний підхід також має деякі недоліки, що є спільними з імперативним підходом. Фактично, проблеми імперативного підходу можуть ще гостріше виявитися при об'єктному підході, оскільки більшість об’єктно-орієнтованих мов, зокрема C++, а також певною мірою Java, є набагато складнішими ніж інші мови. Якщо не буде зроблено обмеження складності в матеріалі, який викладається, то деталі окремої мови можуть поглинути увесь навчальний час студентів, які прослуховують вступний курс.

Підхід «з орієнтацією на функціональне програмування» (далі – функціональний підхід), який було вперше використано в Массачусетському технологічному інституті в 1980-х р.р. характеризується використанням на

першому курсі функціональної мови, такої як Scheme. Порівняно з іншими реалізаціями, зазначений підхід має важливі переваги (знання програмування є необхідною умовою для багатьох поглиблених курсів з інформатики; багатьом студентам програмування подобається більше, ніж інші аспекти інформатики). Використання мови, яка не так часто використовується в практиці програмування, дещо зменшує ефект від різниці в підготовці студентів-початківців, частина з яких завжди має певний досвід в програмуванні.

Однак, мінімалістський синтаксис функціональних мов надає можливість викладачам фокусуватися на фундаментальних питаннях. Деякі важливі ідеї, особливо рекурсія, зв'язані структури даних і використання функцій як даних, природним чином вписуються до зазначеного підходу, і можуть бути представлені для вивчення на самому початку курсу навчання.

Проте, при застосуванні функціонального підходу теж спостерігаються певні проблеми.

1. Перша проблема полягає в тому, що може відзначатися деяке скептичне ставлення студентів щодо вивчення мови, яку вони надалі можливо ніколи не будуть використовувати. Для студентів, які обрали інформатику як майбутню спеціальність, зазначені проблеми можна вирішити шляхом демонстрації позитивних сторін мови програмування та завдань, що можуть бути розв’язані за її використання. Студентам, які прослуховують вступний курс з метою визначення своєї основної спеціальності, і особливо студентам, які розглядають цей курс як можливість отримання деяких практичних навичок програмування, функціональні мови даються набагато важче.

2. Друга проблема полягає в тому, що цей підхід, як правило, вимагає від студентів, на ранній стадії навчання, абстрактного мислення, ніж при використанні традиційних мов програмування. Звісно навчання студентів абстрактно мислити є необхідним і важливим, але настільки ранній перехід до абстракції може викликати не сприйняття інформатики студентами, у яких здатність до абстрактного способу мислення ще не сформована.

Для подання матеріалу на першому році навчання, вступний курс, створений за функціональним підходом, має бути продовжений інтенсивним курсом, який охоплює об'єктно-орієнтоване програмування та проектування.

Протягом багатьох років фахівці в галузі інформатики вбачали проблему в тому, що традиційне спрямування на програмування формує обмежений погляд студентів на дисципліну. Однак, інформатика є дисципліною, що постійно розширюється та передбачає, окрім програмування, включення інших видів діяльності. Тому курси, що концентруються лише на програмуванні, не надають можливості студентам використати знання із інших галузей та застосовувати інші способи мислення, що є частиною сучасної інформатики.

Для надання студентам більш цілісного погляду на дисципліну, багато викладачів віддають перевагу підходу, який формулюється як «підхід з максимальним охопленням матеріалу». Впровадження зазначеного підходу на першому курсі навчання передбачає розгляд ширшого спектру тем. Цей підхід був рекомендований звітом CC1991, в якому стверджувалося, що «на перших курсах інформатики повинні викладати не лише програмування, алгоритми і структури даних, але також і матеріал з усіх інших дисциплін», а «математика та інші теоретичні дисципліни мають бути інтегровані в програму навчання інформатики» [2].

Проте, розробка успішної реалізації підходу з максимальним охопленням матеріалу виявилося складним завданням. Аналіз наукових робіт показав, що найчастіше реалізацією зазначеної ідеї було створення вступного оглядового курсу, розрахованого як на студентів, які спеціалізуються в галузі інформатики, так і на всіх інших студентів. Такий курс дає студентам уявлення про цілу низку цікавих і важливих тем. Студенти, яких зацікавила та чи інша галузь, отримують можливість надалі розпочати «звичайний» однорічний вступний цикл. Таким чином, в більшості випадків йдеться про впровадження одного вступного курсу з максимальним охопленням матеріалу як відправної позиції для всіх студентів. Студенти, які пройшли подібний курс навчання, можуть

просуватися далі до будь-якого іншого вступного циклу, де детальніше викладається обраний предмет.

Перевагою використання курсу з максимальним охопленням матеріалу в якості вступного курсу є те, що при такому підході студенти можуть відразу ж оцінити обсяг інформатики, що дозволить їм швидше визначитися, чи хочуть вони більш глибоко вивчати дану галузь. У свою чергу, основним недоліком цього підходу є те, що він додає один курс до профілюючих курсів і затримує на один семестр завершення вступного циклу.

У підході з орієнтацією на алгоритми основні концепції інформатики представляються з використанням псевдокоду замість реальної мови програмування. За рахунок ознайомлення студентів з основними алгоритмічними концепціями та логічними структурами незалежно від мови програмування, цей підхід мінімізує зусилля, що спрямовані на вивчення специфічних синтаксичних конструкцій. Замість цього, від студентів вимагається обґрунтування та роз'яснення алгоритмів, які вони створюють, відлагоджуючи їх на папері. Це дозволяє студентам працювати з широким діапазоном типів даних і логічних структур, без необхідності врахування різних специфічних особливостей, що обов’язково присутні в популярних мовах програмування. Більш того, оскільки студенти звільнені від необхідності виконувати свої програми на комп'ютерах, цей підхід дозволяє студентам значно швидше ознайомитися з різноманітністю таких конструкцій. Як тільки студенти опановують основні алгоритми і типи даних, вони, уже до кінця першого семестру, можуть починати використовувати одну з поширених мов програмування чи, найпізніше, на початку другого семестру. Оскільки до цього моменту студенти вже знайомі з широким спектром структур даних та управляючих структур, традиційне програмування може бути вивчено набагато швидше, а аудиторний час може бути присвячений більш глибшому вивченню практичних питань ефективного програмування та сприянню навичок налагоджування.

Завдяки виключенню з навчальної програми терміну, відведеного на вивчення синтаксису та деталей певного середовища програмування, вступний курс, за підходом з орієнтацією на алгоритми, може включати додаткові теоретичні теми, такі як оцінка ефективності та основи обчислення. Зазначене може бути корисним в двох аспектах:

1. Студенти, які спеціалізуються на інших дисциплінах, отримують деяке уявлення про інформатику як «науку».

2. Студенти, які спеціалізуються на інформатиці, починають знайомитися з відповідними аспектами теорії з перших днів навчання, скорочуючи ризик того, що під кінець навчання вони сприйматимуть теорію як недоречний «додаток» до навчальної програми.

В той же час, підхід з орієнтацією на алгоритми, на думку фахівців має декілька критичних недоліків.

1. По-перше, відсутність можливості отримання першокурсниками практичного досвіду в галузі інформатики. Курси, що зводяться лише до конструювання алгоритмів в псевдокоді, викликають у студентів деяке розчарування. Тому рекомендується об'єднати підхід «з орієнтацією на алгоритми» з лабораторними роботами, що передбачають практичне використання сучасних засобів розробки. Зазначена стратегія дозволяє сформувати практичні навички студентів, які для студентів інших дисциплін можуть бути навіть важливішими, ніж традиційне програмування. При належній синхронізації програми лабораторних робіт з лекційним матеріалом, студенти можуть на особистому досвіді відчути значимість, наприклад, структур даних у контексті роботи з базами даних, структур управління в контексті розробки електронних таблиць і високорівневого проектування в контексті створення Web-сторінок.

2. По-друге, орієнтація на псевдокод також позбавляє студентів від необхідності демонстрації роботи алгоритмів та їх функціональної реалізації. Також студенти не навчаються компілювати програми, хоча формування цих навичок є необхідним для подальшої успішної роботи. Тому студенти повинні

отримувати практичні навички наладки на ранніх етапах свого навчання. Процес налагодження псевдокоду істотно відрізняється від налагодження програми, яка виконується. У першому випадку мова йде про «перевірку за столом» та бесіди про алгоритм; у другому, як правило, про інтерпретацію симптомів і про дії, що виконуються в процесі знаходження програмних помилок. Обидві навички важливі, і досить складно визначити, як застосування підходу з орієнтацією на алгоритми впливає на здатність студентів налагоджувати програми.

3. По-третє, потребуються певні зусилля для проведення оцінювання рівня знань студентів. В дійсності, не зовсім вірно оцінювати якість програм лише на підставі результатів проходження набору контрольних прикладів, однак можливість проведення подібних перевірок, зазвичай, допомагає викладачам швидше знаходити алгоритмічні помилки. В той же час, вбачається значно важчим завданням оцінка псевдокоду на коректність, яке, зазвичай, вимагає залучення великої кількості асистентів.

Отже, на першому курсі навчання починається з обговорення алгоритмів та їх застосувань, далі, до кінця курсу, вивчаються основи об'єктно-орієнтованого програмування. На другому курсі пропонується ретельніше вивчення об'єктно-орієнтованого програмування.

Підхід «з орієнтацією на апаратну частину» (скорочено «апаратний підхід») передбачає вивчення основ інформатики, починаючи з машинного рівня з переходом на більш абстрактніші концепції.

Таким чином, студенти отримують можливість вивчити послідовно інформатику. Програма курсу починається з вивчення перемикальних схем, що потім використовуються для конструювання простих регістрів і арифметичних пристроїв, з яких, у свою чергу, будується проста фон нейманівська машина. Лише після формування у студентів твердого розуміння будови апаратної частини, в програмі відбувається перехід до розгляду програмування мовами високого рівня.

Перший курс циклу передбачає вивчення будови комп'ютера; на другому курсі отримані знання використовуються як базис для розвитку навичок програмування у студентів та введення в об'єктно орієнтовані технології.

Слід відзначити, що впровадження зазначеного підходу використовується для формування детального розуміння студентами процесу обчислення. Однак, він є менш ефективним для заохочення студентів бачити цілісні концепції, що стоять за механізмом реалізації. Апаратний підхід також не досить добре узгоджується із зростаючою централізацією всіх процесів відносно програмного забезпечення та сучасної тенденції вдосконалення віртуальних машин. В той же час, подібний підхід може знайти застосування в програмах з проектування комп'ютерів (computer engineering), в яких передбачається раннє ознайомлення з технічними засобами.

Висновки. Отже, кожен з підходів до вивчення теоретичних основ інформатики, що використовуються у зарубіжних університетах, має певні позитивні сторони та недоліки. Однак, слід наголосити, що вони мають спільну ознаку, яка їх об’єднує – ставлення до теоретичних основ інформатики, що визначаються як один з трьох основних базисів інформатики; основні математичні концепції мають вивчатися студентам на початку їх навчання у ВНЗ, а на старших курсах вони повинні регулярно використовуватися. Для цього визначено галузь знань, яка складається з дискретної математики, обов'язкової для університетської програми.

Загальна світова тенденція до формування професійних якостей майбутнього фахівця вимагає розглядати його як багатогранну особистість, яка має високий ресурсний потенціал для самовдосконалення та творчого становлення. Формування та розвиток інформаційного суспільства ставить нові вимоги до сучасних фахівців як високих професіоналів, лідерів, всебічно розвинених особистостей. Для розкриття та самореалізації особистості необхідно застосовувати такі методи навчання, що спрямовані на максимальне розкриття особистісних здібностей студента у майбутній професійної діяльності.

Література:

1. Denning P. J. Great Principles of Computing / P. J. Denning [Електронний ресурс]. – Режим доступу : http://mrw.interscience.wiley.com/emrw/9780470050118/ ecse/article/ecse548/. – [Article Online Posting Date: March 16, 2009].

2. Computing as a Discipline / P. J. Denning, D. Comer, D. Gries, M. C. Mulder // Communications of the ACM (CACM). – 1989. – № 1. – P. 9-23.

3. Computer Science Curriculum 2008 : An Interim Revision of CS 2001 (December 2008) / [Lillian Cassel, Alan Clements, Gordon Davies et all]. – 2008. – 108 p.

4. Wilf H. S. Algorithms and Complexity / H. S. Wilf. – 2nd edition. – Wellesley : A. K. Peters, Natick, 2002. – 219 p.