Меджитова Л.М.
РВНЗ «Кримський інженерно-педагогічний університет»

Підхід до будування вступного курсу програмування для студентів комп'ютерних спеціальностей

Одним із завдань, в яке «занурюються» студенти першого курсу комп'ютерних спеціальностей, є вивчення основ програмування. Це одна з дисциплін, вивчення якої повинно формувати правильне уявлення про процес побудови розв'язків різноманітних задач за допомогою комп'ютера. У зв'язку з цим важливим питанням є розробка змісту такої дисципліни: від тематики лекцій до прикладів і вправ.

На першому курсі спеціальності «Інформатика» у Кримському інженерно-педагогічному університеті такою дисципліною є «Програмування». Мета навчання дисципліни полягає у підготовці базису для оволодіння спеціальністю інженера-програміста і розвитку умінь пошуку ефективних способів розв'язування задач з наступною їх програмною реалізацією. Для досягнення поставленої мети необхідно сформувати у студентів вміння розробляти алгоритми і записувати їх в різних формах, виділяти і описувати об'єкти завдання та їх взаємозв'язки, читати і розуміти готові алгоритми, записувати програмний код, розуміти семантику основних структур управління в програмному коді, ефективно використовувати інтегроване програмне середовище в ході розв'язування задач, володіти термінологією.

Вступ до програмування, поряд з деякими іншими дисциплінами, утворює платформу для вивчення інших дисциплін спеціальності, що можна побачити на рис. 1.

На рис. 1 зображено перший рівень, представлений трьома дисциплінами: «Вступ до спеціальності», «Програмування» та «Дискретна математика». На цьому рівні базується вивчення таких дисциплін як «Програмне забезпечення SOHO», яке тісно пов'язане з курсами введення в спеціальність і введення в програмування, оскільки знайомить студентів з компонентами, які супроводжують професійне програмне забезпечення. На основі знань і умінь, отриманих в ході вивчення програмування, будується інший важливий курс - об'єктно-орієнтоване програмування. Крім того, знання основ програмування та дискретної математики дозволять студентам успішно засвоїти такі питання сфери інформаційних технологій, як аналіз алгоритмів і робота зі складними структурами даних, а також ознайомитися з основами моделювання комп'ютерної анімації. Таким чином вибудовуються все більш складні рівні освоєння спеціальності.

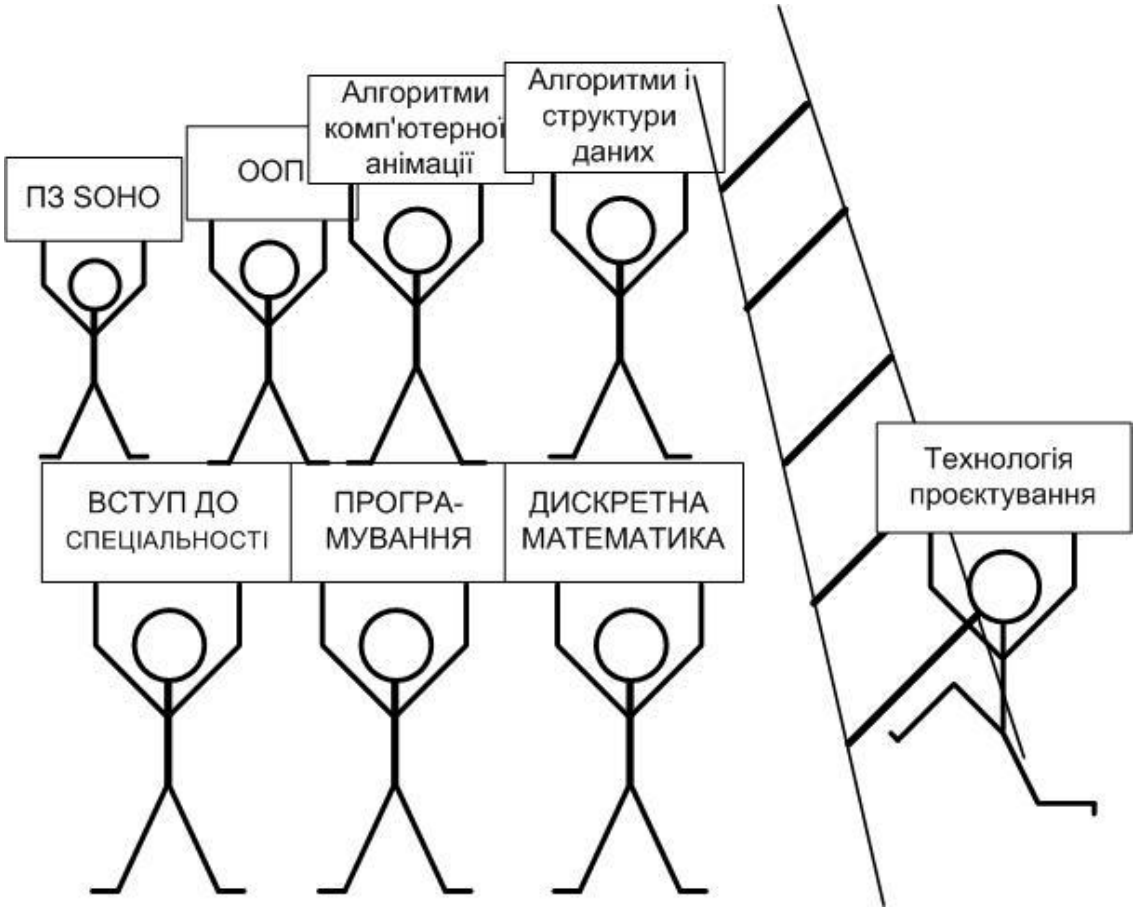

Рис. 1. Зв’язок дисциплін

В [1] виділяється шість підходів до вступу до програмування на молодших курсах: імперативний, об’єктний, функціональний, широкий, алгоритмічний і апаратний. Згідно з певним підходом вибудовується ланцюжок навчальних курсів, результатом вивчення яких є знання основ програмування і володіння різними аспектами використання комп’ютера для розв’язування задач.

У статті «A New Approach to Teaching Programming» [2] описується підхід, заснований на трьох припущеннях, які відображають бачення авторів процесу навчання вступу до програмування. Перше припущення полягає в необхідності інтеграції теорії та практики за допомогою вивчення технічних засобів, замість їх розподілу. Друге припущення авторів полягає в необхідності зміщення уваги від написання коду до проектування розв’язку задачі. Третє припущення – багатогранність процесу розробки програмного забезпечення. Для розв’язування різних задач можуть знадобитися зовсім різні підходи. У зв’язку з цим автори переконані в тому, що програміст повинен володіти різними парадигмами.

Пропозиція авторів статті «Teaching programming using visualization» [3] полягає у використанні прикладів з реального світу для пояснення понять і концепцій. Автори роботи підкреслюють, що програмування по суті своїй ґрунтується на абстракціях і математиці, а це викликає труднощі у студентів молодших курсів. І дійсно, при вивченні програмування перед студентом постає завдання розглянути реальний об’єкт або перебіг процесу (наприклад, перехід людини через дорогу) і описати алгоритм. У зв’язку з цим одним з основних припущень авторів роботи [3] є використання мультимедіа в навчальному процесі, включаючи анімацію, звук і відео, за допомогою яких вивчаються, наприклад, структури управління ходом розв’язування.

У роботі [4] розглядається структура дисципліни, що є введенням до програмування. Необхідні для студентів навички діляться тут на дві категорії: вміння знаходити шляхи розв’язування задач і знання синтаксису і семантики мови програмування. Автори пропонують мови C та C ++, які однак є стартом для подальшого вивчення інших мов програмування.

При визначенні підходу до введення в програмування в більшості джерел орієнтуються на студентів, які мають невеликий досвід програмування або не мають його взагалі. Так, наприклад, побудовані курси «Introduction to Computer Science and Programming», представлений на сайті Массачусетського технологічного інституту [5], і «Introduction to Computer Science: Programming

Methodology», представлений на сайті Стенфордського університету [6]. У зв'язку з цим будь-який із зазначених вище підходів може однаково успішно застосовуватися на молодших курсах.

Дослідження різних джерел і різних підходів до навчання вступу до програмування став першим кроком до визначення змісту дисципліни «Програмування», що вивчається на першому курсі майбутніми інженерами-програмістами.

Актуальність проблеми визначення змісту дисципліни, яка дає перші поняття про розв'язування задач за допомогою комп'ютера, і безліч різних підходів до її вирішення у науковій літературі визначили вибір теми даної статті: «Підхід до будування вступного курсу програмування для студентів комп'ютерних спеціальностей».

Довгий час серед підходів до навчання основ програмування найбільш поширеним залишається імперативний підхід. У цій парадигмі процес розв'язування задачі подається як набір інструкцій, що застосовуються до вхідних даних. Виконання будь-якого завдання за інструкцією є легким для розуміння студентів молодших курсів і зустрічається їм у повсякденному житті. Звідси безліч простих і цікавих прикладів, які можна використовувати при вивченні навчального матеріалу. З іншого боку імперативний підхід є частиною об'єктно-орієнтованого стилю програмування і використовується для опису конкретних методів класу. У зв'язку з цим зручно застосовувати одну із об'єктно-орієнтованих мов програмування, таких як С ++, в якій поєднуються обидва підходи.

У якості середовища програмування використовується Microsoft Visual Studio 2010, яке містить весь необхідний спектр інструментів. Крім того, це середовище відрізняється високим ступенем «юзабіліті» (наприклад, підсвічування коду і Intellysense). Важливою перевагою з точки зору процесу навчання є те, що компанією Microsoft продукт надається студентам і викладачам безкоштовно в рамках програми співробітництва з навчальними закладами Microsoft Developer Network Academic Alliance (MSDN AA). Поширення цілого ряду програмних продуктів Microsoft відбувається за допомогою спеціальної системи e-academy License Management System (ELMS). За допомогою цієї системи, представленої у формі web-сайту, студенти отримують свій примірник ліцензійного програмного продукту. Крім того є безкоштовна лінійка Microsoft Visual Studio 2010 Express, спеціально розроблена для студентів і початкуючих програмістів. З дещо урізаною функціональністю ця версія середовища розробки достатня для того рівня програм, над якими працюють першокурсники комп'ютерних спеціальностей.

Вивчення дисципліни «Програмування» на першому курсі спеціальності «Інформатика» пропонується побудувати на основі рівнів, показаних на рис. 2 і зображених у вигляді спіралі.

Кожного разу на наступному рівні висвітлюються питання, розуміння яких засноване на знаннях, сформованих раніше. На рис. 1 зображується розподіл тим дисципліни в ході її вивчення. Відправною точкою традиційно є алгоритм. За допомогою алгоритмів можна найбільш природно висловити ситуації і завдання, що зустрічаються першокурсникам у повсякденному житті: перейти через дорогу, поповнити телефонний рахунок, зробити покупки в магазині і так далі. Тобто почати навчання програмування можна з елементарних прикладів і опису їх у словесній формі.

На цьому етапі в якості одного з прикладів студентам пропонується записати алгоритм простенької гри «Лялька». Суть її полягає в тому, що є загадане слово і є кілька спроб відгадати його за літерами. Якщо літера вгадана, то вона відкривається скрізь, де зустрічається в слові. В іншому випадку на шибениці малюється чергова частина ляльки. Якщо лялька намальована, то гравець програв. Постановка завдання уточнюється за допомогою словесного алгоритму, який може бути записаний так:

1. Ввести букву;
2. Якщо слово містить букву, то відкрити вгадану букву, інакше перейти до кроку 4;
3. Якщо слово вгадано, то перемога, інакше перейти до кроку 1;
4. Намалювати чергову частину ляльки;
5. Якщо лялька намальована, то програш, інакше перейти до кроку 1.

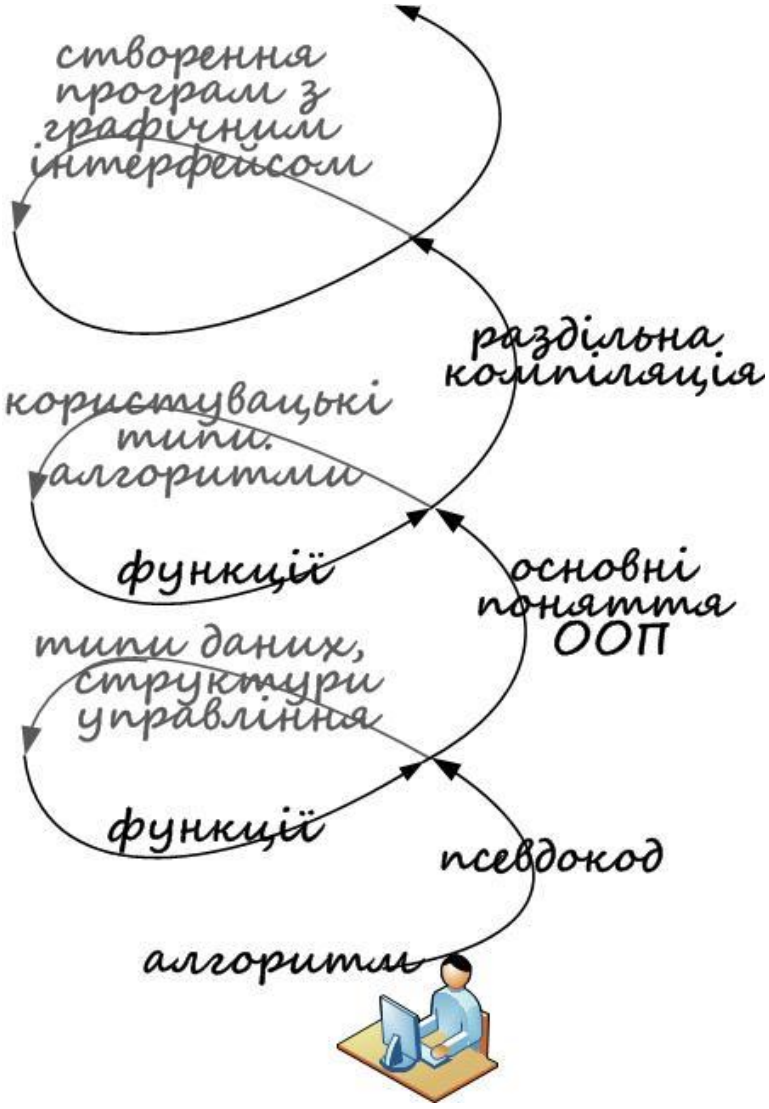

Рис. 2. Тематика лекцій з дисципліни «Програмування»

Тепер можна описати розв'язок за допомогою псевдокоду. Псевдокод більш близький до мови програмування, а його конструкції більш точні й однозначні, ніж фрази природною мовою. Запис алгоритму виконання цього завдання на псевдокоді може виглядати наступним чином:

```
1. while true
2.         ▷ ввести значення змінної letter
3.         if CheckLetter(letter) then PrintLetter(letter)
4.                                 if WordRevealed() then
5.                                                 ▷ вивести «Перемога»
6.                                                 quit
7.                         else Draw()
8.                                 if doll() then
9.                                                 ▷ вивести «Програш»
10.                                                quit
```

На початковій стадії вивчення мови програмування С ++ студенти першого курсу знайомляться з основними типами даних і структурами управління, а потім, як показано на рис.1, знову повертаються до типів даних, але вже в їх більш складній формі. Тут студенти знайомляться з деякими структурами даних (масиви, списки) і з алгоритмами, що застосовуються до них. У зв'язку з цим знову розглядаються структури управління, але вже в рамках роботи з новими алгоритмами.

Закріплення знань студентів та визначення їхніх навчальних досягнень за вивченими питаннями здійснюється в ході виконання лабораторних робіт. Важливо відзначити, що при цьому використовується диференційований підхід, який полягає в підготовці завдань різного рівня складності. На перших заняттях при проведенні оцінювальної роботи студентам надається

можливість самостійно вибрати бажаний рівень складності завдання. Однак згодом на основі результатів попередніх робіт викладач допомагає у виборі посильного завдання. При необхідності рівень складності завдання для конкретного студента може бути дещо знижений або навпаки підвищений. Виконання кожної оцінювальної роботи і подання її викладачеві відбувається в строго встановлені терміни. При цьому переслідується така мета: студент не повинен відкладати виконання завдання до такого часу, коли він вже повинен буде працювати над наступним.

Крім лабораторних робіт, які полягають у написанні алгоритму і програмного коду, студентам пропонується виконати ряд тестів. Всі студенти виконують однакові завдання одночасно і в умовах обмеженого часу. Тест складається з двох частин. При виконанні завдань першої частини тесту передбачається вибір правильних відповідей з кількох запропонованих. Такі завдання зручні з точки зору здійснення вимірювання навчальних досягнень для всіх студентів курсу одночасно. Для успішного виконання тесту від студентів вимагається знання синтаксису й семантики конструкцій мови С ++. Як приклад можна навести такі завдання:

Які ідентифікатори з наведених неприпустимі в С ++?

(a) int double = 3.456;
(b) char 1_or_2 = '1';
(c) int number = 3;
(d) double real_num = 4.123;

Яке значення буде надано змінній val в результаті виконання даного фрагмента коду?

```
int val=3, i=7;
val=val*i+(++i);
```

(a) 32;
(b) 28;
(c) 29;

При виконанні завдань другої частини тесту передбачаються відкриті відповіді у вигляді програмного коду. При цьому студентові пропонується алгоритм виконання завдання, у якому пропущені деякі важливі фрагменти. Студенту необхідно дописати цей код. Наприклад, це може бути таке завдання:

Заданий фрагмент програми. Доповніть пропущені місця в наведеному коді так, щоб на екран були виведені цифри від 1 до 5 у такому вигляді:

```
1       1       1       1       1
2       2       2       2
3       3       3
4       4
5
int i, j, k;
for (i = ...; i <=...; i++)
{
        k = ...;
        for (j = k; j>=...; j--)
                cout<<...<< '\t';
        ...
}
```

Вивчення поняття функції також відбувається впродовж кількох етапів. Все починається з першого знайомства зі структурою опису програми мовою С ++. Тут вперше згадується таке поняття, як контейнер, що містить логічно завершений і незалежний алгоритм з деякими вхідними і вихідними даними (або ж їх відсутністю). На цьому етапі відбувається знайомство з головною функцією, яка містить алгоритм розв'язування всієї задачі.

На наступному рівні студенти вчаться визначати довільні функції, що містять алгоритм розв'язання однієї підзадачі, виділяти її аргументи і вихідне значення. В якості прикладів розглядаються прості завдання, в яких не потребується повернення кількох значень або яких-небудь змін вхідних параметрів. Тут важливо навчитися виокремлювати підзадачі і засвоїти синтаксис. Згодом з таких іменованих фрагментів нескладно буде «зібрати» розв'язок задачі. У рамках же більш високого рівня, маючи досвід роботи з адресами, студенти знову розглядають функції, передавання параметрів за адресою, ітеративні і рекурсивні алгоритми.

У ході навчання студенти розглядають різні сторони процесу розв'язування задачі: уточнюють постановку задачі за допомогою словесного опису, уточнюють спосіб розв'язування

задачі з допомогою запису алгоритму на псевдокоді, описують відповідний алгоритм мовою програмування С ++, розглядають різні нюанси процесу компіляції програми. Окрема лекція і спеціально дібрані вправи призначені для навчання налагодження програм (передбаченими для цього засобами) та їх тестування, які є важливими етапами в процесі розв'язування задачі за допомогою комп'ютера. Тут велику частину лекційного матеріалу складає презентація інструментальних засобів для здійснення цієї частини роботи програміста. З цієї точки зору можна порівняти різні середовища програмування на предмет зручності їх використання (usability). Як вправи до цієї теми пропонується програмний код, скажімо програми-калькулятора, в якому навмисно допущені логічні помилки. Для пошуку цих помилок студентами використовується налагоджувач, що є в розпорядженні MS Visual Studio 2010. Серед студентів першого курсу спеціальності «Інформатика» робота над цією вправою була організована в командному режимі. Крім пошуку логічних помилок в програмі, до кожної команди пропонувалося написати сценарії тесту та здійснити їх, фіксуючи результати.

На одному з рівнів вивчення дисципліни «Програмування» студенти знайомляться з основними поняттями об'єктно-орієнтованого підходу. Тут знову можна звернутися до прикладу гри «Лялька» і розглянути її в рамках такого підходу. Студенти вчаться виокремлювати об'єкти та визначати їх взаємозв'язки. Крім того, так їм можна продемонструвати можливість використання двох різних підходів до виконання конкретного завдання.

Закінчується цей навчальний курс створенням програми з графічним інтерфейсом, для чого було необхідно попередньо розглянути поняття класів і їх ієрархії. На цьому етапі результатом роботи студента є програма, схожа на ті, якими вони вже звикли користуватися, що є доброю мотивацією для першокурсника. В якості творчого завдання можна запропонувати студентам нарешті реалізувати гру «Лялька» у формі прикладної програми з графічним інтерфейсом і протестувати її.

Таким чином, вступний курс з програмування для студентів комп'ютерних спеціальностей можна побудувати за принципом дотримання рівнів, використовуючи при цьому імперативну і об'єктно-орієнтовану парадигму. При такому підході на першому етапі навчання студенти за допомогою простих і знайомих прикладів зможуть легко адаптуватися, а до кінцевого етапу накопичити «багаж» знань і навичок, які будуть необхідні для освоєння більш складних дисциплін спеціальності. При цьому доцільно обрати мову програмування С ++ і середовище Microsoft Visual Studio 2010 як інструментарій для реалізації запропонованого підходу до навчання основ програмування.

Література

1. Сейдаметова З.С. Модели обучения основам программирования на младших курсах компьютерных специальностей университетов / З.С. Сейдаметова, Л.М. Меджитова /

2. Jackson D. A New Approach to Teaching Programming [Электронный ресурс] / D. Jackson, R. Miller / – Режим доступа: http://people.csail.mit.edu/dnj/articles/teaching-6005.pdf

3. Rudder A. Teaching programming using visualization [Электронный ресурс] / A. Rudder, M. Bernard, S. Mohammed / – Режим доступа: http://sta.uwi.edu/staff/mbernard/research_ files/WBE2007.pdf

4. Musa Z. Alternative approach for teaching programming subjects [Электронный ресурс] / Z. Musa, R.A. Arshah, R. Mohammed, R.A. Bakar, A. Abdullah / – Режим доступа: http://umpir.ump.edu.my/911/1/4_a_Tuty.pdf

5. Introduction to Computer Science and Programming [Электронный ресурс]. – Режим доступа: http://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-00-introduction-to-computer-science-and-programming-fall-2008/index.htm

6. Introduction to Computer Science: Programming Methodology [Электронный ресурс]. – Режим доступа: http://see.stanford.edu/see/courseinfo.aspx?coll=824a47e1-135f-4508-a5aa-866adcae1111