ISSN 2224-087X. Електроніка та інформаційні технології. 2014. Випуск 4. С. 148–155
Electronics and information technologies. 2014. Issue 4. P. 148–155

ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ В НАВЧАЛЬНОМУ ПРОЦЕСІ

УДК 681.3.06

ПРОГРАМНА ПІДТРИМКА ПРАКТИЧНИХ ЗАНЯТЬ З ПРОГРАМУВАННЯ

В. Бондарєв

Харківський національний університет радіоелектроніки,
пр. Леніна, 14, 61060, Харків, Україна

Запропоновано програмне та методичне забезпечення для проведення практичних занять з програмування. Методичну основу пропозиції становить збірник задач і автоматична перевірка програм, написаних студентами. Програмне забезпечення складається з клієнт-серверного застосування, що працює в локальній мережі. Як засвідчив досвід, пропонований підхід підвищує зацікавленість студентів, робить їхню роботу інтенсивнішою і водночас розвантажує викладача від рутини й дає йому змогу індивідуалізувати процес навчання.

Ключові слова: програмування, навчання, автоматична перевірка програм.

Відомо, що програмування потребує вправ, і не можна навчитися програмувати, тільки слухаючи лекції і читаючи книги. Практичні заняття з вирішенням завдань і розбиранням вирішень відіграють важливу роль у навчальному процесі. Однак групові заняття в традиційній формі, коли одна людина вирішує завдання біля дошки, а решта 15–20 студентів за цим спостерігають, не дають великого ефекту. Вони марні і для студентів з рівнем знань нижче середнього, які завдання не розуміють, і для студентів з рівнем вище середнього, які на такому занятті просто нудьгують. Було б добре давати студентам різні завдання – складніші для відмінників, легші для тих, які відстають, однак обмеженням стають можливості викладача, йому одному за всім не встежити. Та вихід є, і він полягає в автоматичній перевірці студентських вирішень.

Перевірку вирішень і низку інших функцій виконує клієнт-серверна комп'ютерна програма. Програма працює в бездротовій локальній мережі, утвореній з портативного комп'ютера викладача і ноутбуків або планшетів студентів за допомогою маршрутизатора Wi-Fi. Таку мережу викладач може створити "у декілька кліків" безпосередньо перед початком заняття.

Наголосимо, що є програмні засоби для віддаленої компіляції та виконання програмного коду [1–3] і масиви завдань, призначені для вирішення [4, 5]. Тут їх детально не розбиратимемо, проте з упевненістю зазначимо, що жодне з них не забезпечує необхідної оперативності й гнучкості для вирішення поставленого завдання – адаптації практичних занять до індивідуального рівня підготовки студентів.

У пропонованого програмного засобу два види користувачів – студенти і викладач, який проводить заняття. З боку викладача сценарій застосування програми виглядає так. Перед початком заняття викладач вносить до бази даних програми обліковий склад гру-

пи і заздалегідь підготовлені завдання. Зрозуміло, і список студентів, і завдання можна використовувати більше ніж для одного заняття.

На початку заняття викладач за допомогою програми видає кожному студенту одне або кілька завдань для вирішення. Студенти починають працювати, а викладач за допомогою програми спостерігає за вирішенням завдань і втручається в цей процес з власної ініціативи, а радше на прохання студента. Контроль правильності вирішення програма виконує автоматично.

Викладач додає нові завдання у міру їх вирішення і може видаляти або замінювати завдання, якщо початковий вибір був невдалим.

Студент відкриває вікно веб-браузера, у якому бачить умови завдань, запропонованих для вирішення. Він обирає одне з них, уводить код вирішення у відповідне поле веб-форми і надсилає код на сервер для автоматичної перевірки. На сервері код вміщається в контекст завдання, компілюється разом з контекстом, і якщо компіляція вдалася, виконується. Залежно від результату студент отримує повідомлення про помилку компіляції або відповідь сервера, що вирішення пройшло або не пройшло перевірки на правильність. Цикл виправлення коду студентом і автоматичної перевірки на сервері може повторюватися скільки завгодно разів.

Крім того, студент може отримувати автоматичні підказки з виконання завдання, якщо вони були закладені в умову викладачем. Робота студента потребує тільки стандартного веб-браузера і приймача Wi-Fi, її можна виконувати за допомогою планшета або мобільного телефону, хоча зі справжньою клавіатурою набагато зручніше.

Користувацький інтерфейс викладача становлять два вікна – головне і вікно стану завдання (рис. 1).

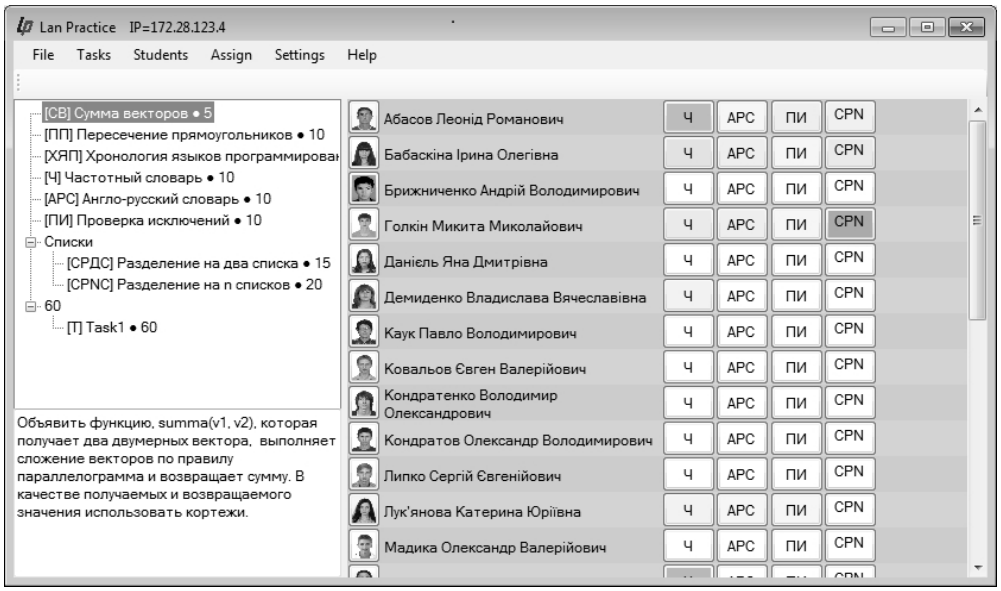

Рис. 1. Головне вікно програми.

Головне вікно розділене на дві частини. Ліву частину займає панель завдань, праву – список студентів.

На панелі завдань розташований задачник у формі дерева, що нагадує дерево файлової системи. Завдання можна вибирати, створювати, видаляти і змінювати, тобто виконувати стандартний перелік операцій маніпулювання даними.

Список студентів є послідовністю інформаційних панелей. Панель починається з фотографії, імені та прізвища студента. Всі видані конкретному студенту завдання показані у вигляді кнопок на інформаційній панелі праворуч від прізвища студента. Ще не вирішені завдання видно як незабарвлені кнопки, кнопки вирішених завдань зелені. Щоб видати завдання, потрібно вибрати його в панелі завдань у лівій частині головного вікна і натиснути на кнопку біля прізвища студента. Отже, одне завдання можна видавати кільком студентам.

У разі натискання на кнопку відкривається вікно стану завдання (рис. 2).

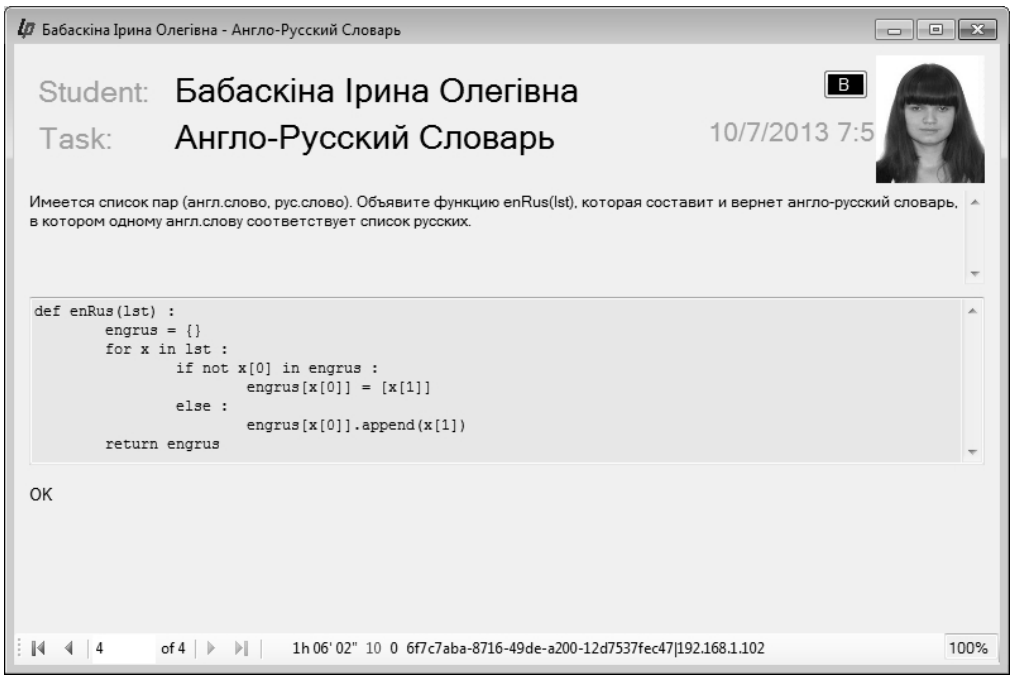

Рис. 2. Вікно стану завдання.

У вікні стану задачі видно особисті дані студента, назву й умову завдання, хронологію процесу його вирішення. Хронологія – це горизонтальна смуга внизу вікна з точками на ній. Точки відображають моменти часу, коли студент надсилав вирішення на перевірку.

Викладач, обравши точку на смузі, бачить код, надісланий студентом у відповідний момент часу, і результат перевірки цього коду на сервері. Так викладач може простежити

весь процес вирішення завдання і визначити, яких знань не вистачало студенту для його успішного виконання.

Студент, підключившись до локальної мережі і набравши в браузері відповідну адресу, бачить обліковий склад своєї групи. Він вибирає в списку своє ім'я, і перед ним відкривається вікно вирішення завдань (рис. 3).

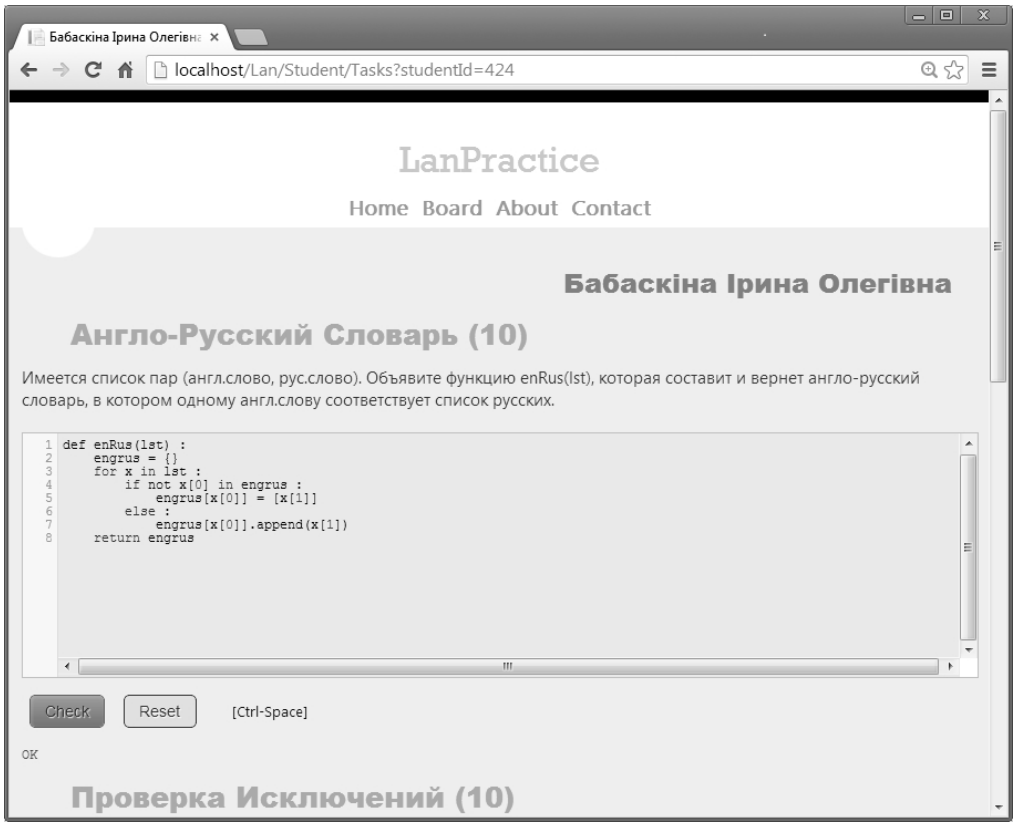

Рис. 3. Вікно вирішення завдань.

Вікно вирішення завдань містить усі завдання, задані студенту викладачем. Кожне завдання має назву, умову і поле для введення програмного коду. Студент обирає будь-яке зі ще не вирішених і вводить код вирішення в поле введення. Коли студент хоче перевірити своє вирішення на сервері, він натискає на кнопку "Перевірити" і невдовзі бачить відповідь сервера. Кнопка "Скидання" повертає поле введення в початковий стан.

Якщо завдання має підказки, то поряд з кнопками "Перевірка" та "Скидання" є кнопки підказок. На кожній кнопці написано "Підказка" і число, що визначає ціну підказки.

Особливість цієї системи полягає в тому, що в пропонованих студентам завданнях не потрібно писати закінчені програми з уведенням даних, друкуванням результатів і точкою входу, а треба розробити фрагмент коду, який повинен безпомилково працювати в певному контексті. Опис контексту та вимог до коду і становить умову завдання. Такий

підхід, по-перше, дає змогу формулювати різні завдання і, по-друге, позбавляє студента від написання рутинного коду, економлячи час заняття. Наведемо кілька прикладів завдань, які треба вирішувати мовою C#.

Привласнення. Оголосити цілу змінну a і зробити її такою, що дорівнює 5. Оголосити цілу змінну b і присвоїти їй 1. Збільшити обидві змінні на 1, скласти їх і суму помістити в змінну c.

Обмін значень змінних. Є дві цілі змінні a і b (тобто вони вже оголошені й отримали якісь значення). Написати код, який поміняє їхні значення місцями.

Сторони трикутника. Є три дійсні змінні: a, b, c. Перевірити, чи можуть їхні значення бути довжинами сторін трикутника. Якщо можуть, то занести значення true в логічну змінну t. Якщо не можуть, то занести туди ж значення false.

Факторіал. Оголосити статичний метод Fact (), який отримує невід'ємне ціле число і повертає факторіал цього числа. Зауважити, що 0! = 1.

Заповнення масиву. Оголосити статичний метод Fill(), який отримує ціле число n і повертає цілий масив з n елементів, заповнений числами 0, 1, 2,..., n - 1.

Окружність з конструкторами. Оголосити клас Circle – коло з відкритими властивостями подвійної точності X і Y (координати центра) та R (радіус). У класі повинен бути відкритий конструктор з трьома параметрами (ордината, абсциса і радіус) і конструктор за замовчуванням, який надає всім властивостям нульових значень.

Нульові елементи. Оголосити клас MyList, який успадковує бібліотечний клас List<T> і має відкриту властивість ZeroCount тільки для читання, яка показує, скільки в колекції нульових елементів. Нульовим уважають елемент, значення якого дорівнює default(T).

Узагальнений делегат. Оголосити узагальнений делегат з ім'ям Del. Делегат повинен посилатися на функції, які мають два однотипні параметри і повертають логічне значення.

Ітератор для ферзя. Оголосити статичний ітератор Queen(), який отримує місце розташування ферзя на шаховій дошці і в довільному порядку перераховує всі клітини, що перебувають під боєм цього ферзя. Клітина, на якій стоїть ферзь, під боєм не перебуває. Клітина задана рядком з двох символів, наприклад, "A1", "E2", "B8".

Підготовлена задача має чотири частини: назву, умову, підказки і контекст.

Назва складається зі шляху та імені завдання. Шлях визначає місце завдання в задачнику, бо сам задачник має ієрархічну структуру. Ім'я завдання має бути унікальним для завдань, що мають однаковий шлях.

Умова завдання – це довільний текст мовою, зрозумілою студенту.

Підказки повинні допомогти студенту виконати завдання, якщо в нього виникають труднощі. Підказок може бути декілька, однак не менше однієї, тому що перша підказка є такою лише за формою. Зміст "першої підказки" буде виведено у вікні коду в разі отримання завдання студентом. Цілком нормально, якщо першою підказкою буде порожній рядок.

Контекст – це програма, що компілюється. Контекст включає в себе і код авторського вирішення задачі. У контексті цей код відокремлений "дужками" – рядками //BEGIN і //END для мов C++ і C# і рядками #//BEGIN і #//END для мови Пітон. Такі обмежувачі є прозорими для компіляторів і це спрощує викладачу підготовку авторського вирішення.

У базі даних системи завдання зберігаються у форматі XML. Зовнішнє подання завдання може збігатися з внутрішнім, тобто бути тим же XML, а може мати альтернативну форму, так званий простий формат.

У простому форматі всі частини завдання розташовані послідовно і розділені рядком, що складається не менше ніж з трьох дефісів. Рядок з дефісів може завершуватися інформаційним маркером. Зокрема, роздільник, що передує умові, завершується назвою мови програмування – cs, cpp або py, роздільник, що передує підказці, завершується числом – ціною завдання після підказки. Логічно, якщо з кожною наступною підказкою це число буде меншим від попереднього, однак це не обов'язково. Нижче наведено приклад завдання в простому форматі.

```
присвоєння/Обмін Значень Змінних
---------------------- CS
Є дві цілі змінні a і b (вони вже оголошені й отримали якісь
значення). Напишіть код, який поміняє їх значення місцями.
------------------- 10
// int a = 3, b = 5; - не треба розкоментовувати цей рядок
------------------- 9
// int a = 3, b = 5; - не треба розкоментовувати цей рядок
// Скористайтеся третьою змінною
int t = a;
---------------------
using System;
public class Program
{
    public static int Main()
    {
        int a = 3, b = 5;
//BEGIN
        int t = a;
        a = b;
        b = t;
//END
        if (a == 5 && b == 3)
            return 0;
        return 1;
    }
}
```

З отриманого досвіду підготовка одного завдання потребує від 5 до 20 хв, однак її роблять один раз, після чого завдання можна використовувати багаторазово. Корисним може виявитися спільний масив підготовлених завдань, з якого можуть брати завдання всі викладачі.

Програмний проект складається з двох додатків: мережевого, LanPractice, і настільного, LanTutor,та двох бібліотек: Compilers и Repository.

Веб-застосування LanPractice забезпечує роботу студентів. За його допомогою студент отримує завдання, пише код вирішення, надсилає його на перевірку і бачить результати цієї перевірки. Настільне застосування LanTutor потрібно для роботи викладача.

Воно дає змогу керувати завданнями, списком студентів, видавати завдання студентам і спостерігати за ходом їхнього виконання.

Обидві програми пов'язані через загальну базу даних, доступ до якої забезпечує бібліотека Repository. Інша бібліотека, Compilers, дає змогу оперативно компілювати студентський код і виконувати його в тестовому оточенні розв'язуваної задачі. Цією бібліотекою також користуються обидві програми. Загальна схема проекту показана на рис. 4.

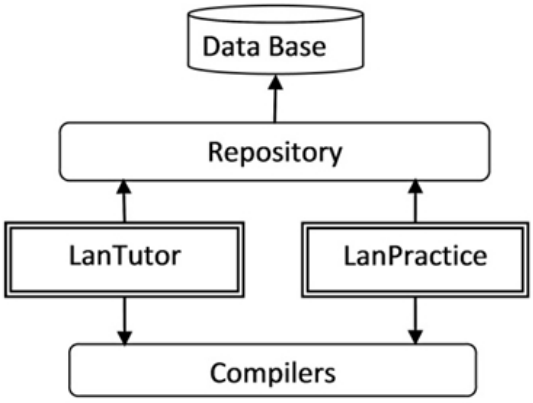

Рис. 4. Схема програми. Стрілки на схемі означають залежності між її елементами.

Самостійне вирішення завдань – єдиний спосіб навчитися програмувати. Пропонований підхід суттєво збільшує кількість завдань, що їх виконують студенти на практичних заняттях, і робить навчання більш інтенсивним.

У поточній версії програми студенти можуть виконувати завдання мовами C#, C++ і Python, однак система є відкритою щодо мов програмування, і перелік дозволених мов можна легко розширювати.

Важливою частиною проекту є задачник – збірник завдань, спеціально підготовлених до використання в системі. Кількісний і якісний склад завдань буде ліпшим, якщо у створенні задачника візьмуть участь усі викладачі, що користуються системою.

Запропоноване технічне вирішення є портативним варіантом. Після того, як корисність підходу буде визнана в навчальному закладі, можна перейти до вирішення, централізованого на рівні закладу, що позбавить викладача необхідності самому створювати локальну мережу і мати на своєму комп'ютері встановлене програмне забезпечення.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. Pascal ABC WEB. Современное программирование на языке Pascal / [Электронный ресурс]. – Режим доступа: http://pascalabc.net/.
2. TopCoder – корпорація, яка проводить змагання зі спортивного програмування / [Електронний ресурс]. – Режим доступу: http://www.topcoder.com/.
3. ejudge – система для проведення різних заходів, в яких необхідна автоматична перевірка програм / [Електронний ресурс]. – Режим доступу: http://ejudge.ru/.
4. *Абрамян М. Э.* Веб-среда разработки и обучения / М. Э. Абрамян, С. С. Михалкович // Открытые системы. СУБД. – 2012. – № 10. – С. 56–59.
5. Programming Taskbook. Электронный задачник по программированию / [Електронний ресурс]. – Режим доступу: http://ptaskbook.com/ru/

Стаття: надійшла до редакції 16.10.2013,
доопрацьована 10.11.2013,
прийнята до друку 27.11.2013.

SOFTWARE SUPPORT OF PRACTICAL LESSONS IN PROGRAMMING

V. Bondariev

Kharkov National University of Radio Electronics,
Ukraine, Kharkov, 61166, Lenin ave., 14

The paper proposed the software and methodology support for practical lessons on programming. Methodological basis of proposals is a collection of tasks and automatic verification of programs written by students. The software consists of a client-server application that runs on the LAN. As experience has shown the proposed approach increases student interest and makes them work more intensively and at the same time relieves the teacher from routine and allows him to individualize the learning process.

Key words: programming , training, automated test applications.

ПРОГРАММНАЯ ПОДДЕРЖКА ПРАКТИЧЕСКИХ ЗАНЯТИЙ ПО ПРОГРАММИРОВАНИЮ

В. Бондарев

Харьковский национальный университет радиоэлектроники,
пр. Ленина, 14, 61060, Харьков, Украина

Предложено программное и методическое обеспечение для проведения практических занятий по программированию. Методическую основу предложения составляет сборник задач и автоматическая проверка программ, написанных студентами. Программное обеспечение состоит из клиент-серверного приложения, работающего в локальной сети. Как показал опыт, предлагаемый подход повышает заинтересованность студентов, делает их работу более интенсивной и в то самое время разгружает преподавателя от рутины и дает ему возможность индивидуализировать процесс обучения.

Ключевые слова: программирование, обучение, автоматическая проверка программ.