Мережева підтримка навчання програмуванню

Бондарев В.М.[1], Черепанова Ю.Ю.[2]

[2]Проф., к.т.н., кафедри програмної інженерії, Харківський національний університет радіоелектроніки
пр. Леніна, 14, м. Харків, Україна, vladmikhbond@gmail.com
[2]Ст. викл. кафедри програмної інженерії, Харківський національний університет радіоелектроніки
пр. Леніна, 14, м. Харків, Україна, yulya.chery@gmail.com

***Анотація* — З метою поліпшення викладання програмування запропонована програмна система автоматичної перевірки навчальних завдань. У центрі системи знаходиться веб-служба, яка надає умови завдань і здійснює перевірку їх рішень. На периферії системи розташовані додатки: для самостійної роботи, проведення іспитів, підтримки електронних лекцій. Запропонована система знаходиться в експлуатації і добре себе зарекомендувала.**

Ключові слова — навчання програмуванню, автоматична перевірка рішень, інтенсивне навчання.

Network support teaching programming

Bondarev V.M.[1], Cherepanova Y.Y.[2]

[1] Dr. of Science, Professor, Department of Software Engineering, Kharkiv National University of Radio Electronics
Lenin Ave, 14, Kharkov, Ukraine, vladmikhbond@gmail.com
[2] Senior teacher, Department of Software Engineering, Kharkiv National University of Radio Electronics
Lenin Ave, 14, Kharkov, Ukraine, yulya.chery@gmail.com

***Abstract* — In order to improve teaching of programming a software system of automatic testing for school assignment is offered. At the heart of the system is a web service that provides the conditions of tasks, and checks their solutions. On the periphery of the system such applications are located: for self education, the examination, electronic support lectures. The proposed system is in operation and works well.**

Keywords — learning programming, automatic test solutions, intensive training.

I. Мета розробки

Основною метою навчання програмуванню є вміння писати програми. Знання мов, алгоритмів, бібліотек або здатність правильно поводитися на співбесіді теж потрібні, але це скоріше засоби, а не мета. Найкоротшим, якщо не єдиним, шляхом до мети є вправи, тому забезпечити студента задачами з програмування і, що теж важливо, мотивувати до їх вирішення є перший обов'язок викладача.

Що ж заважає викладачеві в повній мірі цей обов'язок виконати? Питання майже риторичне, звичайно ж брак коштів, але – не тільки. Ще нестача часу і сил, адже задачу потрібно не лише поставити, але й перевірити її вирішення. Якщо не перевіряти або перевіряти недбало, то більшість студентів і вирішувати задачу не забажає.

Подолати проблему може автоматична перевірка студентських рішень, якій і присвячено це повідомлення.

II. Завдання з програмування

Спочатку трохи про задачі. Задача це не обов'язково програма, яка щось вводить, обчислює і виводить. Навпаки, робота програміста полягає в написанні фрагментів коду, які увійдуть у вже наявну кодову базу. Такими ж повинні бути і навчальні завдання. Основною формою задачі може бути визначення функції, що задовольняє певним умовам, але це може бути і визначення класу, методу, події чи іншого програмного об'єкта із заданими властивостями. Ще це може бути будь-якою частиною усього переліченого, наприклад, є код з пропуском або помилкою, потрібно цей пропуск заповнити або помилку виправити.

Задачу готує викладач. Він повинен чітко сформулювати умову, написати програму, яка компілюється, і виділити в ній частину, яку назвемо авторським рішенням завдання. Код програми за межами виділеної частини повинен забезпечити достатню з погляду викладача перевірку рішення. Здавалося б, викладачеві не обов'язково надавати свою версію рішення, але це необхідно, щоб упевнитися у коректності підготовленої задачі.

Студент отримує умову задачі і пише свою версію рішення. Система автоматичної перевірки підставляє рішення студента замість авторського, компілює і виконує програму. Якщо програма завершується успішно, рішення вважається правильним. В іншому випадку студенту надсилається повідомлення, що рішення відкинуто. У повідомленні може бути вказана причина, по якій рішення не пройшло перевірку. Якщо рішення не компілюється, студент отримає звіт про помилки компіляції. У будь-якому випадку студент може виправити рішення і повторити спробу, аж до закінчення часу, відведеного на задачу.

Для ілюстрації наведемо приклад задачі.

Умова: "Визначте функцію gcd (a, b), яка повертає найбільший спільний дільник двох цілих додатних чисел. Мова програмування JavaScript".

Програма:

```
//BEGIN
function gcd(m, r) {
    if (m < r) {
        var t = m; m = r; r = t;
    }
    while (r > 0) {
       t = r;
       r = m % r
       m = t;
    }
    return m;
}
//END
if (gcd(12, 8) != 4)throw
   new Error("Wrong.gcd(12,8) != 4");
if (gcd(6, 10) != 2)throw
   new Error("Wrong.gcd(6,10) != 2");
if (gcd(3, 3) != 3) throw
new Error("Wrong.gcd(3,3) != 3");
throw new Error ("OK");
```

У програмі авторське рішення виділено "дужками" //BEGIN ... //END.

Крім умови і програми, завдання має необов'язкові елементи: підказки і прототип рішення. Підказки видаються на вимогу студента і стосуються алгоритму рішення або техніки програмування. У даному прикладі підказка може бути такою:

```
gcd(a,0) = a;  gcd(a,b) = gcd(b, a%b)
```

Прототип рішення пропонується студенту разом з умовою задачі і в даному прикладі може бути таким:

```
function gcd (m, r) { }
```

Час, потрібний на підготовку одного завдання коливається від 10 хвилин до години, залежно від його складності.

III. Реалізація системи

В основі системи лежить веб-служба TSS (Task Set Service), яка надає умови завдань і забезпечує перевірку їх рішень. На рис. 1 наведено схему системи.

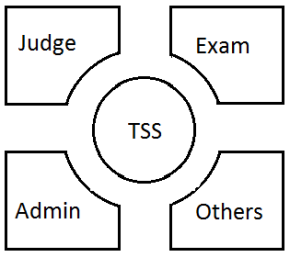

Рисунок 1 – Схема системи

Навколо цієї служби будуються мережеві додатки: для самостійної роботи (Judge), для проведення контрольних робіт (Exam), для ведення задачника (Admin), для проведення змагань та інші. Декілька слів про важливіші додатки.

Додаток для самостійного вирішення задач, Judge, має дві категорії зареєстрованих користувачів, назвемо їх викладачі та студенти. Викладачі можуть створювати серії задач із тих, що надає служба. Студенти можуть відправляти рішення на перевірку і вести облік завдань, які вирішили особисто (зберігати умови і свої власні рішення). Викладачі можуть відстежувати активність студентів стосовно будь-якої задачі і переглядати надіслані рішення.

У додатку для оцінювання знань, Exam, зареєстрованими користувачами є лише викладачі. Викладач створює "іспит", вибираючи студентів і задачу, винесену на іспит, та встановлює тривалість іспиту. Студенти вирішують запропоновану задачу протягом встановленого часу. У ході іспиту і після нього викладач може спостерігати стан студента (вирішує завдання, вже вирішив або ще не почав роботу) і передивлятись надіслані рішення, в тому числі і невдалі. Наприкінці іспиту викладач отримує звіт. Вся накопичена в ході іспиту інформація зберігається на сервері і згодом може бути використана для аналізу.

Додаток для ведення задачника (Admin) дозволяє аутентифікованим викладачам додавати, видаляти і змінювати завдання. Завдання визначає мову, якою слід писати рішення. Наразі є можливість створювати завдання мовами C#, C++, Python, Haskell та JavaScript. Перелік мов є відкритим і легко може бути розширений за потребою.

Описана система використовується на кафедрі програмної інженерії ХНУРЕ при викладанні дисциплін "Основи програмування", "Об'єктно-орієнтоване програмування" та "Функціональне програмування".

Висновки

З впровадженням системи практичні заняття з програмування якісно змінилися, якщо раніше працювала, від сили, половина групи, то тепер в них залучені всі студенти.

Змінився процес контролю знань. Підвищилася довіра студентів до отриманих оцінок, у студентів немає підстав сумніватися в їх об'єктивності. Знизилося навантаження на викладача, йому залишається вибрати задачу і простежити за тим, щоб студенти вирішували її самостійно.

Студенти стали більше завдань вирішувати поза заняттями, справедливо очікуючи, що схожі завдання будуть і на тестуванні. Завдяки автоматичному збору статистики, викладач має краще уявлення про обсяг самостійної роботи того чи іншого студента.

Цінність запропонованої системи залежить від кількості завдань в її базі, тому розвиток системі ми вбачаємо в поширенні бази задач и переліку доступних мов програмування.