Мельник А. Алгоритми генерації множини задач та їх автоматичне розв'язання в прикладному програмному середовищі / Мельник А., Пасічник Р. // Вісник ТНТУ. — 2011. — Том 17. — № 2. — С.157-163. — (приладобудування та інформаційно-вимірювальні технології).

УДК 519.7:378.147

А. Мельник; Р. Пасічник, канд. фіз.-мат. наук

Тернопільський національний економічний університет

АЛГОРИТМИ ГЕНЕРАЦІЇ МНОЖИНИ ЗАДАЧ ТА ЇХ АВТОМАТИЧНЕ РОЗВ'ЯЗАННЯ В ПРИКЛАДНОМУ ПРОГРАМНОМУ СЕРЕДОВИЩІ

***Резюме**. Досліджено проблему формалізації процесу побудови та розв'язання задач з метою підвищення якості передавань знань та практичних навиків. Проаналізовано відомі підходи до вирішення проблеми автоматичної генерації практичних задач та їх розв'язання. Запропоновано метод генерації множини задач та їх автоматичного розв'язання в прикладному програмному середовищі. Показано, що даний метод володіє низкою переваг перед існуючими, що підтверджується експериментальними дослідженнями.*

***Ключові слова:** навчальний процес, практичні навички, автоматична генерація задач.*

A. Melnyk, R. Pasichnyk

AUTOMATIC GENERATION OF TASKS AND THEIR AUTOMATIC SOLUTIONS IN APPLICATION SOFTWARE

***The summary.** In this paper the problem of formalizing the process of building and solving tasks is analized, existing approaches to solve this problem is analized, described their main advantages and disadvantages. The method of automatic generation of tasks and their solutions in application software are proposed. This method has several advantages over existing, as evidenced by experiments.*

***Key words:** educational process, practical skills, automatic generation of tasks.*

Постановка проблеми. Однією із основних проблем процесу навчання є складність передавання знань і практичних навиків під час розв'язання задач. Суб'єкт навчання вирішує дану проблему власним шляхом: хто самостійно, хто з викладачем, а дехто взагалі намагається пропустити даний етап через складності або нерозуміння предмета. Важко переоцінити роль задач у процесі навчання. Під час розв'язання практичних завдань набуваємо нових знань, оскільки виникає необхідність знайомства з новою ситуацією, описаною в задачі, можливість застосування різноманітних теорій та методів до її розв'язання. При оволодіванні новим методом розв'язання задач у людини формується вміння вирішувати проблемні ситуації, а при достатньому тренуванні формуються практичні навички, які прискорюють процес розв'язку. Застосування задач в навчальному процесі дозволяє адаптувати суб'єкта навчання до практичної діяльності в майбутньому [1].

У процесі побудови, використання та перевірки розв'язків задач виникають великі часові та інтелектуальні затрати праці викладачів, що часто спонукають до скорочення варіативності під час перевірки засвоєння та використання знань і навиків. Виникає проблема формалізації практичних задач для їх автоматичного формування, пошуку розв'язків з метою отримання достатньої множини унікальних завдань, які будуть максимально близькими до практичних ситуацій. Їх використання в прикладних програмних середовищах дозволить суттєво економити час викладачів, що сприятиме підвищенню ефективності навчання.

Аналіз останніх досліджень і публікацій. На сьогодні питання автоматичної генерації задач для перевірки засвоєння практичних навичок мало досліджені [2]. Більшість відомих систем з даної проблематики мають закритий або комерційний характер. Серед відомих підходів важливо відзначити параметризовані тести [3]. Суть даного методу полягає в поданні різним студентам шаблонного завдання, яке відрізнятиметься певними параметрами, що генеруються автоматично. Кожен студент отримує індивідуальне завдання, а система за певною формулою чи алгоритмом, підставляючи параметри, отримує правильну відповідь для перевірки відповіді, введеної студентом. Недоліком підходу є його вузька предметна спрямованість, складність побудови шаблонів задач та змінність лише визначених параметрів.

Описані вище підходи не дозволяють отримувати варіативні набори завдань, а також вимагають значних часових затрат на їх реалізацію. Це породжує задачу розроблення технологій, яка б усувала згадані недоліки.

Метою дослідження є створення алгоритмів генерації множини задач та їх автоматичного розв'язання в прикладному програмному середовищі.

Постановка завдання. Виходячи з мети, необхідно виділити такі основні задачі дослідження:

- систематизацію основних проблем, які виникають у процесі формалізації структурних елементів задач і вибір шляхів їх розв'язання;
- розроблення моделей даних для формалізації структури основних технічних задач, яка спростить їх опрацювання в програмному середовищі;
- створення алгоритму генерації множини задач на основі формалізованого представлення її структурних елементів, а також алгоритму автоматичного контролю та перевірки згенерованих задач у програмному середовищі.

Задачі як засіб перевірки вмінь та навиків

Задача – це постановка або формулювання проблеми, яка ставиться суб'єкту навчання. Метою аналізу задачі є отримання розв'язку або пояснення причин його відсутності. Зупинимося детальніше на завданнях, які використовуються в процесі вивчення технічних дисциплін, оскільки вони простіше піддаються імітації та комп'ютерній обробці. Зазвичай формулювання задачі складається з умови і даних, які необхідно знайти. Між умовою та даними існують внутрішні відношення (зв'язки, залежності), характер яких визначає структуру задачі [4].

Процес розв'язання технічних задач містить:

- вибір стратегії розв'язку, яка включає розуміння умови та її аналіз, формулювання змісту, побудова моделі задачі та її розв'язок;
- визначення загальних і часткових правил, які можливо ефективно застосувати для отримання розв'язку задачі.

Найчастіше в навчальному процесі використовують задачі, формулювання яких представлено у вигляді тексту. Текст задачі може містити факти, важливість яких для розв'язку задачі є неочевидною, тому для отримання розв'язку необхідно вибрати тільки важливі. Це досягається в процесі побудови плану розв'язку і змушує переробляти інформацію, яка міститься в умові. В процесі розв'язку задачі потрібно встановити, який тип даних представлений в умові, оскільки від цього залежить вибір методу її розв'язання [4]. Виходячи зі структури задачі, виникає проблема формалізації її основних складових з метою подальшої комп'ютерної обробки та проведення автоматизованого тестування.

Модель формалізації задач для їх програмної обробки.

Формалізуємо структуру задач, які виникають у процесі вивчення дисциплін з елементами програмування, тобто тих, що містять засоби автоматичного опрацювання формалізованих лінгвістичних компонент. Такі інструменти дозволяють проводити перевірку розв'язків згенерованих задач. Згідно з концепцією, наведеною в [4], задача *Task* з технічних дисциплін складається з умови *Condition* , вхідних даних *Inp _ Value* та змінних, які необхідно знайти $\mathrm{Re}s\_Value$:

$$Task = \langle Condition,\ Inp\_Value,\ \mathrm{Re}s\_Value \rangle . \quad (1)$$

У розвиток концепції [4] умову задачі *Condition* опишемо так:

$$Condition = \langle Ft,\ Tf,\ Tfil \rangle , \quad (2)$$

де *Ft* – множина текстових представлень умови задачі; *Tf* – множина допустимих типів значень вхідних і результуючих змінних, *Tfil* – множина текстових представлень для уточнення умови задачі.

$$Ft = \{Id,\ Nmf,\ Lg\_Int, Tcv, Ttv\}, \quad (3)$$

де *Id* – унікальний код, призначений для ідентифікації варіанта представлення; *Nmf* – системна функція, яка програмно реалізована в конкретному середовищі програмування; *Lg _ Int* – лінгвістична інтерпретація *Nmf* , яка дозволяє отримувати задачу "на природній мові"; *Tcv* – кількість вхідних змінних, які опрацьовуються конкретною реалізацією *Nmf* ; *Ttv* – ідентифікатор типу результуючих і вхідних змінних.

$$Tf = \{Id, Type\}, \quad (4)$$

де *Type* – тип значень вхідних і результуючих змінних, наприклад, string, integer, date тощо.

$$Tfil = \{Id,\ Nmfil,\ Lg\_Int, Tcv, Ttv\}, \quad (5)$$

де *Nmfil* – функція, яка програмно реалізована в конкретному середовищі програмування і використовується для фільтрування вхідних змінних.

Разом із функціями, які використовуються в якості уточнюючих елементів умови задачі, використовується множина операцій над вхідними змінними *Op* , яка описується відношенням

$$Op = \{Id,\ Oper,\ Lg\_Oper\}, \quad (6)$$

де *Oper* – назва операції; *Lg _ Oper* – лінгвістична інтерпретація операцій *Oper* .

Виходячи з умови задачі *Condition* , формується множина допустимих значень вхідних змінних *Inp _ Value* та відповідно результуючих $\mathrm{Re}s\_Value$:

$$Inp\_Value = \{Id,\ Value,\ Id\_Tf\}, \quad (7)$$

$$\mathrm{Re}s\_Value = \{Id,\ Value,\ Id\_Tf\}, \quad (8)$$

де *Value* – значення змінної; *Id _ Tf* – ідентифікатор типу вхідних змінних.

Із відношень (2)–(8) отримаємо узагальнене представлення для *Task* , яке описується п'ятіркою

$$Task = \langle Ft,\ Tf,\ Tfil,\ Inp\_Value,\ \mathrm{Re}s\_Value \rangle , \quad (9)$$

що дозволяє формалізувати основні задачі, які застосовуються для перевірки засвоєння знань з технічних дисциплін.

Алгоритм генерації множини задач.

На основі формалізації (1)–(9) запропоновано алгоритм генерації множини задач, який дозволяє отримувати варіативні набори з достатнім рівнем педагогічної та практичної цінності. Даний алгоритм представимо у вигляді такої послідовності кроків:

1. Із відношення Ft випадковим чином вибираємо Nmf, та відповідно Lg_Int, тобто використовуємо процедуру

$$Rand(\{Id, Nmf, Lg\_Int\}), \tag{10}$$

де $Rand$ – операція вибору випадкового елемента; Lg_Int – представлення умови задачі на природній мові; Nmf – внутрішньо-програмне представлення.

2. До відношення $Tfil$ застосовуємо процедуру

$$Rand(\{Id, Nmfil, Lg\_Int\}) \tag{11}$$

за умови виконання такого правила:

$$Tfil(Tcv) = Ft(Tcv), \tag{12}$$

$$Tfil(Ttv) = Ft(Ttv), \tag{13}$$

забезпечуючи таким чином узгодженість кількості вхідних параметрів та їх типів.

3. Формуємо множину вхідних змінних із відношення Inp_Value, тобто $Rand(\{Value\})$, причому

$$Inp\_Value(Id\_Tf) = Ft(Ttv), \tag{14}$$

$$Inp\_Value(Id\_Tf) = Filt(Ttv). \tag{15}$$

Описані вище кроки дозволяють отримувати варіативні множини задач, які можна розв'язувати у конкретно вибраному програмному середовищі. Далі розглянемо процес представлення та автоматичної перевірки правильності отриманих розв'язків.

Алгоритм представлення, автоматичного контролю та перевірки згенерованих задач у програмному середовищі.

Після реалізації формалізованого представлення задачі (1)–(9) та описаного вище алгоритму генерації множини задач виникає необхідність розроблення алгоритму, який дозволяє проводити автоматизований контроль розв'язання згенерованої задачі в процесі оцінювання вмінь та навичок суб'єкта навчання. Запропонований алгоритм представлено на рисунку 1.

На першому кроці реалізуємо алгоритм генерації умови задачі. В прикладі, що на рисунку 1, використовується формалізоване представлення задачі для знаходження максимального елемента масиву. На другому кроці проводимо уточнення умови задачі з метою ускладнення її структури. На третьому кроці проводимо ідентифікацію структури вхідної інформації, кількості вхідних змінних задачі, а також їх типу. На четвертому кроці необхідно представити вхідну інформацію, яка буде використовуватися для автоматичного опрацювання, для прикладу, це може бути масив з даними, таблиця з даними в базі даних, різноманітні структури. Вибір представлення вхідної інформації залежить від інтерпретатора, який буде її опрацьовувати.

На п'ятому кроці автоматично генеруємо завдання для інтерпретатора на основі раніше згенерованої умови. В якості інтерпретатора може бути компілятор мови програмування, SQL-інтерпретатор, або створений власно інтерпретатор, який здатний виконувати послідовність певних службових операцій. На шостому кроці генеруємо завдання для студента на природній мові, шляхом комбінування відповідно до

формалізованих структур задачі та заміни службових конструкцій команд їх аналогами на природній мові. На цьому кроці проводиться порівняння результатів, які повертає інтерпретатор на запит студента та автоматично згенерований запит зі встановленням правильності відповіді.

Даний підхід дозволяє автоматично генерувати завдання практичного спрямування із варіативною структурою. В процесі його використання суттєво економляться викладацькі ресурси, які доцільно направити на формалізацію нових типів завдань та удосконалення змісту навчального курсу.

Експериментальні дослідження. На основі описаних вище підходів було проведено ряд експериментальних досліджень, які дозволили підтвердити коректність та оцінити якість згенерованих практичних задач. Дані алгоритми лягли в основу реалізації web-орієнтовної системи керування процесом навчання на факультеті комп'ютерних інформаційних технологій Тернопільського національного економічного університету. На рисунку 2 продемонстровано згенероване практичне завдання з дисципліни «Основи Web-програмування» в середовищі системи програмування PHP версії 5.3.1 та метод її розв'язання, який пропонує студент у ході вирішення поставленого завдання.

Перевагою запропонованих алгоритмів над відомими рішеннями є те, що крім автоматичної генерації множини вхідних змінних, автоматично генерується умова задачі, що дозволяє отримувати достатньо варіативні набори задач для практичної перевірки засвоєних знань.

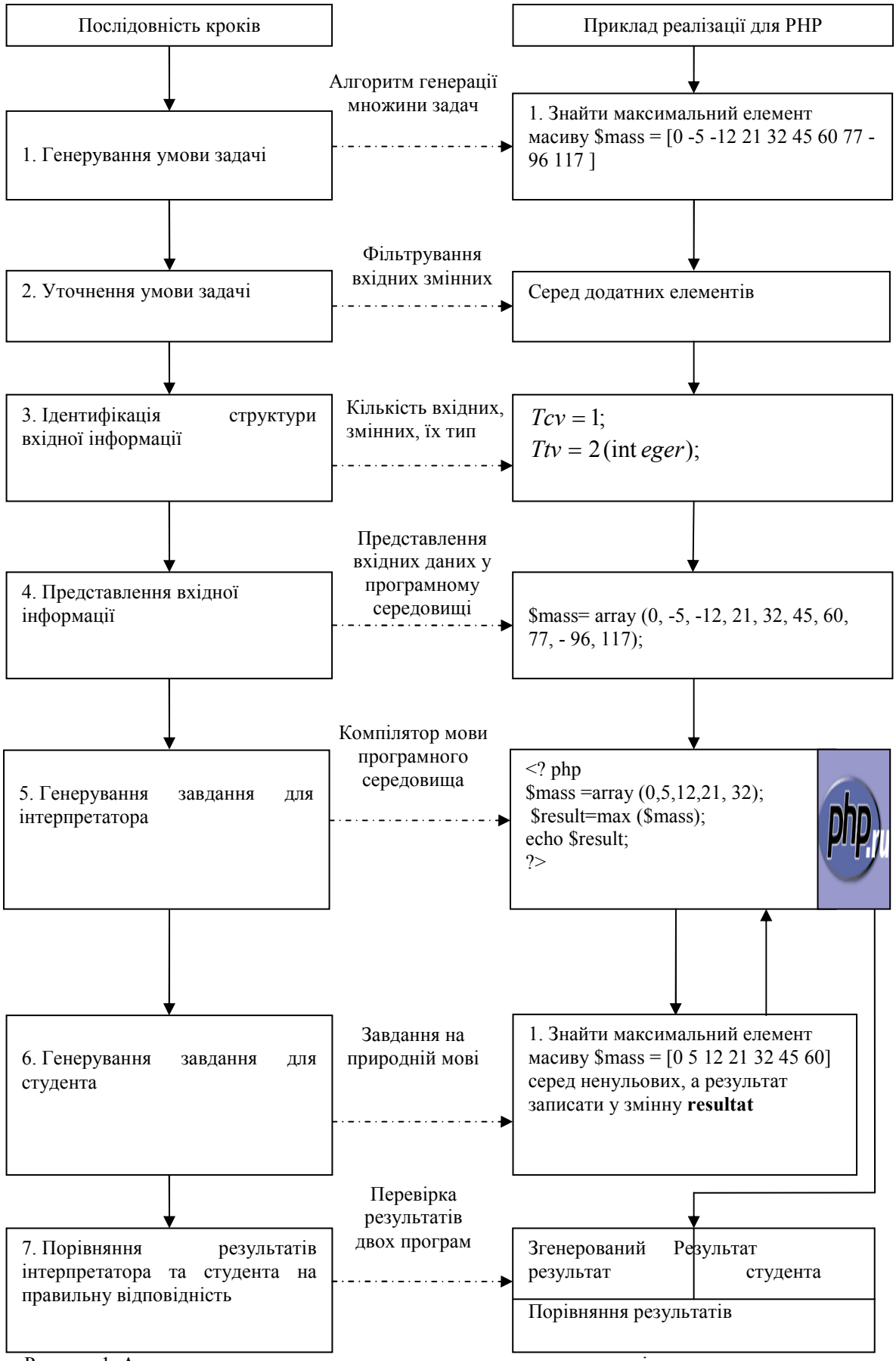

Рисунок 1. Алгоритм представлення, автоматичного контролю та перевірки згенерованих задач у програмному середовищі

Задано наступне завдання, небхідно написати програму для його вирішення

1 . Знайти кількість елементів масиву $mass = [0 -1 0 3 8 15 24 35 48 63 80 99 120 143 168] серед додатніх , а результат записати у змінну **resultat**

Вікно для написання програми:

```
<?php
$c='$var >0';
// функція фільтрування
// позитивних елементів масиву
function plus($var)
{
    return ('$c');
}
// ініціалізація масиву
$mass = array (0, -1, 0, 3, 8, 15, 24, 35, 48, 63, 80, 99, 120, 143, 168);
// побудова масиву додатніх елементів
$mass_res =array_filter($mass, "plus");
// знаходження кількості додатніх елементів масиву
$result=count($mass_res);
echo "Кількість додатніх елементів мавису складає ", $result;
?>
```

Перевірити результат

Рисунок 2. Приклад згенерованої задачі для PHP мови програмування та розв'язку її студентом

Висновки. В процесі проведених досліджень отримано такі наукові та практичні результати:

- на основі реляційної моделі даних побудовано формалізацію постановки навчальних задач технічного плану. Реалізовано формалізацію засобів розв'язання формалізованих задач у довільному програмному середовищі;

- на основі проведеної формалізації вперше запропоновано алгоритми генерації задач варіативної структури із автоматизацією перевірки правильності їх розв'язання.

Література

1. Электронная хрестоматия по методике преподавания математики [Электронный ресурс]. – Режим доступа: http://fmi.asf.ru/Library/Book/Mpm/9b.html.
2. Мельник, А.М. Методи та засоби автоматичної генерації тестових завдань різних форм [Текст] / А.Мельник // Інноваційні комп'ютерні технології у вищій школі: Матеріали 2-ї науково-практичної конференції, 23–25 листопада 2010 р. – Львів: Видавництво Національного університету "Львівська політехніка", 2010. – С. 147–152.
3. Мельник, А.М. Метод генерації тестових завдань на основі системи семантичних класів [Текст] / А.М. Мельник, Р.М. Пасічник // Вісник ТДТУ. – 2010. – Т. 15, № 1. – С. 187–193.
4. Дмитриев, В.М. Формализованное представление задач для компьютерного моделирования [Текст] / В.М. Дмитриев, А.Ю. Филиппов, О.Н. Шарова // Вестник Московского городского педагогического университета. – 2004. – №3. – С. 53–59.

Отримано 22.03.2011