Тестове за оценка на знанията по програмиране

Николай Киров

24 юни 2019 г.

Абстракт

Проверката на знанията и уменията на ученици и студенти с тестове е важна и добре разработена тема. В тази статия се предлагат специален вид тестове, свързани със спецификата на учебните дисциплини програмиране и стуктури от данни. Особено внимание се обръща на обучителния елемент в процеса на подготовката, провеждането и проверката на тестовете. Представена е и софтуерна система за генериране на този вид тестове, за проверката им и получаване на данни за анализ на резултатите, която е със свободен достъп.

1 Въведение

Преподаването на програмиране е една трудна и отговорна задача. Това се обуславя от спецификата на програмирането, предварителната подготовка на обучаемите, и др.

Усвояването на учебния материал е свързано с: научаването на принципите на програмирането като терминология, идеология и техники, и придобиване на практически умения за писане на програми. Затова преподаването на програмиране е съставено от две части – теоретична и упражнения. Съответно проверката и оценката на знанията и уменията по прграмиране се състои от 3 компонента: тест; домашни работи за писане на програми по зададени индивидуални задачи; контролна работа за писане на работеща програма на компютър.

Целта на теста е да се стимулират обучаемите за преглед още веднъж на учебния материал, да се повиши умението за търсене на определена информация в учебници, за осъзнаване на понятия, ... и техники, за самостоятелно решаване на малки задачи и на последно място за проверка и оценка на придобитите до този момент знания и умения.

2 Структура на индивидуалния тест

Тестовете се състоят от задачи с неопределен брой верни отговори. За всяко предложение за отговор обучаемият има 3 възможности за определянето му: знам, че това е (верен) отговор на задачата; знам, че това не е отговор на задачата (дистрактор); не мога да преценя, не знам дали предложението е отговор на задачата.

Всеки индивидуален тест се състои от 10-15 задачи и всяка тестова задача има предложения за 4 отговора, маркирани с a), b), c) и d). Броят на верните отговори за всеки въпрос може да бъде 0, 1, 2, 3 или 4. Възможностите за верен/неверен (в/н) отговор или отговор/дистрактор според четирите предложения са: всички са верни (вввв); три са верни (ввнн, вннв, нвнв, ннвв); два са верни (ввнн, внвн, вннв, нввн, нвнв, ннвв); един е верен (вввн, ввнв, внвв, нввв); няма верен (нннн). Броят на вариантите за отговор на задачата е $4^2 = 16$. Ако обучаемият отговаря

по случаен начин само с двете възможности верен/неверен (отговор/дистрактор), вероятността да са правилни и четирите отговора на един въпрос е 1/16.

Този вид тестове имат своите предимства и недостатъци, като в конкретната предметна област са добър инструмент както за проверка и оценка на знания, така и за придобиване на нови знания от обучаемите [4].

3 Подготовка на тестовете от преподавателя

Подготовката на тестовете включва формулиране на задачите и подбор на отговорите и дистракторите. Тъй като всеки индивидуален тест съдържа по 4 предложения на задача, то всяка задача трябва да има подготвени най-малко 4 отговора. За да се избегне преписване от съседа, е добре да се подготвят поне по десетина предложения за отговори на всяка задача. Практиката показва, че около 20 предложения са напълно достатъчни за генериране на индивидуални тестове с относително различни и разбъркани отговори и дистрактори.

Задачите и предложенията за отговори формират текстов файл със строго определена структура. Този файл служи за вход на програма `test_generator`, която генерира индивидуалните тестове — толкова, колкото е броят на обучаемите. Използва се генератор на случайни числа за избор на задачите (ако са повече от включените в индивидуалния тест) и на предложенията за отговори на всеки индивидуален тест. Всеки такъв тест има уникален 4-цифров номер и се състои от 10 до 20 задачи, с по 4 предложения за отговор на всеки въпрос. Най-добре е броят на въпросите да е 15 по две причини: първо, при добра подготовка, времето от два учебни часа е достатъчно за този обем и второ, лесно се преобразуват получените точки на теста в оценка по шестобалната система.

Програмата `test_generator` създава LaTeX файл с индивидуалните тестове (out.tex), файл с отговорите на всеки индивидуален тест (tab.tex) и трети файл (data.tex), който се използва от програмата за обработка на тестове `test_checker`. Файлът с индивидуалните тестове (tab.tex) се включва в заглавен файл на LaTeX (main.tex), след което се компилира (напр. със системата MiKTeX) и така се произвежда pdf файл, готов за отпечатване на индивидуалните тестове.

4 Подготовка на обучаемите за теста

Поне една седмица преди датата на провеждане на теста задачите и по две примерни предложения за отговор (верен и грешен) стават достъпни за обучаемите (напр. публикуват се в Интернет). По този начин обучаемите имат възможност да се запознаят с теста предварително. Добра практика е няколко дни преди провеждане на теста да се организира консултация, на която се разглеждат подробно задачите на теста и се обсъждат възможни отговори. Дават се насоки за начините за подготовка за теста, напр. кой учебен материал е включен в този тест, какви програми могат да се напишат предварително, които да улеснят отговорите на някои задачи и т.н.

5 Провеждане на теста

По време на провеждане на теста на обучаемите използват свободно лекции, учебници и всякакви други печатни материали. Често е разрешено и използване на компютри – или само за четене или и с използване на среда за програмиране (което е и моята практика през последните години).

По-подробно за феномена преписване. Класическият подход за проверка на знание или умения (контролни, тестове, явяване се на изпит и др.) е на обучаемия да се се забрани използването външни източници на информация. Това означава, че учебният материал се запомня и после или се възпроизвежда или се прилага — например за решаване на задачи или писане (на хартия) на програми. В специфичната дейност програмиране, която е основна цел като придобито умение, крайният продукт е компютърна програма. Написването и до голяма степен е творчески процес (който е основен предмет на оценяването) и ограничаването на използваната информация е напълно безсмислено. Такава е и практиката във фирмите от ИТ сектора. Когато се даде задача за написване на софтуер, изискването обикновено е да се намери литература и да се създаде софтуера според най-новите теории и практики в специфичната област. Този подход следваме и ние при провеждането на теста.

Целта от провеждане на теста е не само да се проверят знанията на обучаемите, а и да се стимулират (принудят) да отворят учебника, да потърсят конкретна информация там, да направят логическата връзка между задачите на теста и написаното в учебника и т.н. Да проверят с компютъра синтаксис или действието на някаква програмна конструкция.

В часа за теста всеки обучаем получава лист хартия A4 със задачите на индивидуалния тест и още една бланка за отговорите. Всички тестове са различни (поне според подредбата на задачите и предложенията за отговори), което до голяма степен елиминира елемента на преписване от съседа при попълване на теста. В бланката за отговори, срещу всяко предложение за отговор (a), b), c), d)) обучаемият трябва да даде положителен отговор (да, вярно, истина; yes, correct, true; 1) или да даде отрицателен отговор (не, невярно, неистина; no, incorrect, false; 0) или да не отговори (–, не знам; или нищо; I don't know; -). По този начин се диференцира знанието на обучаемия: „знам, че това го знам“ или „знам, че това не го знам“ и до някаква степен се елиминира даването на случаен отговор.

Времето да провеждане на теста е 2 учебни часа. Обучаемият предава двата листа хартия – индивидуалния тест и попълнената бланка с отговорите.

6 Проверка на индивидуалните тестове от преподавателя

При проверката на индивидуален тест всяка задача получава тестови точки в интервала $[-4, 4]$. Те са сума от точките за всеки отговор, които могат да бъдат: $+1$ когато студентът е дал правилен отговор (да или не); -1 при неправилен (обратен) отговор; 0 при отговор "не знам".

Често в литературата се срещат препоръки от вида: "Не се препоръчва приписването на наказателни точки за неправилен отговор, тъй като тогава изпитваните се страхуват да отговарят, ако не са напълно убедени, и така не показват истинското ниво на своите знания/умения." [3]. Аргументът за неспазване на това правило е, че програмирането е специфична дейност и знанията в тази област трябва да са категорични — програмистът трябва да има ясна представа какво знае и какво не знае. Една съвсем дребна грешка в компютърна програма (напр. липса на запетая) може да доведе да абсолютно непредвидими последици — от „невинна“ правописна грешка в някой текст до застрашаване на човешки живот или загуба на космически кораб. Освен това възможността за справки в учебници и използване на компютър и Интернет дават възможност на обучаемия да намери точния отговор на всяка задача, като стимулира и развива уменията за търсене на необходимата информация в книги и Интернет.

Проверката на тестовете се извършва ръчно с помощта на таблицата и/или автоматично с програмата `test_checker`. Въвеждането на отговорите на студентите може да е ръчно с програмата или да се използва специален шаблон и скенер. Програмата създава текстов файл (`save.txt`), съдържащ резултата от проверката.

7 Проверка на индивидуалния тест от обучаемия

В началото на следващото занятие се обявяват резултатите от теста и се връщат индивидуалните тестове и проверените бланки с отговорите на обучаемите. Всеки трябва внимателно да види какво е сгрешил, да прецени дали е съгласен с отбелязаните грешки и ако нещо не му е ясно, да попита. Целта на тази проверка е обучаемите да осъзнаят грешките си, което очевидно спомага за по-доброто усвояване на учебния материал. Освен това, тъй като не е лесно да се формулират кратко и точно всички тестови задачи, може да има и не съвсем ясни предложения за отговори, практиката е след дискусия да се приемат мненията на обучаемите, които интерпретират по по-различен начин някоя задача или някой отговор и да се увеличават получените точки.

8 Анализ на теста от преподавателя

Целта на този анализ е на основата на статистики от проверените тестове да се направят изводи за трудността на въпросите и степента на подгготовката на обучаемите по съответните теми.

Програмата `test_checker` създава файл (`data_result.txt`), аналогичен на файла, съдържащ всички въпроси и отговори (вход за `test_generator`) с добавени данни за резултатите. След всеки отговор на всеки въпрос са дадени 6 числа във формат $a_1 : a_2 : a_3 b_1 : b_2 : b_3$, където: a_1 е брой тестове, в които има (се е паднал случайно) този въпрос и отговор; a_2 е брой тестове с верен отговор; a_3 е брой тестове с грешен (обратен) отговор. %enditemize За втората тройка числа $b_1 : b_2 : b_3$ смисълът е същия, но се отчитат само тестове, които имат не по-малко от половината на максималния брой точки.

9 За задачите и предложенията за отговори

Видове задачи:
– за понятия, дефиниции, отношения,
Пример ООП: Отбележете верни/неверни твърдения за обработка на изключения.
(да) Когато дадена функция открие грешка, тя може да изхвърли изключение към някоя друга част на програмата, чиято работа е да обработва грешки.
(не) Ключовата дума `throw` показва, че изпълнението на функциятя спира незабавно и управлението се предава на извикващата функция.

– за синтаксис и/или логика на операции, оператори и др. конструкции на езика
Пример ООП: Дадени са свързан списък и итератор:

```
list<int> bip;
list<int>::iterator it;
```

Списъкът съдържа 10 елемента и итераторът сочи втория елемент на свързания списък. Верни ли са (синтактично и/или логически) дадените изрази?
(да) `*it > 0`
(не) `*it != *bip.end()`

– за резултата от работата на оператори, функции или малки програми, дадени в теста
Пример ООП: Даден е класа:

```
class  Btree{
public:
```

```
    Btree();
    Btree(const Btree& b);
    ~Btree();
    Btree& operator=(const Btree& b);
private:
...
};
```

и обект b от този клас. Проверете твърденията за оператори от функция `main`:
(да) В оператора `Btree b1(b);` се извиква конструктора за копиране.
(не) В оператора `Btree b1 = b;` се извиква предефинираната операция присвояване.

– за твърдения, работа, резултат от промяна в разглеждяни на занятия тектове на програми *Пример П*: Може ли функцията `raise_salary`, дефинирана във файла `raisesal.cpp`, да бъде извикана по следния начин (при условие, че е дефиниран обект `harry` от клас `Employee`): (да) `raise_salary(harry, 5)`; (не) `raise_salary(harry)`;

– за реализация на алгоритми върху конкретен пример, оценки за сложност, *Пример СД*: Дадено е AVL дърво:
`44(17(-,32),78(50(48,62),88))`. Външните възли не са включени в представянето. Определете дали даденото след операцията AVL дърво е получено като резултат от прилагане на операцията.
(да) `insertItem(54) 44(17(-,32),62(50(48,54),78(-,88)))`
(не) `removeElement(32) 44(17,78(50(48,62),88))`

Въпросителното изречение към всяка задача, което да предполага двоичен отговор (да/не) може да бъде формулирано по два начина: директно питане за определяне вярно/невярно или да се напише резултата от викане на функция, стойност на израз, резултат от пресмятане. Във втория случай функцията е предикат (връща true или false), изразът има стойност 1 или 0, резултатът от пресмятането е отново 1 или 0, true или false.

10 Заключение

В условията на относителна свобода при провеждането на теста, обучаемите биха могли най-адекватно да отразят своите знания в правилни отговори на теста. Фактът, че всяко твърдение може да се провери в записки, учебници, Интернет, с писане на код и използване на компилатор освен, че допринася за увеличаване на знанията на обучаемия, дава и добра оценка за постиженията му. Резултатът от теста отразява не запомненото, а действително осъзнатите знания по дадената тема. Необходимостта от предварителна работа по теста (обучаемите знаят, че ако се отидат „просто така“, няма да успеят дя „минат“ теста) е допълнителен стимул да се прочете отново учебния материал, да се потърси къде би могло да се намери отговор на всяка задача и да се проиграят някои от въпросите на теста (с писане на код и предположения за възможни отговори).

Текстовете на програмите `test_generator` и `test_checker`с примери и кратки описания са свободно достъпни в GitHub ([6], [5]): Написани са на C++ с платформата Qt. Кратко описание на тези програми, както и резултати от проведени тестове може да се видят в [4].

Литература

[1] Angelov A. & V. Dzivev (2017). Test Generator, *Mathematics and Informatics*, 4, 344-350. [Ангелов, А. & В. Дзивев (2017). Генератор на тестове, *Математика и информатика*, 4,

344-350.]

[2] Bankov, K. (2012). *Introduction in Testology.* Sofia:Izkustva. [Банков, К. (2012). *Увод в тестологията*, София:Изкуства.

[3] Bizhkov, G. (1992). *Theory and Methods of teaching tests*, Sofia:Education. [Бижков, Г. (1992). *Теория и методика на дидактическите тестове*, София:Просвета.]

[4] Kirov N. (2014). A test system for checking and evaluation the students' programming knowledge, *Proc. 10th Ann. Int. Conf. Comp. Sci. & Educ. Comp. Sci.*, Albena, Bulgaria (ISSN 1313-8624), 125-134.

[5] `htps://github.com/nkirov/tests_checker`

[6] `https://github.com/nkirov/tests_generator`

[7] `http://nikolay.kirov.be/`

Tests for assessing of knowledge in programming

Abstract. Assessing the knowledge and skills of students with tests is an important and well-developed topic. This article provides a special type of tests related to the specifics of the courses programming and data structures. Particular attention is paid to the training elements in the processes of preparing, conducting and checking the tests. A software system for generating this type of test, for checking and obtaining data for analysis of the results is presented in the article and is freely available.

Dr. Nikolay Kirov, Assoc. Prof.
Department of Computer Science
New Bulgarian University
21, Montevideo Blvd.
1618 Sofia, Bulgaria
E-mail: nkirov@nbu.bg